UNIT-I

SCRIPTING

> Web page Designing using HTML

> Scripting basics
  - Client side and Server side Scripting

> Java Script

> Object

> names

> literals

> operators and expressions

> statements and features

> events

> windows

> documents

> frames

> data types

> built-in functions

> Browser object model

> verifying forms

> HTML 5, CSS3 & HTML5 Canvas

> Web site Creation using tools.

Part - I : Scripting

⇒ Web page Designing using HTML :-

-* Creating an HTML document :

> The first step of creating a web page is to create an HTML document.

> HTML is known as Hyper Text Markup Language, an HTML document can be created in any text editor even on a notepad.

Steps :

step-① :- Open your text editor such as Notepad Sublimetext, etc...

step-② :- Write the code given below in the text editor.

HTML

```
<! DOCTYPE html>
<html>
<head>
 <title> first HTML file </title>
 </head>
  <body>
   Hello Everyone !!
   </body>
   </html>
```

Step-③ :- Save this file with the .html /.htm extension

Step-④ :- Open that file with any browser. The Output will be displayed.

→ Designing web page :-

1. <h1>...</h1> to <h5>...</h5> :-

* These tags are called header tags and they are used to give heading to your webpage of different sizes.

* <h1>...</h1> being the largest Heading to <h5>...</h5> being the smallest Heading.

2. Bold Tag (<strong>...</strong> (or)<b>....</b>):-

* These tags are used to make text look bold.

3. Italic Tag (<i>...</i> (or) <em>...</em>):-

* These tags are used to make text look italics.

* The only difference between <i> and <em> is that <em> semantically emphasis on important text (or) word whereas <i> tag is used to make is just used to make text italics.

4. Ordered List (<ol> ....</ol>):-

* The HTML <ol> tag defines an ordered list.

* An ordered list can be numerical (or) alphabetical, An ordered list starts and ends with a <ol> tag.

5. Unordered List (<ul>...</ul>):-

* It displays elements in bulletin form.

* An Unordered list starts with <ul> tag and each item starts with <li> tag.

6. Image Tag:-

* If we need to add an image to our website we need to use the following syntax.

* <img src = "filename" alt = " name / bit about image ">.

7. Anchor Tag :-

* This tag is mainly used to connect one webpage to another.

⇒ Scripting basics - Client side and Server Side Scripting :-

1> Client side Scripting :-

* Client side scripting source code is visible to the user.
* It usually depends on the browser and its version.
* It runs on the user's computer.
* It does not provide Security for data.
* In this client side Scripting HTML, CSS, and java script are used.
* No need of interaction with the server.

2> Server side Scripting :-

* Server side Scripting source code is not visible to the user.
* It does not depend on the Client side server.
* It runs on the web server.
* It provides more security for data.
* In this server side Scripting PHP, Python, Java, Ruby are used.
* It is all about interacting with the servers.

⇒ JAVASCRIPT OBJECTS :-

* Javascript objects are the most important data type and form the building blocks for modern Javascripts.

* The 'object' class represents the javascript data types.

Syntax: objectName. methodName().

The complete list of Javascript object is listed below:

1. Javascript object properties:- A Javascript property is a member of an object that associates a key with a value.

*Instance properties: An instance property is a property that has a new copy for every new instance of the class.

2. Javascript object Methods:- Javascript methods are actions that can be performed on objects.

* Static Methods: If the method is called using the objects class itself then it is called a static method of object.

* Instance Methods: If the method is called on an instance of a date object then it is called an instance method.

⇒ JAVASCRIPT VARIABLE NAMES :-

* Javascript Variable are the building blocks of any programming language.

* In Java script, variables can be used to store reusable values.

* Javascript identifiers :-
Javascript variables must have unique names, These names are called identifiers.

* There are some basic rules to declare a variable in Javascript :-

* These are case-sensitive
* Can only begin with a letter, underscore ("_") or "$" symbol.
* It can contain letters, numbers, underscore (or) "$" symbol.
*. A variable name cannot be a reserved keyword.
*. We can declare variables in javascript in 3 ways:

i. Javascript var keyword.
ii. Javascript let keyword.
iii. Javascript const keyword.

i. Javascript Var :-
* Javascript Var is a keyword used to declare variables in javascript that are function scoped.
* Before the introduction of ES6 all the keywords in javascript were declared with only "Var" keyword.

Syntax: Var VariableName = ValueOfVar ;

ii. Javascript let :-

* Javascript let is a keyword used to declare variables in javascript that are block scoped.

* Two new keywords were added in the ES6 (or) ES2015 version of Javascript.

Syntax :- let variable-name = value;

iii. Javascript const :-

* ES2015 (ES6) introduced the const keyword to define a new variable.

* The different in const variable declaration to others is that it cannot be re-assigned.

Syntax : const const-name;

```
         const x;
```

⇒ JAVASCRIPT LITERALS :-

* Literals are the way you represent values in Javascript.

* These are fixed values that you literally provide in your application. Source, and are not variables.

Examples :

1. 42
2. 3.14159
3. "To be (or) not to be".

* Integers :-

* Integers can be expressed in decimal, hexadecimal (or) octal format.

* A decimal integer literal consists of a sequence of digits without a leading 0.

* Floating point literals :-

* A Floating point literal can have the following parts: a decimal integer, a decimal point (" . "), a fraction, an exponent, and a type suffix.

* The Exponent part is an "e" (or) "E" followed by an integer, which can be signed.

examples: 1) 3.1415
2) -3.1E12
3) .1e12
4) 2E-12.

* Boolean Literals:-

* The boolean type has two literal values : true and false.

* String Literals :-

* A String literal is zero (or) more Characters enclosed in double (") (or) Single (') quotes.

* A String must be delimited by quotes of the same type; that is, either both single quotes (or) double quotes.

examples:-

1) " blah"
2) ' blah'
3) " 1234"
4) " one line \n another line".

⇒ JAVASCRIPT OPERATORS :-

① Arithmetic Operators :- These are the Operators that operate upon the numerical values and return a numerical value.

> Addition (+) :- Addition '+' operator performs addition on two operands, this '+' operator can also be used to concatenate strings.

Ex: y = 5+5 gives y = 10.

> Subtraction (-) :- Subtraction '—' operator Performs Subtraction on two operands.

Ex:- y = 5-3 gives y = 2.

> Multiplication (*) :- Multiplication '*' operator performs multiplication on two operands.

Ex:- y = 5*5 gives y = 25.

> Division (/) :- Division '/' operator performs division on two operands.

Ex :- y = 5/5 gives y = 1.

> Modulus (%) :- Modulus '%' operator gives a remainder of an integer division.

Ex:- A % B means remainder (A/B)

y = 5%4 gives y = 1.

> Exponentiation (**) :- Exponentiation '**' operator give the power of the first operator raised to the second operator.

Ex :-

y = 5**3 gives y = 125

> Increment (++) :- Increment '++' operator increases an integer Value by one.

Ex:- Let A = 10 and Y = A++ then A = 11 , Y = 10.

if A = 10 and Y = ++A then A = 11 , Y = 11.

> Decrement (--) :- Decrement '--' operator decreases an integer value by one.

Ex:- Let A = 10 and Y = A-- then A = 9 , Y = 9.

if A = 10 and Y = --A then A = 9 , Y = 9.

> Unary (+) :- Unary '+' is the fastest and preferred way of converting something into a number.

Ex:- +a means "a" is a positive number.

> Negation (-) :- Negation '-' operator gives the negation of an operand.

Ex:- -a means "a" is a negative number.

②. Assignment Operators :- The Assignment operator assigns the right operand Value to the left Operand.

> Addition Assignment (+=) :- Sums up left and right operand values and then assigns the result to the left operand.

Ex:- Y+ = 1 gives Y = Y+1.

> Subtraction Assignment (-=) :- It Subtracts the right side Value from the left side value and then assigns the result to the left operand.

Ex:- ~~Y/7~~ Y-=1 gives Y=Y-1.

> Multiplication Assignment (* =) :- It multipli a Variable by the value of the right operand an assigns the result to the variable.

Ex:- Y* =A is equivalent to Y=Y*A

> Division Assignment (/=) :- It divides a Variable by the value Of the right operand and assigns the result to the Variable.

Ex:- Y/=A is equivalent to Y=Y/A.

> Modules / Remainder Assignment (% =) :- It divides a variable by the value of the right operand and assigns the remainder to the Variable.

Ex:- Y % = A is equivalent to Y=Y%A.

> Exponentiation Assignment (** =) :- This raises the value of a variable to the power of the right operand.

Ex:- Y** =A is equivalent to Y=Y**A

> Left shift Assignment (<<=) :- It moves the specified amount of bits to the left and assigns the result to the Variable.

Ex:- Y<<=A is equivalent to Y=Y<<A.

> Right shift Assignment (>>=) :- It moves the Specified amount of bits to the right and assigns the result to the Variable.

Ex:- Y>>=A is equivalent to Y=Y>>A.

③ Comparision Operators :- Comparision Operators are mainly used to perform the logical operations that determine the equality (or) different between the values.

> Equality (==) :- Compares the equality of two operands, if equal then the condition is true otherwise false.

Ex:- Y=5 and X=6
Y==X is false.

> Strict equality (===) :- Compares the equality of two operands with type, if both value and type are equal then the condition is true otherwise false.

Ex:- Y=5 and X='5'
Y===X is false.

> Inequality (!=) :- Compares inequality of two operands, True if operands are not equal.

Ex:- Let X=10 then X!=11 is true.

> Strict inequality (!==) :- Compares the inequality of two operands with type, if both value and type are equal then the condition is true otherwise false.

Ex:- Let X=10 then x!=='10' is true.

④ Logical Operators :- These operators are used to determine the logic between variables (or) values.

> Logical AND (&&) :- It checks whether two operands are non-zero, if yes then return the last operand when evaluating from left to right.

Ex:- Y=5 and X=6
Y && X is 6

>Logical OR (||) :- It checks whether two Operand are non-zero, if yes then return the first operand when evaluating from left to right

Ex:- Y=5 and X=0

Y||X is 5

> Logical NOT (!) :- It reverses the boolean result of the operand.

Ex:- Y=5 and X=0.

!(Y||X) is false.

⑤. Bitwise Operator :- The bitwise operator in Javascript is used to convert the number to a 32-bit binary number and perform the bitwise operation.

> Bitwise AND (&) : The operator returns true only if both the operands are true.

Ex:- A=6, B=1

A&B=0

> Bitwise OR (|) :- The Operator returns true even if one of the operands is true.

Ex:- A=6 , B=1

A|B=7

> Bitwise XOR (^) :- The Operator returns true if both operators are distinct.

Ex: A=6 , B=1

A^B=7

> Bitwise NOT (~):- This operator is used to invert the boolean value of the operand.

Ex:- A = 6

-A = -7

> Bitwise Left Shift(<<):- In this two operators are used where the first operand is the number and the Second operand is the number of bits to shift to the left.

Ex:- A = 6 , B = 1

A << B = 12

> Bitwise Right Shift (>>):- In this two operators are used where the first operand is the number and the Second operand is the number of bits to Shift to the right.

Ex:- A = 6 , B = 1

A >> B = 3.

⇒ JAVASCRIPT EXPRESSIONS :-

* Javascript expression is a valid set of literals, variables, operators, and expressions that evaluate a single value that is an expression.

* This single value can be a number, a string, (or) a logical value depending on the expression.

* Javascript primary Expressions :

i> this keyword :- that defines the current line of code's execution context.

ii> Await function :- Checks that we are not breaking the execution thread.

iii> Grouping operator :- the grouping operator consists of a pair of parentthesis around an expression.

iv> function Expression:- define a function inside an expression.

v> Class expression:- The class name is used internally, but not outside of the class.

⇒ JAVASCRIPT STATEMENTS :-

* The Programming instructions written in a program in a programming language are known as statements.

i. Semicolons :-

* Semicolons separate javascript statements
* A semicolon marks the end of a statement in javascript.

ii. Code Blocks :-

* Javascript statements can be grouped together inside curly brackets, such groups are known as code blocks.
* The purpose of grouping is to define statements to be executed together.

iii. White space :-

* Javascript ignores multiple white spaces.

iv. Line length and line Breaks :-

* Javascript code's preferred line length by most programmers is upto 80 characters.
* The Best place to break a code line in javascript, if it doesn't fit, is after an operator.

V. keywords :-

* Keywords are reserved words and cannot be used as a variable name.

* A Javascript Keyword tells about what kind of operation it will perform.

* Some Commonly used keywords are :-

→ break and Continue :- break keyword is used to terminate a loop and continue is used to skip a particular iteration in a loop and move to the next iteration.

→ do.... while :- In this, the statements written written the do block are executed till the condition in while is true.

→ for :- It helps in executing a block of statements till the condition is true.

→ switch :- This helps in executing a block of codes depending on different cases.

⇒ JAVASCRIPT FEATURES :-

* Javascript is one of the most popular programming languages which includes numerous features when it comes to web development.

1. Lightweight Scripting Language :-

* Javascript is a lightweight scripting language because it is made for data handling in the browser (or) the client side.

* Because javascript is meant for client-side execution for web applications, hence the lightweight nature of javascript is a great feature.

2. Dynamic Typing :-

* Javascript supports dynamic typing which means types of the variable are defined based on the stored value.

* If you declare a variable x then you can store either a string (or) a number type value in javascript (or) an array (or) an object, is known as dynamic typing.

3. Object-oriented programming support :-

* Starting from ES6, the concept of class and oop is better deffined.

* Also, in javascript, two important principles with oop in javascript are :

i. Object creation patterns.
ii. Code Reuse patterns.

4. Prototype-based language :-

* Javascript is a prototype-based scripting language.

* This means javascript uses prototypes instead of classes (or) inheritance.

⇒ JAVASCRIPT EVENTS :-

* Javascripts has events to provide a dynamic interface to a webpage.

* These events are hooked to elements in the Document object Mode (DOM).

* There are some javascript events:-

1. Javascript onclick events: This is a mouse event and provokes any logic defined if the user clicks on the elements it is bound to.

2. Javascript onkeyup event: This event is a keyboard event and executes instructions whenever a key is released after pressing.

3. onmouseover event: This event corresponds to hovering the mouse pointer over the element and its children, to which it is bound to.

4. Javascript onmouseout event: whenever the mouse cursor leaves the element which handles a mouseout event, a function associated with it is executed.

5. Javascript onchange event: This event detects the change in value of any elements listing to this event.

6. Javascript onload event: When an element is loaded completely, this event is evoked.

7. Javascript onfocus event: An element listing to this event executes instructions whenever it receives focus.

8. Javascript onblur event: This event is evoked when an element loses focus.

⇒ WINDOWS :-

* It represents the browser window in which you are seeing the Content.

* The properties related to it are stored in the window object.

* It is loaded before the document because window container document.

* The Window is the global elements for all object, functions, etc...

Syntax: window object:

```
window. propertyname;
```

Example :

window. document. titie will return the title of the document.

⇒ DOCUMENTS :

* It represents the document loaded inside the window (or) browser.

* The properties related to it are stored in the document object.

* It is loaded after the loading window because window contains a document.

* It is the root element of the document object model.

syntax : document object :

document. propertyname.

Example : document.title will return the title of the document.

⇒ HTML /<frame> Tag :-

* HTML Frames are used to divide the web browser window into multiple sections where each section can be loaded separately.

* A Frameset tag is the Collection of frames in the browser window.

→ * Creating Frames :-

* Instead of using body tag, use frameset tag in HTML to use frames in web browser.

* But this Tag is deprecated in HTML 5.

* The Framest tag is used to define how to divide the browser.

* Each frame is indicated by frame tag and it basically defines which HTML documents shall open into the frame.

→ * Attributes Of frameset tag :-

→ cols :

* The cols attribute is used to create vertical frames in web browser.

* This attribute is basically used to define the no.of columns and its size inside the frame set tag.

Example :-

<frameset cols = " 300 , 400, 300 ">

→ rows :-

* The rows attribute is used to create horizontal frames in web browser.

* This attribute is used to define no.of rows and its size inside the frameset tag.

Example:-

<frameset rows = " 300 , 400, 300 ">.

→ border :-

* This attribute of frameset tag defines the width of border of each frames in pixels.

* zero value is used for no border.

Example :-

<frameset border = " 4 " frameset >.

→ frameborder :-

* This attribute of frameset tag is used to specify whether the three- dimensional border should be displayed between the frames (or) not for this use two values 0 and 1, where 0 defines for no border and value 1 signifies for yes there will be border.

⇒ DATA TYPES :-

① Primitive Data Types :-

* The predefined data types provided by javascript language are known as primitive data types.

> Number :- Javascript numbers are always stored in double-precision 64-bit binary format. IEEE 754.

```
Ex:- let num = 2;
     let num2 = 1.3;
     let num3 = Infinity;
     let num4 = 'something here too'/2;
```

> String : Javascript strings are similar to sentences, they are made up of a list of characters, which is essentially just an "array of characters, like" Hello GeeksforGeeks etc...

```
Ex:- let str = "Hello There";
     let str2 = 'single quotes works fine';
     let phrase = 'can embed ${str}';
```

> Boolean : Represent a logical entity and can have two values : true (or) false.

```
Ex! let is coding = true;
    let isold = false;
```

> Null : This type has only one value that is null.

```
Ex:- let age = null;
```

> undefined : A variable that has not been assigned a value is undefined

```
Ex:  let x;
     console.log(x);
```

⑤. Non-primitive Data Types :

* The Data types that are derived from primitive data types that are of the javascript language are known as non-primitive data types.

> object : It is the most important data type and forms the building blocks for modern javascript.

```
Ex:- let person = new object();
     let person = { };
```

⇒ BUILT-IN- FUNCTIONS :-

1. eval :-

* eval evaluates a string and returns a value.

* Expression is evaluated, and if the result is not a string, the result is returned, if the result is a string, it is taken to be a javascript program, and it is evaluated.

```
Ex :  Var x = 2
      Var y = 39
      Var z = "42"
      println (eval ("x+y+1"))
      println (eval (z))
```

2. parseInt :

* Parses a string argument and returns an integer of the specified radix (or) base

```
Syntax:  ParseInt (string)
         ParseInt (string, radix)
```

Ex:-
```
parse Int ("F",16)
parse Int ("17", 8)
parse Int ("15", 10)
parse Int ("15.99, 10)
parseInt ("FXX123", 16)
parseInt ("1111", 2)
parseInt ("15* 3", 10)
```

3. ParseFloat :-

* Parses a string argument and returns a floating point number.

Syntax: parse Float (string)

Ex:-
```
parseFloat ("3.14")
parse Float ("314 e-2")
Parse Float ("0.0314 E+2")
var x = "3.14"
Parse Float (x).
```

4. escape:

* Returns the hexadecimal encoding of an argument in the Iso Latin-1 character set.

Syntax! escape ("string")

ex: escape ("abc&%")

5. Unescape:

* Returns the ASCII string for the specified value.

Syntax: Unescape ("string")

Ex: Unescape ("%26").

⇒ BROWSER OBJECT MODEL (BOM) :-

* The Browser Object Model (BOM) is a browser specific convention referring to all the objects exposed by the web browser.

* The BOM allows javascript to "interact with" the browser.

Properties:

> Screen.width : The Screen.width property returns the user's screen width in pixels.

Ex:
```
<p id = "GFG"></p>
<Script>
document.get Element By Id("GFG").innerHTML =
                    " The Screen width is" + Screen.width;

</script>
```

> Screen.height : The Screen.height property returns the users screen height in pixels.

Ex:
```
<p id = "GFG"></p>
<Script>
document.getElement ById("GFG").innerHTML=
" The Screen height is"+screen.height;
</Script>
```

> screen.availwidth : The Screen.availwidth property returns the users screen width in pixels, excluding the interface features.

Ex:
```
<p id = "GFG"></p>
<Script>
document.get ElementById("GFG").innerHTML=
“The availability screen width is” + screen.availwidth;
</script>.
```

> screen.availHeight : The screen.availHeight property returns the users screen height in pixels excluding the interface features.

Ex :

```
<p id = “GFG”></p>
<script>
document.getElementById (“GFG”).innerHTML =
“The availability screen height is” + screen.availHeight ;
</script>.
```

> Screen.ColorDepth : The screen.ColorDepth property returns the bits to be used to display one color.

Ex :

```
<p id = “GFG”></p>
<script>
document.getElementById (“GFG”).innerHTML =
“The screen color depth is” + screen.ColorDepth ;
</script>.
```

> screen.pixelDepth : The screen.pixelDepth property returns the pixel depth of the screen.

Ex :

```
<p id = “GFG”></P>
<script>
document.getElementById (“GFG”).innerHTML =
“The screen pixel depth is” + screen.pixel Depth ;
</script>.
```

⇒ VERIFYING FORMS:

* Forms are used on webpages for the user to enter their required details that further send to the Server for processing.

* A form is also known as a webform (or) html form.

prerequistes:

> HTML: HTML is used to create the form.

> CSS: CSS to design the layout of the form.

> Javascript: Javascript to validate the form.

Validating a form: The data entered into a form needs to be in the right format and certain fields needs to be filled in order to effectively use the submitted form.

Validations used in below code snippet:

```
// Javascript reGex for ~~Am~~ Email Validation
Var reg Email = /^\w+([\.-]?\w+)*@\w+([\..]?\w+)*
                (\.\w{2,3})+$/g;
Var reg phone = /^\d{10}$/;
Var reg Name = /\d+$/g;
```

⇒ HTML 5:

* HTML Stands for Hyper Text Markup Language.

* It is used to design web pages using a markup language.

Features:

* It has introduced new multimedia features which supports both audio and video controls by using

<audio> and <video> tags.

* There are new graphics elements including vector. graphics and tags.

Removed Elements from HTML 5 :-

1. <acronym>
2. <applet>
3. <basefont>
4. <big>
5. <centre>
6. <dir>
7. <font>
8. <frame>
9. <frameset>
10. <object>

New added Elements in HTML 5 :

1. <article> : The <article> tag is used to represent an article, the Content within the <article> tag is independent from the other Content of the site.

2. <aside> : The <aside> tag is used to describe the main object of the web page in a shorter way like a highlighter.

3. <figcaption> : The <figcaption> tag in HTML is used to set a caption to the figure element in a document.

4. <figure> : The <figure> tag in HTML is used to add Self-Contained Content like illustrations, diagrams, Photos (or) Codes listing in a document.

5. <header> : It Contains the section heading as well as other content, such as a navigation links, table of Contents, etc....

⇒ CSS 3 :-

* CSS 3 stands for Cascading Style Sheet level 3 which is the advanced version of css.

* It is used for structuring, styling and formatting web pages.

* Several new features have been added to css 3 and it is supported by all modern web browser.

* The Most important feature Of css 3 is the splitting Of css standards into separate modules that are simpler to learn and use.

New features Of css 3 :

Combinator : css 3 has a new General sibling combinator which matches up with sibling elements via the tilde (~) combinator.

css Selectors : Css 3 selectors are much advanced in comparision to simple selectors offered by css, and are termed as a sequence of easy to use and simple selectors.

Pseudo - elements : plenty of new pseudo-elements have been added to css 3 to give easy styling in depth.

Border style : The Latest css 3 also has new border styling features like border-radius, image-slice, image-source, and values for "width stretch" etc...

Background style properties: New features like background - Clip, size, style, and origin properties have been added to CSS3.

⇒ HTML CANVAS :-

* The HTML "canvas" element is used to draw graphics via javascript.

* The "canvas" element is only a container for graphics.

* One must use javascript to actually draw the graphics.

* Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

syntax:

```
    <canvas>
        Content....
     </canvas>
```

HTML Canvas Methods:

SetInterval (callback, time): This method repeatedly executes the supplied code after a given time.

SetTimeOut (callback, time): This method executes the supplied code only once after a given time.

⇒ WEB SITE CREATION USING TOOLS :-

* Web development refers to the creating, building, and maintaining of websites.

* It includes aspects such as web design, web publishing, web programming, and data-base management.

The word web Development is madeup Of tw
words, that is:

web: It refers to websites, web pages (or) anything that works over the internet.

Development: It refers to building the application from scratch.

web Development can be Classified into two ways:

1. Frotend Development:

* The Part of a website where the user in-teracts directly is termed as frontend.

* It is also referred to as the 'client side' of the application.

Popular Frontend Technologies:

HTML: HTML stands for Hyper Text Markup Language, it is used to de-sign the front end portion of web pages using markup language.

css: Cascading Style Sheets fondly refferred to as css is a simply designed language intended to simplify the process of making web pages presentable.

Javascript: Javascript is a scripting language used to provide a dynamic behaviour to our website

Bootstrap: Bootstrap is a free and open-source tool collection for creating responsive websites and web applications.

2. Backend Development:

* Backend is the server side of a website.

* It is a part of the website that users cannot see and interact with.

popular Backend Technologies:

PHP: PHP is a server-side scripting language designed specifically for web development.

Java: Java is one of the most popular and widely used programming languages.

python: Python is a programming language that lets you work quickly and integrate systems more efficiently.

Node.js: Node.js is an open source and cross-platform runtime environment for executiong javascript code outside a browser.

UNIT-II

JAVA

- → Introduction to Object-Oriented programming.
- → Features of Java.
- → Data types.
- → Variables and arrays.
- → Operators.
- → Control statements
- → Classes and Methods
- → Inheritance
- → packages and interfaces.
- → Exception Handling
- → Multithreaded programming
- → Input / output
- → Files
- → Utility Classes
- → String Handling.

⇒INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING

1. OOPS [object Oriented Programming] is a core of java programming, which is used for designing a program using classes and objects.

OOPS Concepts:

1. class
2. Objects
3. Data Abstraction
4. Encapsulation
5. Inheritance
6. polymorphism.

1. Class: A class is a user-defined data type, it consists of data members and member functions, which can be accessed and used by creating an instance of that class.

2. objects: An object is an instance of a class, when a class is defined, no memory is allocated but when it is instantiated (i.e., an object is created) memory is allocated.

3. Data Abstraction: Data Abstraction refers to providing only essential information about the background details (or) implementation.

4. Encapsulation: Encapsulation is defined as the wrapping of data and methods into a single unit is known as encapsulation and it is also known as data-hiding.

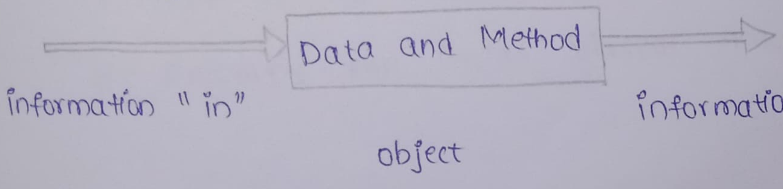
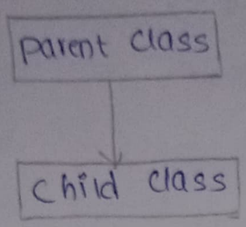
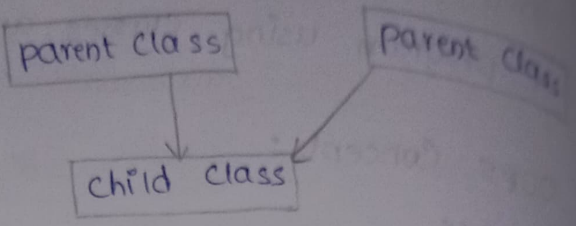


Fig: Encapsulation

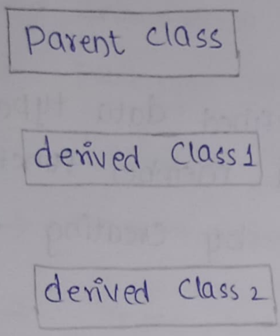
5. Inheritance: The Capability of a class to d properties and characteristics from another class i called inheritance.



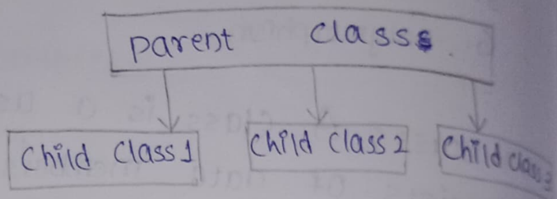
Single inheritance



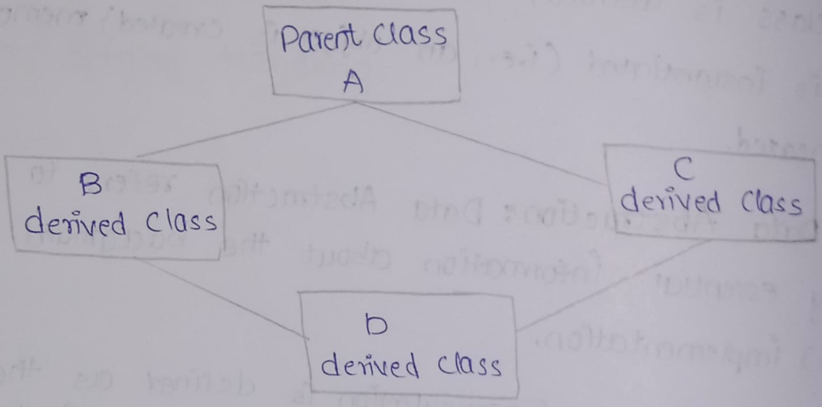
Multiple inheritance.



Multilevel inheritance



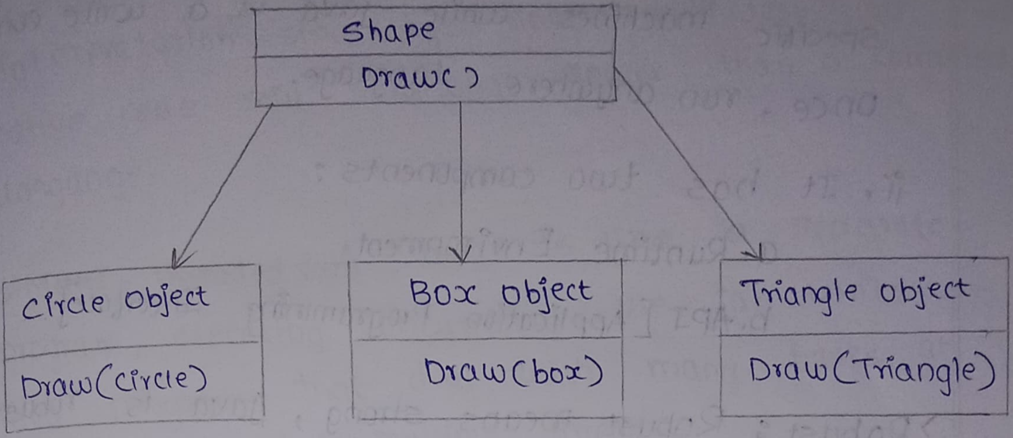
Hierarchical inheritance



Hybrid Class

6. Polymorphism: Polymorphism is defined as the ability of a message to be displayed in more than one form.

example:



⇒ FEATURES OF JAVA :-

* The Primary objective of Java Programming language creation was to make it portable, simple and secure programming language.

* There are mainly 12 types of features of Java, They are:

> simple: Java language is simple because:

i. syntax is based on C++

ii. Removed many confusing and/or rarely-used features.

> Object-Oriented: OOPS (Object Oriented Programming) is a core of java programming, which is used for designing a program using classes and objects.

> secured: Java is secured because:

i. There is no explicit pointers

ii. Programs run inside virtual machine sandbox.

> Platform Independent : Java is Platform independent because:

i. It is different from other languages like C++ etc..., which are Compiled into platform Specific machines while java is a write once, run anywhere language.

ii. It has two components :

a. Runtime Environment

b. API [Application Programming Interface].

> Robust : Robust means strong, Java is robust because:

a. It uses strong memory management.

b. There is a lack of pointers that avoids security problems.

> Portable : We may carry the java bytecode to any platform.

> Architecture - neutral : There are no implementation dependent features.

Ex : size of primitive types is set.

> Dynamic : Java is a dynamic loading of classes. language because:

i. It supports the dynamic loading of classes.

ii. It also supports functions from its native languages, i.e., C and C++.

> Interpreted : Java byte Code is translated on the fly to native machine instructions and is not stored anywhere.

> High Performance : Java is faster than traditional interpretation. since byte code is "close" to native code still somewhat slower than a compiled language.

> Multi-threaded : A Thread is like a separate program, executing Concurrently, we can write java programs that deal with many tasks at once by defining multiple threads.

> Distributed : Java is distributed because:

i. It facilitates users to create distributed applications in java.

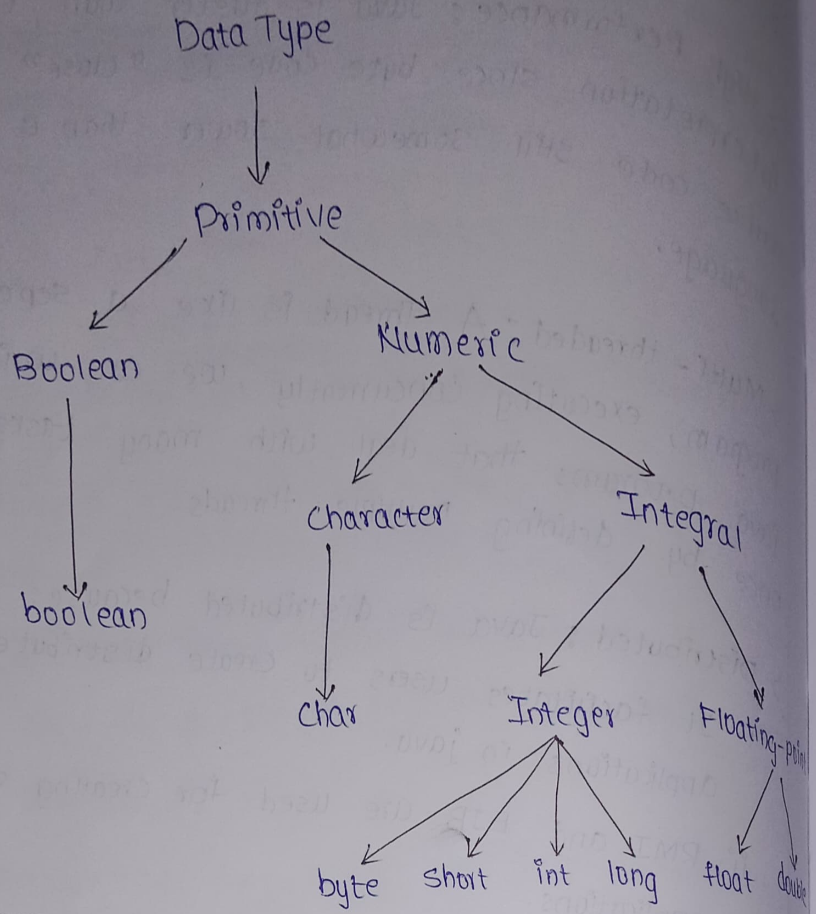
ii. RMI and EJB are used for Creating distributed applications.

⇒ DATA TYPES :

* Data types a in java are of different sizes and values that can be stored in the variable that is made as per convenience and circumstances to cover up all be test cases.

* There are mainly two types of Data types. They are:

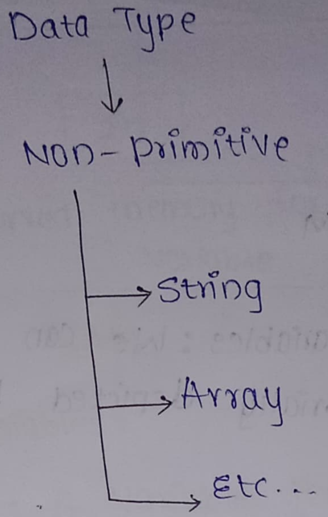
①. Primitive Data Types : Data Types make a Variable to store a single value at a time



Type	Size (byte)	Range	Default
byte	1	-128 to 127	0
Short	2	-32768 to 32767	0
Int	4	-2147483648 to 2147483647	0
Long	8	-9223372036854775808 to 9223372036854775807	0
Float	4	3.4e38 to 1.4e-45	0.0
Double	8	1.e-308 to 4.9e-324	0.0
Boolean	JVM Specific	true (or) false	FALSE
Char	2	0.65535	\u0000

② . Non-Primitive Data Types :

Derived data types are those that are made by using any other data type and can make a variable to store multiple values.



> String : Strings are defined as an array of characters, the string is designed to hold a sequence of characters in a single variable.

Syntax :

<string-Type><string-variable> = "<sequence-of-string>" ;

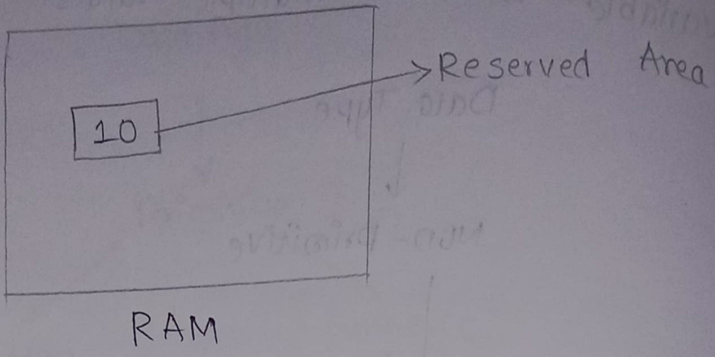
> Array :

* An Array is a group of like-typed variables that are reffered to by a common name.

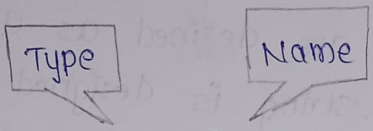
* Arrays in java work differently than they do in c/c++.

⇒ VARIABLES:

Variable is name of reserved area allocated in memory.



Declaration Of Variables: We can declare Variables in java as pictorially depicted below as a visual aid



Int Count;

Data type: Type Of data that can be stored in this variable.

Data_name: Name was given to the variable.

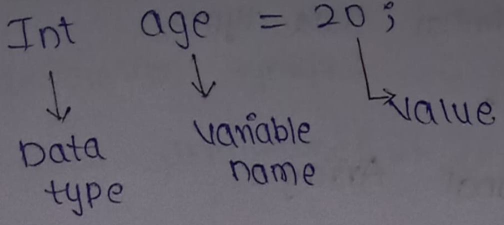
Syntax: Datatype Variable_name;

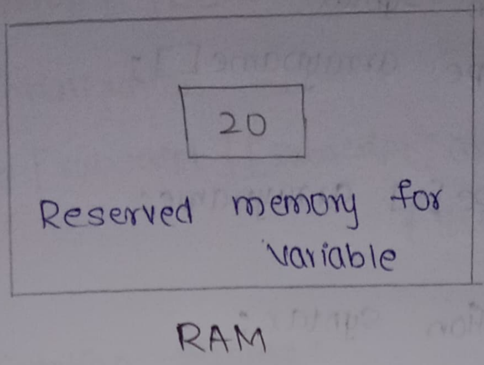
Initialization Of Variables: It can be perceived with the help of 3 Components that areas follows:

> Datatype: type of data that can be stored in this variable.

> Variable_name: Name given to the Variable.

> Value: It is the initial value stored in the Variable.





Types of Variables:

Local Variables: A Variable that is declared inside the method is called Local Variable.

Instance Variable: A Variable that is declared inside the class but outside the method is called Instance Variable.

Static Variable: A Variable that is declared as Static is called Static Variable.

Example:

```
class A
{
     int data = 50; // instance Variable
     static int m = 100; // static Variable
     void method ()
     {
          int n = 90; // local variable
     }
} // end of class.
```

⇒ ARRAYS: Java array is an object the Contain elements of similar data type.

Types of Arrays:

1) Single dimensional Array:

Array Declaration Syntax!

```
Datatype arrayname[];
or
Datatype[] arrayname;
```

Array initialization syntax:

```
arrayname[subscript] = value;
```

Array creation syntax:

```
arrayname = new datatype[size];
```

Rules for creating an Array:-

Rule 1: At the time of array creation compulsory we should specify the size otherwise we will get compile time error.

Rule 2: It is legal to have an array with size zero in java.

Rule 3: If we are taking array size with negative int value then we will not get runtime exception saying Negative Array & size Exception.

Rule 4: The Allowed data type to specify array size are byte, short, char, int.

Rule 5: The Maximum allowed array size in java is maximum value of int size [2147483647]

2> Multi dimensional Array :-

Array declaration syntax :

Datatype arrayname[ ][ ];

or

Datatype [ ][ ] arrayname;

Array intialization syntax :

datatype [subscript] [subscript] arrayname = {list of value};

Array Creation syntax :

arrayname = new datatype [size] [size];

Ex : int a[ ][ ];
a = new int[3] [2];

=> OPERATORS :

* An Operator is a symbol which perform some operation.

* There are mainly '8' types of operators. They are:

i. Arithmetic operators.

ii. Recational operators

iii. Logical operators

iv. Assignment Operators.

v. Increment and decrement operators.

vi. Conditional operators.

vii. Bitwise operators

viii. Special operators.

i> Arithmetic Operators :

* Arithmetic operators are used to perform basic Arithmetic operations.

* There are mainly 5 types of Arithmetic Operators

They are :

- + : Addition
- − : Subtraction
- * : Multiplication
- / : Integer Division
- % : Modulo Division.

Example : a = 5 , b = 2

a+b = 5+2 = 7

a−b = 5−2 = 3

a*b = 5*2 = 10

a/b = 5/2 = 2

a%b = 5%2 = 1

ii> Relational Operators :

* Relational Operators are used to compare two quantities.

* They always give a result of true (or) false.

* There are maily 'six' types of relational operators. They are :

< : less than

<= : lessthan (or) equal to

> : greater than

>= : greater than (or) equal to

== : Equal to

!= : Not equal to

Example: a = 5 , b = 2

a < b => 5 < 2 = False

a > b => 5 > 2 = True

a = = b => 5 = = 2 = False

a <= b => 5 <= 2 = False

a >= b => 5 >= 2 = True

a != b => 5 != 2 = True.

iii> Logical Operators :-

* Logical operators are used to Combine two (or) more conditions.

* They also always give a result of True | False.

* There are mainly 3 types of logical operators. They are :-

&& : Logical AND.

|| : Logical OR

! : Logical NOT

Example: a = 5 , b = 3 , c = 1

(a < b) && (b < c) = False

(a > b) || (b < c) = True

(a < b) ! (b < c) = False.

iv> Assignment Operators :-

* Assignment Operators are used to assign the value obtained from right side expression to left side variable.

* There are mainly two types of assignment operators. They are :

= : Equal to

V op = e : Shorthand Assignment Operator.

V : Variable

op : Operator

e : expression.

Example:

c=a+b

1. a=a+5 can be written as a+=5

2. x=x/y can be written as x/=y

v. Increment and Decrement Operators:

* There are 2 special operators in Web Programming language, Increment (++) operator and Decrement (--) operator.

* The Increment operator increases the value of an operand by 1.

* The Decrement operator decreases the value of an operand by 1.

* They can be used as 'prefix' and 'postfix'.

Example:

x=5;	x=5;	x=5;	x=5;
++x	x++	--x	x--

After execution:

x=6 x=6 x=4 x=4.

vi. Conditional operators: There are mainly 3 types of Conditional operators. They are:

> Uniary Operators: The operators which uses only one operands to perform the operation are known as Uniary Operators.

Ex: (++), (--)

> Binary Operators : The Operators which uses two operands to perform the operation are known as Binary Operators.

Ex:- '+', '-', '*', '/', '%'.

> Ternary Operators : The Operators which uses 3 operands to perform the Operation are known as Ternary Operators.

Ex:- ? :

> Bitwise Operators ?

vii. Bitwise Operators:

* The Bitwise operators are used to perform operations on bits [Binary digits] i.e 0 (or) 1

* There are mainly six types of Bitwise operators. They are:

& : Bitwise AND

! : Bitwise OR

^ : Bitwise XOR

<< : Leftshift

>> : Rightshift

~ : Complement.

viii. Special operators : There are mainly 3 types of Special operators. They are !

i. size of operators

ii. Membership operators (. →)

iii. Pointer operators (*, &).

=> CONTROL STATEMENTS: There are mainly 3 types of Control statements. They are:

1. Decision Making in Java: There are mainly 3 types of Decision making statements. They are

i. If statement:

* "If" statement is used to evaluate a Condi

* There are mainly four types of if statement. They are:

> simple if statement: It evaluates a Boolean expression and enables the program to enter a block of code if the expression evaluates to true.

Syntax: if (condition)

```
        {
            statement 1;
        }
```

> if-else statement: The if-else statement is an extension to the if-statement, which uses another block of code, i.e., else block.

Syntax:

```
        if (condition)
        {
            statement 1;
        }
        else
         {
             statement 2;
         }
```

> Nested if statement : The Nested-if statements the if statement can contain a if (or) if-else statement inside another if (or) else-if statement.

syntax :

```
if (Condition 1)
{
    Statement 1 ;
    if (Condition 2)
    {
        Statement 2 ;
    }
    else
    {
        Statement 3;
    }
}
```

> if - else - if : ~~If~~ The if - else - if statement contains the if - statement followed by multiple else - if statements.

Syntax :

```
if (Condition 1)
{
    Statement 1 ;
}
else if (Condition 2)
{
    Statement 2 ;
}
else
{
    Statement 2 ;
}
```

1. Switch Statement :

* Case value must be of the same type as expression used in switch statement.

* Case value must be a constant (or) literal.

* Case values should be unique.

* If it is duplicate, then program will give compile time error.

2. Looping Statements in Java : There are mainly 4 types of looping statements. They are :-

> while : It checks and evaluates the condition and if it is true then executes the body of loop.

```
Syntax: Initialization ;
        while (condition)
        {
            Statement 1 ;
            increment / decrement ;
        }
```

> Do...while : It has only one difference that in do.... while, condition is checked after the execution of the loop body.

```
Syntax :
        do
        {
            Statement 1 ;
        } while(condition);
```

> For statement : It is the easiest way to construct a loop structure in code as initialization of a variable, a condition and increment / decrement are declared only in a single line of code.

```
syntax : for (initialization; condition; increment/decrement)
         {
              statement ;
         }
```

> For- Each loop statement: for-each-loop is used to traverse through elements in an array.

```
syntax: for ( data-type var : array-name/collection-name)
        {
             // statements

        }
```

3. Branching statements in Java : There are mainly two types of branching statements. They are:

> Break : Break statement is used to terminate the execution and bypass the remaining code in the loop.

> Continue: Continue statement works same as break but the difference is it only comes out of loop for that iteration and continue to execute the code for next iterations.

⇒ CLASSES :- The Java Class defines the blue print of an object, every Class of Java programming language has the following characteristics

Identify : It is the name given to the Class.

State : Represents data values that are associated with an object.

Behaviour : Represents actions can be performed by an object.

Creating a class : In Java, we use the keyword Class to Create a Class, A Class in java contains properties as variables and behaviours as methods.

Syntax :
```
Class <ClassName>
  {
      data members declaration;
      methods definition;
  }
```

Creating an object : When an object of a Class is Created, the Class is said to be instantiated, all the objects are Created using a single Class have the same propertities and methods.

Syntax :
```
<ClassName> <objectName> = new <ClassName>();
```

⇒ Methods : A Method is a block of statements under a name that gets executes only when it is called , Every method declaration has the following characteristics.

return Type : specifies the data type of a return value.

name : Specifies a unique name to identify it.

parameters : The data values it may accept (or) receive.

{ } :- Defines the block belongs to the method.

Creating a method :- A Method is created inside the class and it may be created with any access specifier.

Syntax :

```
    Class <Class Name>
    {
       <access Specifier><return Type><methodName>
       {
          ....
          block Of statements;
          ....
       }
    }
```

Calling a method : In java , a method Call precedes with the object name of the class to which it belongs and a dot operator.

Syntax :

```
<objectName>< MethodName> (actual Arguments);
```

Variable arguments of method: In java, a method can be defined with a variable number of arguments, that means creating a method that receives any number of arguments of the same data type.

Syntax:

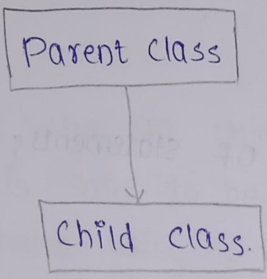
```
<returnType><method Name>(data Type.........
                        Parameter Name);
```

⇒ <u>INHERITANCE</u> :- The Capability of a class to derive properties and characteristics from am another class is called inheritance.

Types of inheritance:

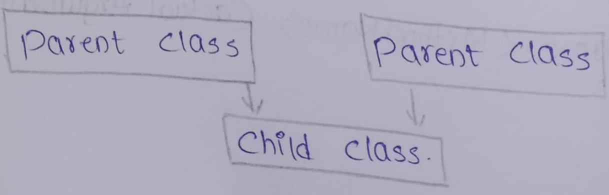
1. Single inheritance: Single inheritance refers to a Child and parent class relationship where a Class extends the another class.

Example:



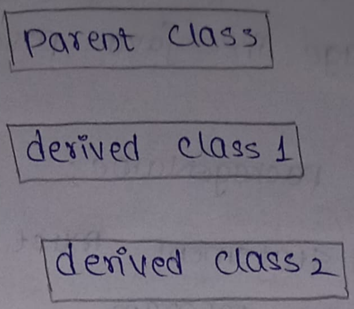
2. Multiple Inheritance: Multiple inheritance refers to the concept of one class extending more than one classes, which means a child clas has two Parent Classes.

Example:



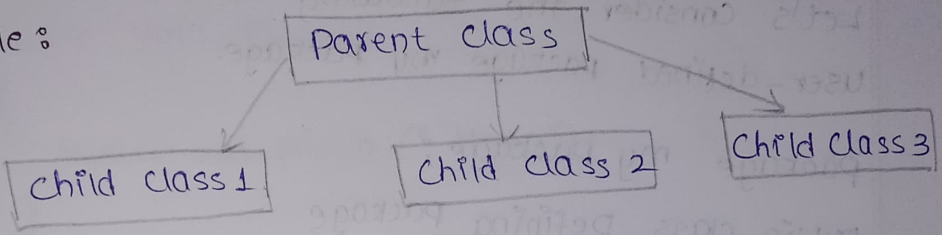
3. Multilevel inheritance: Multilevel inheritance refers to a child and parent class relationship where a class extends the child class.

Example:



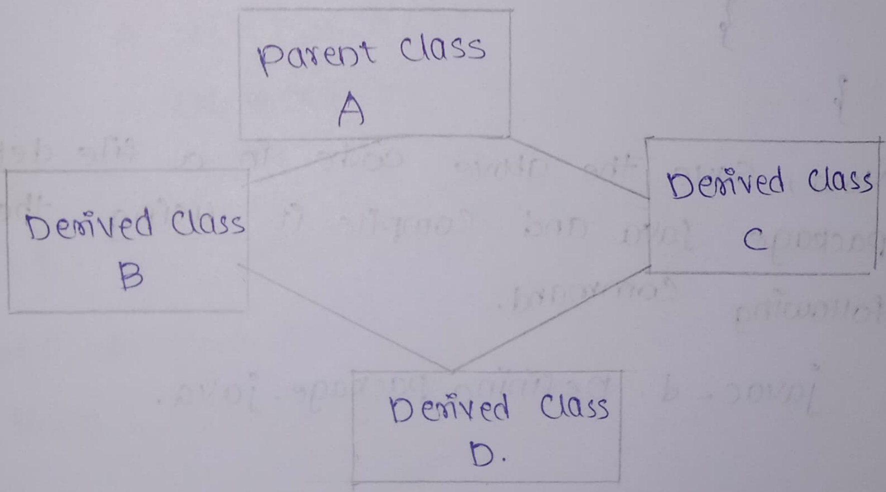
4. Hirerchical inheritance: Hierarchical inheritance refers to a child and parent class relationship where more than one classes extends the same class.

Example:



5. Hybrid inheritance: Combination of more than one type of inheritance in a single program.

Example:



⇒ PACKAGES :-

Defining a package:

* We use the package keyword to create (or) define a package in java programming langua

Syntax:

```
package packageName;
```

* The package statement must be the first statement in the program.

* The package name must be a 'single word'.

* The package name must use 'camel case Notation'

Let's consider the following code to create a user-defined package my package.

```
Package my package;
Public class Defining Package
{
    public static void main(string[] args)
    {
        System.out.println("This class belongs to
        my package");
    }
}
```

Now, save the above code in a file defining package java and compile it using the following command.

```
javac -d. Defining Package.java.
```

Run the program use the following command.

java my package. Defining package.

Importing packages: There are mainly 3 types of importing packages. They are:

> Using packagename*:-

If you use package* , then all the classes and interfaces of this package will be accessible but not subpackages.

Ex:-

```
package pack;
public class A
{
   public void msgs()
   {
      System.out.println ("Hello");
   }
}.
import pack*;
class B
{
   public static void main(string args[])
   {
      A obj = new A();
      Obj . msg();
   }
}
```

output:

```
Hello.
```

> Using packagename. classname :-

If you import package. classname then, only declared class of this package will be accessible.

Ex :-
```
Package pack;
Public class A
{
   Public void msgs()
   {
      System.out.println ("Hello");
   }
}
import pack.A;
Class B
{
   Public static void main (String args[])
   {
      A obj = new A();
      Obj.msg();
   }
}
```

Output :
```
Hello.
```

> Using fully qualified name:

If you use fully qualified name then, only declared class of this package will be accessible.

* Now there is no need to import.

Example :-

```
package pack;
public class A
{
  public void msgs()
  {
     System.out.println (" Hello");
  }
}
class B
{
  public static void main (string args[])
  {
     pack.A Obj = new pack.A();
     Obj.msg();
  }
}
```

Output :

```
Hello.
```

⇒ INTERFACES!

Defining an interface :-

* An interface in java is a "blueprint of a class"

* It is used to achieve and "multiple inheritance"

* It is used to achieve "abstraction"

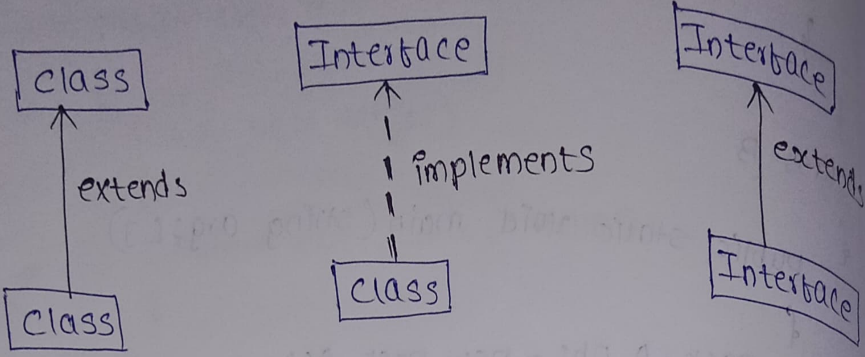
* It is used to achieve "loose coupling"

Types of interfaces:

There are mainly 5 types of interfaces. They are:

> Implementing interfaces:

* As shown in the figure given below, a cla extends another class, an interface extends another interface, but a class implements an interface.



> Nested interfaces :-

* An Interface which is declared within another interface (or) Class is known as nested interface.

* The Nested interfaces are used to group related interfaces so that they can be easy to maintain.

> Applying interfaces:-

* Interface implements multiple inheritance into the same class.

* A class can extend only class but it can implements multiple interfaces.

> variables in interfaces :-

* An Interface can also contain variables just as classes but the variables in interface must be of public, static and final access modifiers.

* If the variable are not specified with access specifiers then during compilation process compiler will provide this access specifier.

> Extending Intefaces:-

* An Interface can extend another interface.

* An Interface cannot extend multiple interfaces.

* An Interface can implement neither an interface nor a class.

* The class that implements child interface needs to provide code for all the methods defined in both child and parent intefaces.

Example program:-

```
interface MyInterface
{
    public void method();
}
Class Demo implements My Interface
{
    public void method()
    {
     System.out.println("Implementation of method");
    }
    public static void main(String arg[])
    {
        Demo obj =new Demo();
        Obj.method();
    }
}
```

Output:

```
Implementation of method.
```

=> EXCEPTION HANDLING :-

Definition:- The Exception Handling is a mechan to handle the runtime errors so that the normal flow of the application can be maintain

* Fundamentals of Exception handling :
Java exception handling is managed via throu 5 keywords. They are :

1. Try : The try block contains a set of statements where an exception can occur.

2. Catch : The catch block is used to handle the uncertain condition of a try block.

3. Throw : The throw keyword is used to transfer control from the try block to the catch block.

4. Throws : The throws keyword is used for exception handling without try and catch block.

5. Finally : It is executed after the catch block.

Types of exception handling :-
1. Checked Exception :
* An Exception that is checked by the compiler at the time of compilation is called a checked exception.

* The following are a few built-in-classes used to handle checked exceptions in java.

i. IO Exception

ii. FileNotFound Exception

iii. ClassNotFound Exception

iv. SQL Exception

v. Data Access Exception

vi. Instantiation Exception

viii. UnknownHost Exception.

2. Unchecked Exception :-

* An Exception that cannot be caught by the compiler but occurs at the time of program execution is called an Unchecked Exception.

* The following are few built-in- classes used to handle uncheeked exceptions in java.

i. Arithmetic Exception

ii. Nullpointer Exception

iii. Number Format Exception

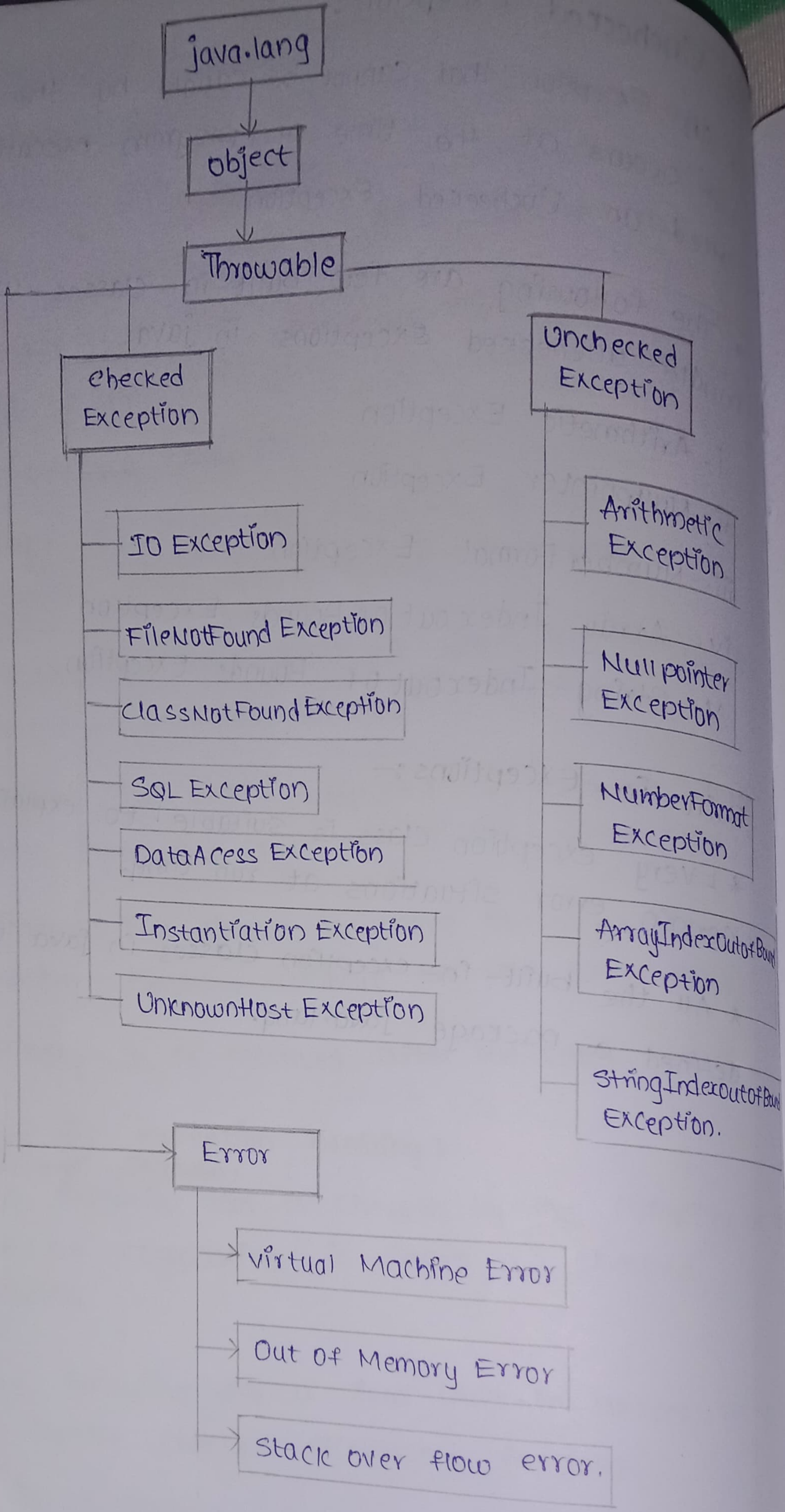
iv. Array Index out of Bounds Exception

v. String Index out of Bounds Exception.

Built-in- Exceptions :-

* Every exception class is suitable to explain certain error situations at run time.

* All the built-in- exception classes in java were defined a package java.lang.


java.lang
object
Throwable
checked Exception
Unchecked Exception
IO Exception
FileNotFound Exception
ClassNotFound Exception
SQL Exception
DataAcess Exception
Instantiation Exception
UnknownHost Exception
Arithmetic Exception
Nullpointer Exception
NumberFormat Exception
ArrayIndexOutofBound Exception
StringIndexoutofBound Exception.
Error
virtual Machine Error
Out of Memory Error
Stack over flow error.

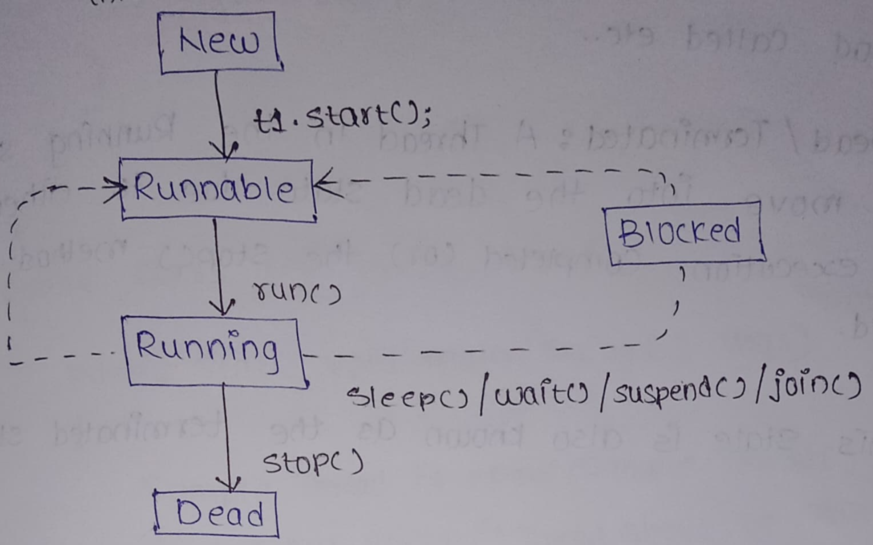
⇒ MULTITHREADED PROGRAMMING:

* A Thread is a light weight process

* A Thread is a subpart of a process that can run individually.

* Thread Life cycle: The Life cycle of a thread is shown in the following figure.

```
Thread t1 = new Thread();
```



1. New/born: When a thread object is created using new, then the thread is said to be in the New state.

* This state is also known as born state.

```
Thread t1 = new Thread();
```

2. Runnable / Ready: When a thread calls start() method, then the thread is said to be in the Runnable state.

* This state is also called as a Ready state

```
t1.start();
```

3. Running : When a thread calls run() method, then the thread is said to be Running.

* The run() method of a thread called automatically by the start() method.

4. Blocked / waiting : A Thread in the Running state may move into the blocked state due to various reasons like sleep() method called, and join() method called etc..

5. Dead / Terminated : A Thread in the Running state may move into the dead state due to either its execution Completed (or) the stop() method Called.

* This state is also known as the terminated state.

* Creating Threads :

There are mainly two ways to create a thread. They are :-

> By Extending thread class: To create a thread by using thread class, follow the steps given below.

Step-1: Create a class that extends thread class.

Step-2: Override the run() method with the code that is to be executed by the thread.

Step-3: Create the object of the newly created class in the main() method.

Step-4: Call the start() method on the object created in the above step.

example :

```
class  Sample Thread extends  Thread
{
  public  void  run()
  {
    System.out.println(" Thread  is under Running...");
    for(int  i=1 ; k=10 ; i++)
    {
        System.out.println("i=" +i);
    }
  }
}
public  class  My-Thread-Test{
  public static void main(string[] args)
  {
      Sample Thread t1 = new Sample Thread();
      System.out.println(" Thread about to start....");
      t1.start();
  }
}
```

Output :

```
 Thread  about  to  start....
 Thread  is  under  Running...
 i=1
 i=2
 i=3
 i=4
 i=5
 i=6
 i=7
 i=8
 i=9
 i=10
```

> By implementing Runnable Interface :- To create a thread using Runnable interface, follow the steps given below.

Step-1: Create a class that implements Runnable interface.

Step-2: Override the run() method with the code that is to be executed by the thread.

Step-3: Create the object of the newly Created class in the main() method.

Step-4: Call the start() method on the thread class object created in the above step.

* Example:

```
Class Sample Thread implements Runnable
{
    Public Void run()
    {
        System.out.println("Thread is under Running");
        for(int i=1; k=10; i++)
        {
            System.out.println("i=" + i);
        }
    }
}
Public Class My-Theard-Test2
{
    Public static void main(String[] args)
    {
        sample Thread thread objects = new SampleThread();
        Thread thread = new Thread(thread object);
        System.out.println("Thread about to start....");
        thread.start();
    }
}
```

Output :-

```
Thread about to start.....
Thread is under Running.....

i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10.
```

* Thread Priorities :-

* Priorities in threads is a concept where each thread is having a priority.

* The default priority is set to 5 as expected

i. Minimum priority is set to 1.

ii. Maximum priority is set to 10.

There are three constants are defined in it namely as follows:

1. public static int NORM_PRIORITY
2. public static int MIN_PRIORITY
3. public static int MAX_PRIORITY.

How to get and set priority of a Thread :-

1. Public final int get priority() :-

*. java. lang' Thread. get priority() method returns priority of given thread.

2. public final void set priority :-

* java. lang. Thread . set priority() method changes the priority of thread to the value new priority.

⇒ STREAM BASED INPUT / OUTPUT :- T

There are mainly two types of stream based Input / output. They are :

> Byte Streams: There are mainly two types of Byte streams. They are :

i. Input Stream:

* It is an abstract class that defines model of streaming byte input.

* It implements Closeable interface.

* It defines methods for performing input function such as

i. Reading Bytes
ii. closing streams
iii. Marking positions in Streams.
iv. skipping a head in Stream.
v. Finding the number of Bytes in a streams.

ii. Output Stream:

* It is an abstract class that defines model of streaming byte output.

* It implements Closeable and flushable interfaces.

* It defines methods for performing ouput functions such as,

→ ~~i~~. writing bytes.
→ Closing streams
→ Flushing streams.

> Character Streams: There are mainly two types of Character streams. They are :

i. Reader stream: It is an abstract class that defines model of streaming character input.

* It implements closeable and readable interfaces.

ii. Writer stream:

* It is an abstract class that defines model of streaming character output.

* It implements closeable, flushable, and appendable interface.

⇒ STREAM BASED FILES:

There are mainly two types of stream based files. They are :-

* Reading / Writing Bytes:

* The subclasses of Reader and writer implements streams that hadle bytes.

* The Two subclasses used are :

i. FileInputStream - for reading bytes.

ii. FileOutputStream - for writing bytes.

* Reading / writing Characters:

* The subclasses of Reader and writer implements streams that hadle characters.

* The two subclasses used are :

i. FileReader - for reading characters

ii. FileWriter - for writing characters.

⇒ UTILITY CLASSES :-

1. Array List: Class provides resizable-array and implements the List interface.

2. HashMap: Class is the Hash table based implementation of the Map interface.

3. Hashset: Class implements the Set interface, backed by a hash table.

4. Priority Queue: Class is an unbounded Priority queue based on a priority heap.

5. Random: class interface instance is used to generate a stream of pseudorandom numbers.

6. scanner: Class is a simple text scanner which can parse primitive type and strings using regular expressions.

7. String Tokenizer: Class allows an application to break a string into tokens.

8. Tree Map: Class is the Red-black tree based implementation of the Map interface.

⇒ STRING:

Definition: String basically represents, sequence of char values.

* Creating string object: There are mainly two ways to create string object. They are:-

> string Literal :

* In Java, string is an object that represents a sequence of characters.

* The java.lang.string class is used to create a string Object.

Ex:- String S1 = "Welcome".

> New keyword :

* The java string is immutable which means it cannot be changed.

* Whenever we change any string, a new instance is created.

Ex: String S_2 = new string ("Welcome");

* String Class methods :-

1. CharAt (int index): returns char value for the particular index.

II. length(): returns string length.

III. Substring(int beginIndex): returns substring for given begin index.

IV. join(): returns a joined string.

V. equals(): Checks the equality of string with the given object.

VI. Concat(): concatenates the specified string.

VII. toLower Case(): returns a string in lowercase

VIII. toUpper Case(): returns a string in Uppercase.

UNIT-III

JDBC

> JDBC overview

> JDBC implementation

> Connection Class

> Statements

> Catching Database Results

> Handling Database Queries

> Networking

> Inet Address class

> URL Class

> TCP sockets

> UDP Sockets

> Java Beans

> RMI.

⇒ JDBC OVERVIEW :-

Definition : JDBC stands for Java Database Connectivity, It is a Java application programming interface (API) that allows to connect and execute the query with the batabase.

JDBC Drivers : JDBC Driver is a Software component that enables java application to interact with the database.

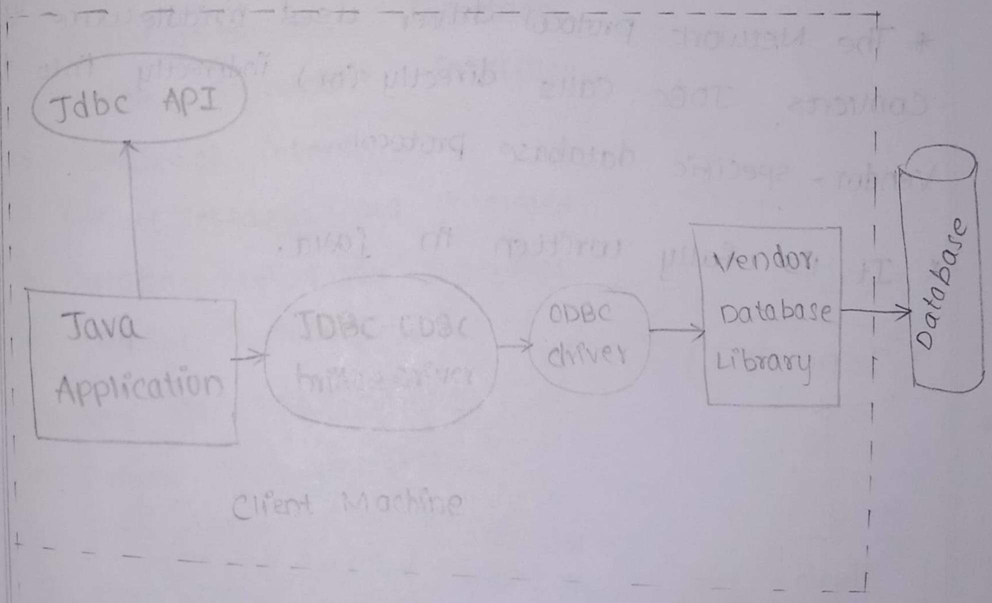
* Types Of JDBC Drivers :-

1. JDBC- ODBC Bridge driver :-

* The JDBC- ODBC bridge driver uses ODBC driver to connect to the database.

* The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls.

* This is now discouraged because Of thin driver.

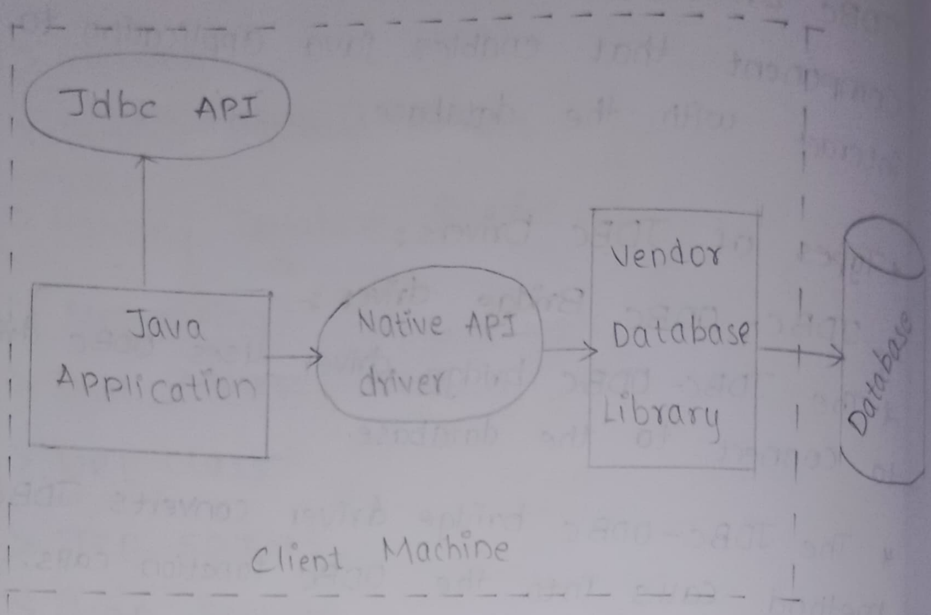


2. Native - API driver :-

* The Native API driver uses the client-side libraries of the database.

* The drivers converts JDBC method calls into native calls of the database API.

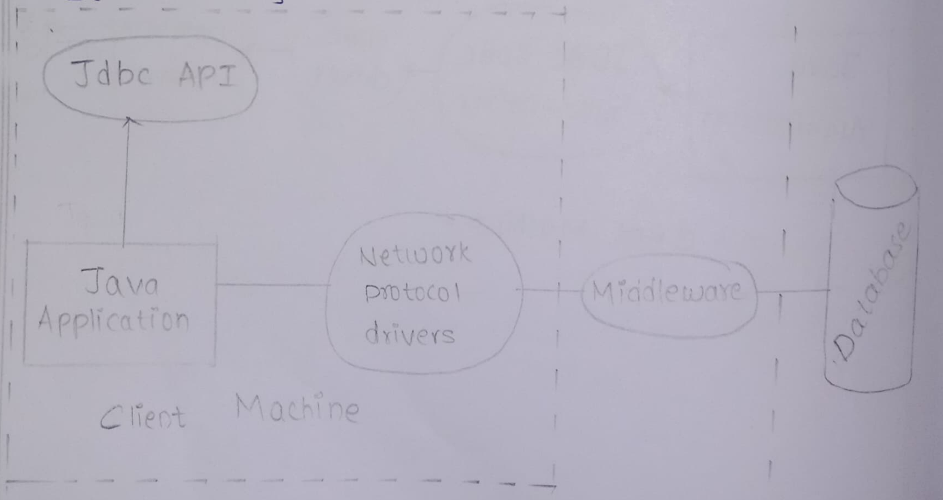
* It is not written entirely in java.



3. Network protocol driver :-

* The Network protocol driver used middleware that converts JDBC calls directly (or) indirectly into the Vendor - specific database protocol.

* It is fully written in java.

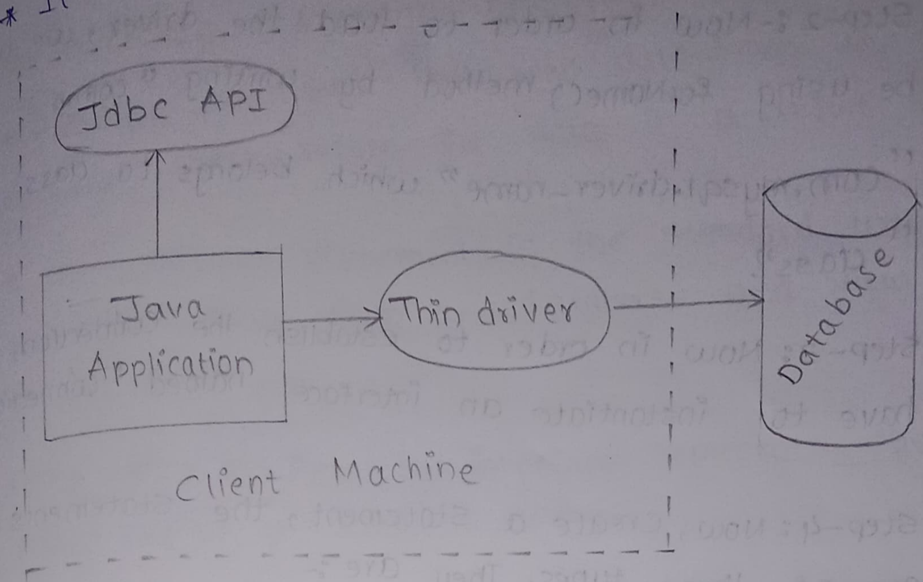


4. Thin driver :-

* The Thin driver converts JDBC calls directly into the vendor-specific database protocol.

* That is why it is knocon as thin driver.

* It is fully written in java language.



Interfaces of JDBC API :-

1. Driver interface
2. Connection interface
3. Statement interface
4. PreparedStatement interface.
5. Callable Statement interface.
6. ResultSet interface.
7. Result SetMeta Data interface
8. Database Meta Data interface.
9. Rowset interface.

Classes of JDBC API :-

1. Driver Manager Class
2. Blob class
3. Clob class
4. Types class

⇒ JDBC IMPLEMENTATION / CONNECTION CLASS / STATEMENTS :-

Step-1 :- First we need to import the package, for sql database it it namely "java.sql.*",

Step-2 :- Now in order to load the driver, we wi be using forName() method by writing " com "com.mysql.driver_name" which belongs to class "class".

Step-3 : Now in order to establish the Connection, we have to instantiate an interface named "connection".

Step-4 : Now, Create a Statement, the Statements are of three types. They are :-

i> Normal Statements : The Normal Statements interface represents the Static SQL Statements, it helps you to create a general purpose SQL statements using java.

ii> Prepared Statements : Prepared Statements represents a recompiled SQL Statements, that can be executed many times.

iii> Callable Statements : The Callable statements interface provides methods to execute stored procedures.

Step-5 : Once the statements is created, now we will be executing the query in our database.

step 6: Now, process the results.

step-7: Lastly, closing the connections to release and free up memory resources by closing the object of interface and statement objects.

⇒ CATCHING DATABASE RESULTS:

* The SQL statements that read data from a database query, return the data in a result set.

* The SELECT statement is the standard way to select rows from a database and view them in a result set.

* The java.sql.ResultSet interface represents the result set of a database query.

* The methods of the ~~re~~ ResultSet interface can be broken down into three categories. ~~Th~~ They are:

I. Navigational methods: Used to move the cursor around.

II. Get methods: Used to view the data in the columns of the current row being pointed by the cursor.

III. Update the methods: used to update the data in the columns of the current row.

* JDBC provides the following connection methods to create statements with desired ResultSet.

> Createstatement (int RsType, int Rs Concurrency);

> Prepare Statement (string SQL, int RsType, int Rs Concurrency);

=> HANDLING DATABASE QUERIES :-

1. Establish a connection to your Database: You can use libraries like "mysql" to connect to different types of database.

2. Write SQL queries: Depending on the type of database you are using, you would write SQL queries.

3. Execute the queries: Using the appropriate functions provided by the database library, you can execute the queries and retrieve the results.

4. Handle the results: Once you receive the results from the database, you can process them further, manipulate the data (or) display it on your web page.

Example: Using Node.js and MySQL. ``` javascript

``` javascript

~~~
Const mysql = require ("mysql");
//Create a connection to the database.
Const Connection = mysql.Create Connection
{
  host :'localhost';
  User : 'Username',
  Password : 'Password',
  database : 'mydatabase',
};

// Execute a query
Connection.query ('SELECT * FROM Users',
      function (error, results, fields)
  {
     if (error) throw error;
~~~
```

```
// process the results
   console.log(results);
 };
// close the connection
   connection.end();

 '''
```

⇒ NETWORKING :-

Definition : Networking is primarily achieved through HTTP (Hypertext Transfer Protocol) is a protocol that defines how client and servers communicate over the internet.

* Concepts of Networking :-

1. HTTP Requests : Clients can send different types of HTTP requests to servers.

2. AJAX : Asynchronous javascript and XML is a technique that allows you to send HTTP requests from a web page without reloading the entire page.

3. Fetch API : The Fetch API is a modern javascript API that provides an interface for making HTTP request.

4. RESTful APIs : REST (Representational State Transfer) is an architectural style that defines a set of principles for building web services.

5. Websocket : Websocket is a communication protocol that provides full-duplex communication between a client and a server over a single, long-lived connection.

⇒INET ADDRESS CLASS:-

Deffinition: Java Inet Address class represents a numerical IP address and domain name of the host.

* Methods of InetAddress Class:

> Inet Address get By Name (String host):

Returns the instance of Inet Address containing LocalHost IP and name.

> Inet Address getLocal Host():

It returns the instance of Inet Address containing local host name and address.

> String getHostName():-

It returns the host name of the IP address.

> String getHost Address():-

It returns the IP Address in String format.

Example:

```
import java.io*;
import java.net*;
Public class Inet Demo
{
    Public static void main (strings[ ] args)
    {
        try {
            Inet Address ip = InetAddress.getByName
                ("www.google.com");
            System.out.println("HostName: "ip.getHostName());
            System.out.println("IPAddress:"+ip.getHostAddress());
        }
        Catch(Exception e)
        {
            system.out.println(e);
        }
    }
}
```

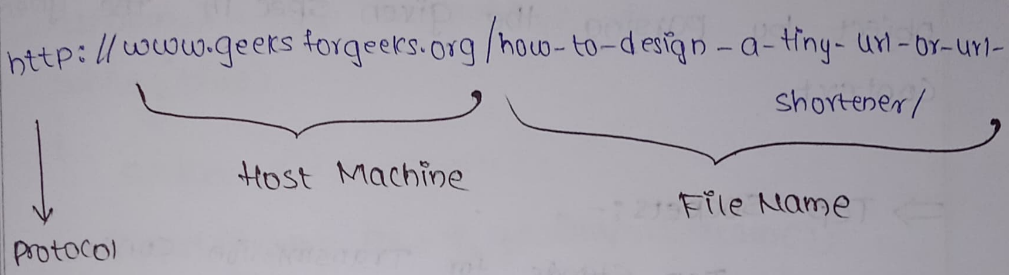
* Output :

Host Name : www.google.com

IP Address : 206.51.231.148.

⇒ URL CLASS :-

Definition : URL stands for Uniform Resource Locator is simply a string of text that identifiers all the resources on the internet, telling us the address of the resource, how to communicate with it, and retrive something from it.


http://www.geeksforgeeks.org/how-to-design-a-tiny-url-or-url-shortener/
Host Machine
File Name
protocol

* Components of a URL Class :

1. protocol : HTTP is the protocol here.

2. Hostname : Name of the machine on which the resource lives.

3. FileName : The PathName to the file on the machine.

4. Port Number : Port Number to which to connect (typically optional).

→* Constructors of the URL Class :-

1. URL (String address) throws Malformed URL Exception: It Creates a URL object from the specified string.

2. URL (String protocol, string host, String file):- Creates a URL object from the specified protocol, host, and filename.

3. URL (String protocol, stringhost, int port, String file):- Creates a URL objects from protocol, host, port, and filename.

4. URL (URL Context, String spec):- Creates a URL object by parsing the given spec in the given Context.

⇒ TCP Sockets:-

Definition:- TCP Stands for Transmission Control protocol, which allows for reliable Communication between two applications, TCP is typically used over the Internet protocol, which is pre° referred to as TCP/IP.

* Establishing a TCP Connection :-

* The Server instantiates a server Socket object, denoting which port number Communication is to occur on.

* The Server invokes the accept() method of the server Socket Class.

* This method waits until a Client Connects to the server on the given port.

* After the server is waiting, a client instantiates a socket object, specifying the server name and port number to connect to.

TCP program:-

server.java :

```
import java.net.* ;
import java.util.*;
import java.io.*;
class server
{
    public static void main (String args[]) throws
    Exception {
    server Socket ss = new ServerSocket (5555);
    Random r = new Random();
    while (true)
    {
        socket s = ss.accept();
        Output Stream os = s.get OutputStream();
        DataOutput stream dos = new DataOutput Stream(os);
        dos.write Int (r.next Int());
        s.close();
    }
  }
}
```

Client. java:

```
import java.net.*;
import java.util.*;
import java.io.*;
Class Client
{
  Public static void main (String argsL J) throws
  Exception
  {
    Socket S = new Socket (" local host", 5555);
    Input Stream is = s.get InputStream ();
    DataInputstream dis = new Data Input Stream (is);
    int i = dis.readInt();
    System.out.printIn ("random value received is="+i
    s.close();
  }
}
```

⇒ UDP Sockets :-

Definition: UDP stands for User Datagram Protocol, a Connection-less protocol that allows for packets of data to be transmitted, between applications.

* Classes of UDP :-

1. Datagram packet :-

* In UDP's terms, data transferred is encapsulated in a unit called datagram.

* You can create a Datagram packet object by using one of the following Constructions.

i. Datagram packet (byte[] buf, int length)

ii. Datagram packet (byte[] buf, int length,
Inet Address address, int port)

2. Datagram Socket :-

* You Use Datagram Socket to send and receve Datagram Packets.

* You can create a datagram Socket object by using one of the following constructors.

i. Datagram Socket().

ii. Datagram Socket(int port).

* UDP Program :

Receiver.java :

```
import java.net.* ;
import java.util.* ;
import java.io.* ;
Class Receiver
{
    public static void main(String args[]) throws
    Exception {
      Datagram Socket ds = new Datagram Socket (4444);
      byte.buf[] = new byte[60];
      while(true){
        Datagram Packet dp = new Datagram Packet
                              (buf, buf.length);

      ds.receive (dp);
      String s = new String(dp.getData());
      System.out.println(s);
    }
  }
}
```

Sender.java :-

```
import java.net.* ;
import java.util.* ;
import java.io.* ;
Class sender
{
    Public static void main (string args[]) throws
    Exception{
     Datagram Socket ds = new Datagram Socket();
     Inet Address i = Inet Address . get ByName("local host");
     bytebuf[ ] = args[0]. get Bytes();
     Datagram Packet dp = new Datagram Packet
                              (buf, buf. length, i, 4444);
     ds.send (dp);
    }
}
```

⇒ JAVA BEANS :-

Definition :- A Java Bean is a specially constructed java class written in the java and coded according to the java Beans API specifications.

POJO
plain older Java obj.

* JavaBeans Properties :-

1. getPropertyName() : For example, if property name is firstName, your method name would be getFirst Name() to read that property, this method is called accessor.

2. SetPropertyName() : For example, if property name is firstName, your method name would be setFirstName() to write that property, this method is called mutator.

Java Beans Example:

Consider a student class with few properties

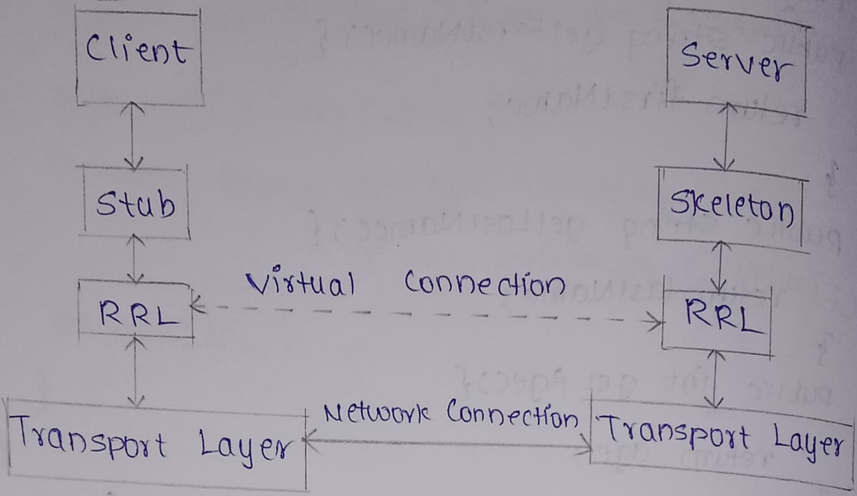
```
package edu.kmit;
public class studentsBean implements java.io.serializable
{
  private string firstName = null;
  private String lastName = null;
  private int age = 0;
  public StudentsBean() {}
  public String getFirstName() {
     return firstName;
  }
  public String getLastName() {
     return lastName;
  }
  public int get Age(){
     return age;
  }
  public void set FirstName(String firstName) {
     this.firstName = first Name;
  }
  public void set LastName (String lastName){
     this.lastName = lastName;
  }
  public void set Age(Integer Age) {
     this.age = age;
  }
}
```

⇒ RMI :

Definition: RMI stands for Remote Method Invocation. It is a mechanism that allows an object residing in one system to access/invoke an object running on another JVM.

* Architecture of an RMI :-

Definit



*Components of this architecture :

> Transport Layer : This Layer Connects the Client and the Server, it manages the existing Connection and also sets up new new Connections.

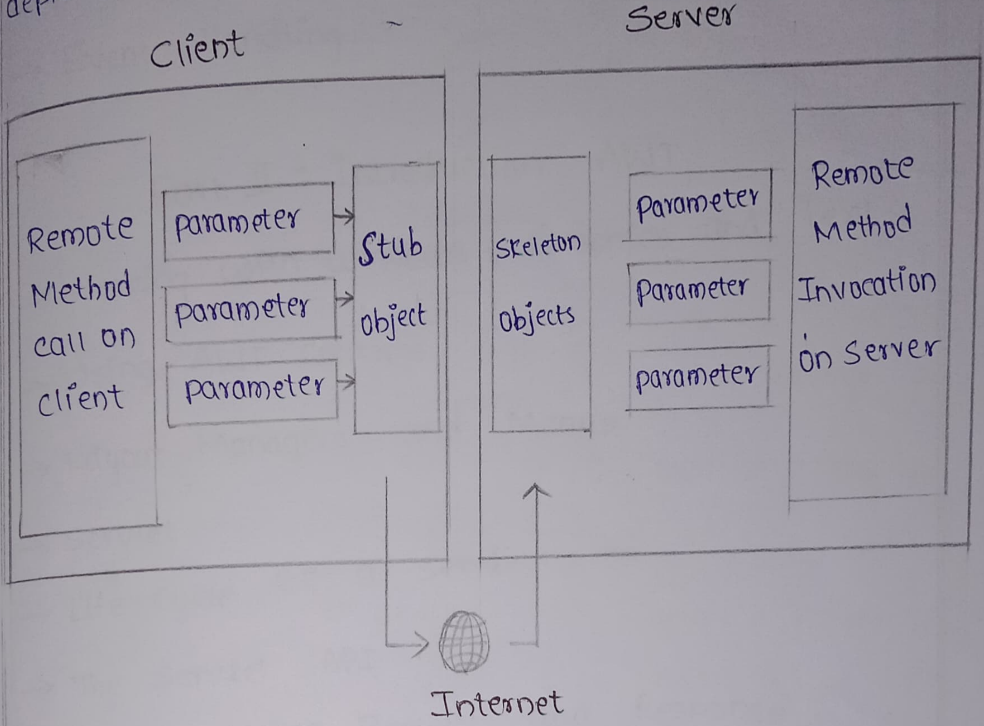
> Stub : A Stub is a representation of the remote object at client, It ~~refler~~ resides in the Client system; it acts as a gateway for the client program.

> Skeleton : This is the object which resides on the Server side, Stub Communicates with this skeleton to pass request to the remote object.

>RRL (Remote Reference Layer) :- It is the layer which manages the references made by the client to the remote Object.

*Working of RMI :-

* The Communication between client and server is handled by using two intermediate objects: stub object and skeleton object as also can be depicted from below media as follows:



* These are the steps to be followed sequentially to implement interface as defined bellow as follows:

Step-1: Defining a remote interface.

Step-2: Implementing the remote interface.

Step-3: Creating stub and skeleton objects from the implementation class using rmic.

Step-4 :- Start the rmi registry.

Step-5 :- Create and execute the server application program.

Step-6 :- Create and execute the Client application program.

UNIT – IV

part I : APPLETS

→ Java Applets

→ Life cycle of an applet

→ Adding images to an applet.

→ passing parameters to an applet.

→ Adding sound to an applet.

→ Event Handling.

Part II : Introduction AWT

→ Working with windows, Graphics and Text.

→ Using AWT Controls.

→ Layout Managers and Menus

→ Servlet

→ Life Cycle Of a Servlet

→ The Servlet API

→ Handling HTTP Request and Response

→ Using Cookies

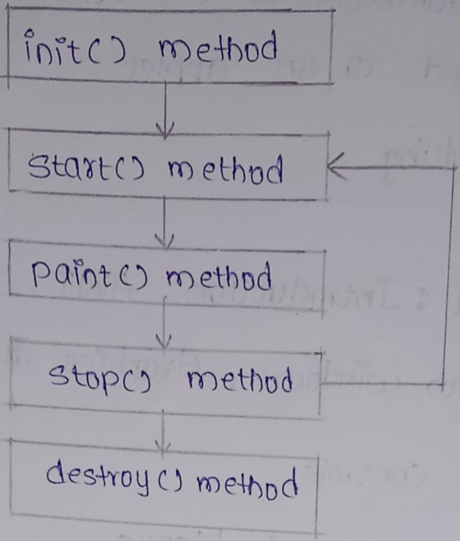
→ Session Tracking

→ Introduction to JSP

Part-I : APPLETS

⇒ **JAVA APPLET** :- Java Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

⇒ ## * LIFE CYCLE OF AN APPLET :-



I. init() method : It is used to initialized the Applet. It is invoked only once.

II. start() method : It is used to start the Applet. It is invoked after the init() method (or) browser is maximized.

III. paint() method : It is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc...

IV. stop() method : It is used to stop the Applet. It is invoked when Applet is stop (or) browser is minimized.

V. destroy() method : It is used to destroy the Applet. It is invoked only once.

ADDING IMAGES TO AN APPLET :-

```
import java.awt.* ;
import java.applet.* ;
/* <applet code = "App005" height = 600
                    width = 600></applet>*/

public class App005 extends Applet

{
  image picture;
  public void init()
  {
    picture = getimage(getDocumentBase(). "img1.jpg");

  }
  public void paint (Graphics g)
  {
    g.drawimage (picture, 30 , 30 , this);
  }
}
```

=> ADDING SOUND TO AN APPLET :-

```
import java.applet.* ;
import java.awt.* ;
import java.awt. event. * ;
/* <applet code = "APP006" height = 600 width = 600>
                                  </applet>*/

Public class APP006 extends Applet implements
                                  Action Listener
{
   Button b1 , b2 ;
   Audioclip audioclip ;
   Public void init()
   {
     b1 = new Button (" play");
     add (b1);
     b1
```

```
    b1.add ActionListener (this);
    b2 = new Button("stop");
    add (b2);
    b2.add ActionListener (this);
    audioclip = get AudioClip(get codeBase(), "audio.wa
  }
  Public void action performed (Action Event ae)
  {
    Button Source = (Button)ae.get Source();
    if(Source.get Label().equals("play"))
    {
      audio Clip.play();
    }
    else if (source.getLabel().equals ("stop"))
    {
      audio Clip.stop();
    }
  }
}
```

⇒ PASSING PARAMETERS TO AN APPLET:-

```
import java.applet.Applet;
import java.awt.Graphics;
/* <applet Code = "App004" width = 400 height = 400>
<param name = "Username" value = "shiva">
            </applet>*/

Public Class APP004 extends Applet
{
  String s;
  public void init()
  {
    s = get Parameter ("Username");
  }
public void paint (Graphics g)
{
    g.drawString (s, 100, 100);

}
}
```

⇒ EVENT HANDLING :-

* Event : Changing the state of an object is known as an event.

Ex: click on button, dragging mouse etc...

* Event Handling :- Java uses event delegation model for handling / processing the event.

* Terminology in event handling :

> Source : Like button, checkbox, applet etc...on which user has generated the event.

> Type of Event : Example if the user clicks on a button, the type of event is called as the action event. If the user clicks on the applet, it is mouse event.

> Listener : The class(s) who wants to know about a particular event and respond.

> Event Registration : The classes who are interested in knowing and processing any type of event on any source must register for the same.

* Event Delegation Model : When an even is generated on the source, JRE will create an object of corresponding Event class with the information about generate event and delegates to the registered listeners.

Now the listener class will invoke the corresponding method which processes the event.

* Event classes and Listener interfaces:

Action Event	ActionListener
Mouse Event	Mouse Listener and Mousel Listen
keyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
Adjustment Event	Adjustment Listener
Window Event	Window Listener.

* Handling Mouse Events - Example :-

```
//making use of inner classes
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
// <applet Code = "a9" width = 400 height = 400>
                                            </applet>
public class a9 extends Applet
{
  public void init()
  {
    add MouseListener (new Mouse Adapter()
    {
      Public void  mouseClicked (Mouse Event me)
      {
        Show status ("mouse Clicked");
      }
    }
    );
  }
}
```

* Handling Keyword Events - Example :-

```
import java.applet.* ;
import java.awt.* ;
import java.awt.event.* ;
//<applet code = "a6" width = 400 height = 400></applet>
public class a6 extends Applet implements
keyListener
{ int x = 30 , y = 30;
String s = "" ;
public void init()
{
add keyListener(this);
}
public void key Typed (key Event ke)
{
 St = ke.getkeyChar();
 repaint();
}
public void key pressed (key Event ke)
{
 showstatus("key pressed");
}
public void key Released (key Event ke)
{
 showstatus ("key Released");
}
Public void paint(Graphics g)
{
 g.draw String (s, x, y);
}
}
```

Part - II

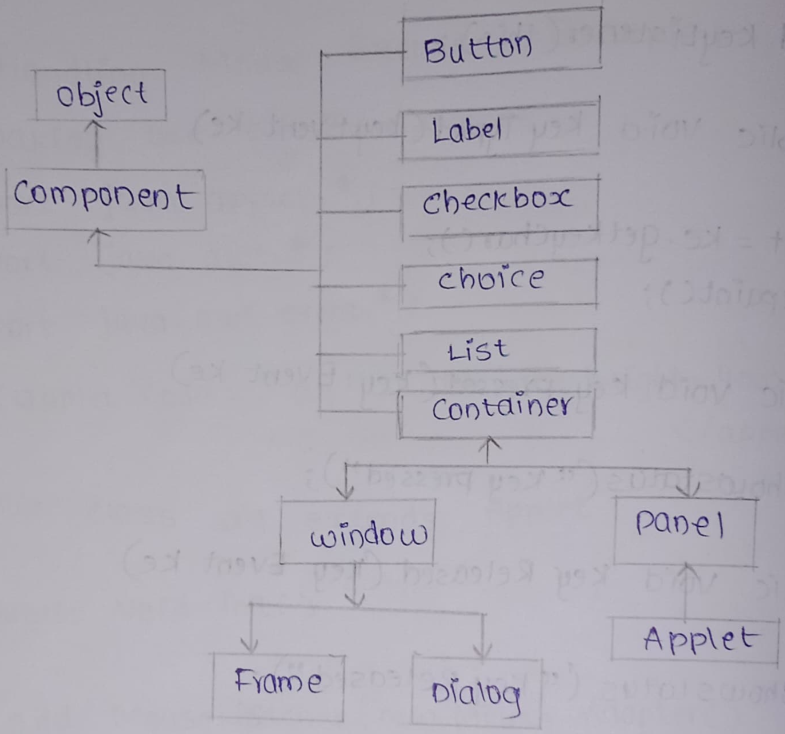
AWT

Introduction

=> JAVA AWT DEFINITION: Java AWT [Abstract Window Toolkit] is an API to develop GUI (or) window- based applications in Java.

=> WORKING WITH WINDOWS, GRAPHICS AND TEXT:-

* The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, choice, List etc....



* Container: The Container is a Component in AWT that can contain another components like buttons, textfields, labels etc...

* Window: The Window is the container that have no borders and menu bars. You must use frame, dialog (or) another window for creating a window.

* Panel: The Panel is the Container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc...

* Frame : The Frame is the container that contain title bar and can have menu bars. It can have other Components like button, textfield etc....

⇒ USING AWT CONTROLS :-

I. Labels : It is used to display a single line of read only text.

II. push Button : Contains a label and generates an event when presses.

III. Check Box : Contains a label and used to turn an option on (or) off. It consists of a small box with a check mark (or) not.

IV. Choice List : It creates a popup menu where user chooses.

V. Lists : Multiple choice and scrolling selection list.

vi. Scroll Bars : Scroll bars are used to select continuous values between a specified minimum and maximum. scroll bars may be oriented horizontally (or) vertically.

vii. TextField : Single line text Entry.

viii. Text Area : Multi line text editor.

⇒ LAYOUT MANAGERS AND MENU'S :

1. Border Layout : It is used to arrange the Components in five regions : north, South, east, West and Centre. Each region may contain one component only.

Ex :-

Container_component.add (Component, Border Layout. NORTH);

2. Grid Layout :- It is used to arrange the components in rectangular grid.

Ex :

```
Container_component. Set Layout (new GridLayout (3, 3));

Container_ Component. add (Component_1);
```

3. Flow Layout : It is used to arrange the components in a line , one after another. It is the default layout of applet (or) panel.

Ex :

```
Container_ Component. set Layout (new FlowLayout
                                  (FlowLayout. RIGHT));

Container_ Component. add (Component_1);
```

4. Card Layout : Class manages the Components in such a manner that only one component is visible at a time.

Ex :-

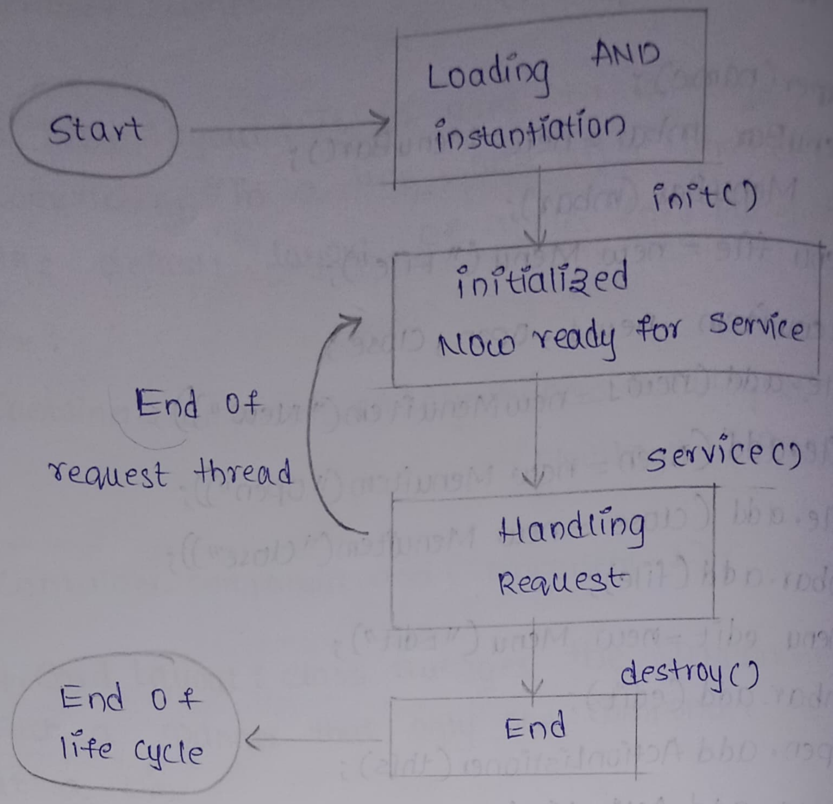
```
c = getContent Panel();
Card = new CardLayout (40, 30);
// Create CardLayout object with 40 hor space and
  30 ver space. set Layout (card);
b1 = new JButton ("Apple");
b2 = new JButton ("Boy");
c.add ("a", b1);
c. add ("b", b2);
```

* Menus - Example :

```
class myframe extends Frame implements
ActionListener
{ string msg = "";
  myframe(string name)
  { super(name);
   MenuBar mbar = new MenuBar();
   set MenuBar (mbar);
   Menu file = new Menu ("File");
   Menu item new1, open, Close;
   file.add (new1 = new Menuitem ("New"));
   file.add (open = new Menuitem ("open"));
   file.add (close = new Menuitem("Close"));
   mbar.add (file);
   Menu edit = new Menu ("Edit");
   mbar.add (edit);
   open.add ActionListioner (this);
   close.add ActionListioner(this);
   public void action performed (Action Event ae)
   {
     msg = (string) ae.get Action Comman d();
     repaint();
   }
   public void paint (Graphics g)
   {
     g.draw string (msg, 100, 100);
   }
  }
}
```

=> SERVLET DEFINITION:- Servlet technology is used to Create a web application (resides at Server side and generates a dynamic web pa

=> LIFE CYCLE OF A SERVLET:-



1. Servlet Class is loaded : The Classloader is responsible to load the Servlet Class. The Servlet Class is loaded when the first request for the Servlet is received by the web Container.

2. Servlet instance is Created : The web Container creates the instance of a Servlet after loading the servlet Class. The Servlet instance is Created only once in the servlet life Cycle.

3. init() method is invoked : The Web Container Calls the init method only once after Creating the Servlet instance. The init method is used to initialize the Servlet.

4. service() method is invoked: The web container calls the service method each time when request for the servlet ~~m~~ is received.

5. destroy() method is invoked: The web container calls the destroy method before removing the servlet instance from the service.

=> THE SERVLET API:-

Interfaces	Description
1. Servlet	Declares life cycle methods that all servlets must implement.
2. Servlet Config	Allows servlet to get initialization parameters.
3. Servlet Request	provides client request information to a servlet.
4. Servlet Response	Assist a servlet in sending a response to the client.

classes	Description
1. Generic Servlet	provides a basic implementation of the servlet interface for protocol independent servlets.
2. ServletInputStream	provides an input stream for reading binary data from a client request.
3. ServletOutputStream	provides an output stream for sending binary data to the client.
4. Servlet Exception	Defines a general exception, a servlet can throw when it encounters difficulty.

* Interfaces and classes from java.servlet. http packag

* Important interfaces in javax. servlet. http pac

> Http Servlet Request.

> Http Servlet Response

> Http Session.

* Important classes in javax. servlet. http Package.

> HttpServlet

> Cookie.

⇒ HANDLING HTTP REQUESTS AND RESPONSE :-

* The HTTP Servlet class provides specialized methods that handle the various type of HTTP request

* A Servlet developer typically overrides one of these methods : do Delete(), doGet(), doHead(), doOptions(), doPost(), doPut(), and doTrace().

Example:

```
//Simple Calculator Example
import java.io.*;
import javax.servlet.* ;
import javax.servlet.http.* ;
public class serv6 extends HttpServlet
{
 protected void doGet (HttpServlet Request request,
         Http Servlet Response response)
 throws servlet Exception , IO Exception
 {
  int x = Integer. parseInt (request. getParameter("t1"));
  int y = integer. parseInt (request. getParameter("t2"));
String opr=request.getParameter("s1");
opr=opr.trim();
int op=integer.parseInt(opr);
int result=0;
switch (op)
{
 case 1: result=x+y;
  break;
 case 2: result=x-y;
  break;
 case 3: result=x*y;
  break;
 case 4: result=x/y;
  break;
}
printwriter pw=response.getwriter();
pw.println("Result= "+result);
}
}
```

⇒ USING COOKIES:

* Cookie is a bit of information created by Server for the first client request and will sent back to the client browser.

* Creating & sending cookies to client:

```
Cookie.ck1 = new Cookie("username","pavani");
response.addCookie(ck1);
```

* Accessing all cookies coming from client:

```
printwriter out=response.getwriter();
Cookie ck[]=request.getCookies();
for(int i=0; i<ck.length; i++)
{
  Out.print(ck[i].getName()+" ");
  Out.print(ck[i].getValue());
}
}
```

=> SESSION TRACKING :-

* Session simply means a particular interval of time.

* Session Tracking is a way to maintain state (data) of user. It is also known as Session management in servlet.

* Creating an instance of current date and time and storing in session :-

```
Date d1 = new Date();
Http Session s1 = request.getSession();
s1.set Attribute("mydate", d1);
```

* Accessing the data (date object) from Session:-

```
Http Session = request.getSession(false);
Date d = (Date)s.get Attribute("mydate");
response.getwriter().println(d);
```

=> INTRODUCTION TO JSP :-

JSP DEFINITION: JSP stands for Java Server pages, it is a Server-side technology which is used for creating web applications, it is used to create dynamic web content.

* Features of JSP:-

> Coding in Jsp is easy: As it is just adding JAVA code to HTML/XML.

> Reduction in the length of code: In JSP we use action tags, custom tags etc...

→ Connection to Database is easier: It is easier to connect website to database and allows to read (or) write data easily to the database.

→ Make Interactive websites: In this we can Create dynamic web pages which helps uses to interact in real time environment.

* JSP Elements:

1. Expression: We can use this tag to output any data on the generated page. These data are automatically Converted to string and printed on the output stream.

Syntax:

JSP Expressions are: <%= "Anything" %>

2. Scriplets: In this tag we can insert any amount of valid java code and these Codes are placed in the jsp service method by the JSP engine.

Syntax: <% // java codes %>

3. Directives: A JSP "directive" starts with <%@ characters. In the directives, we can import packages, and define error handling pages (or) the session information of the JSP page.

Syntax: <%@ directive attribute= "value"%>

4. Declarations: This tag is used for defining the functions and variables to be used in the JSP.

Syntax:

```
<%!
// java codes
%>
```

-* Run a simple Jsp page :-

step-1: save the JSP file using ".jsp" extension (ex- "hello.jsp")

step-2: start the server.

step-3: place your application inside a folder.

step-4: To execute the Jsp script, simply start tomcat server and use a browser to bro an URL of the JSP page i.e.,

http://localhost : port number/your Application Context Root/jspfile then you will see the jsp file is being compiled.

Example Of a Jsp web page:

```
<HTML>
<HEAD>
<TITLE> A Web Page </TITLE>
</HEAD>
<BODY>
<%. out.println(" Hello there!"); %>
</BODY>
</HTML>
```

UNIT - V

XML AND WEB SERVICES.

→ XML
→ Introduction - Form Navigation
→ XML Documents
→ XSL
→ XSLT
→ Web Services
→ UDDI
→ WSDL
→ Java web services
→ Web resources.

* XML DEFINITION :-

* XML stands for eXtensible Markup Language.

* XML is a markup language much like HTML.

* XML was designed to store and transport data.

* XML was designed to ~~st~~ be self-descriptive.

* XML is a W3C Recommendation.

* Applications Of XML :

> Data transfer : You can use XML to transfer data between two systems that store the same data in different formats.

> Web applications : XML gives structure to the data that you can see on webpages. other website technologies, like HTML, work with XML to present consistent and relevant data to website visitors.

> Documentation : You can use XML to specify the structural information of any technical document other programs then process the document structure to present it flexibily.

> Data type : Many programming languages support XML as a data type. With this support you can easily write programs in other languages that work directly with XML files.

=> FORM NAVIGATION :-

* Form Navigation refers to the process of guiding users through different forms on a website.

* It involves creating a logical flow and providing intuitive controls for users to move between forms and complete their tasks.

* One Common approach to form navigation is using buttons (or) links to switch between forms.

* Another approach is using tabs (or) a tabbed interface to display different sections of a form.

* Additionally, form navigation can involve dynamically updating the form based on User input.

* The key is to design form navigation that is clear, intuitive and efficient, ensuring that users can easily navigate through the forms without confusion (or) frustration.

⇒ XML DOCUMENT :-

* An XML document is a basic unit of XML information composed of elements and other markup in an orderly package.

* An XML document can contains wide variety of data.

Example: BOOKS.xml

```
<?xml version=“1.0” encoding=“UTF-8”?> <book store>
<book store>
<book category=“cooking”>
<title lang=“en”> Everyday Italian </title>
<author> Giada De Laurentiis </author>
<year> 2005 </year>
  <price> 30.00</price>
 </book>
<book category=“children”>
 <title lang=“en”> Harry potter </titles
  <author> JK. Rowling </author>
  <year> 2005 </year>
  <price> 29.99 </price>
```

```
</book>
<book Category = "web">
<title lang = "en" Learning XML</title>
<author> Erik T. Ray</author>
<year> 2003 </year>
<price> 39.95 </price>
</book>
</bookstore>
```

⇒ XSL :

* Before learning XSLT, we should first understand XLS which stands for Extensible Stylesheet Language.

* It is similar to XML as CSS is to HTML.

* Need for XSL :

* In case of HTML document, tags are predefined such as table, div, and span; and the browser knows how to add style to them and display those using CSS styles.

* But in case of XML documents, tags are not predefined.

* In order to understand and style an XML document. World wide web Consortium (W3C) developed XSL which can act as XML based stylesheet Language.

* An XSL document specifies how a browser should render an XML document.

Following are the main parts of XSL :

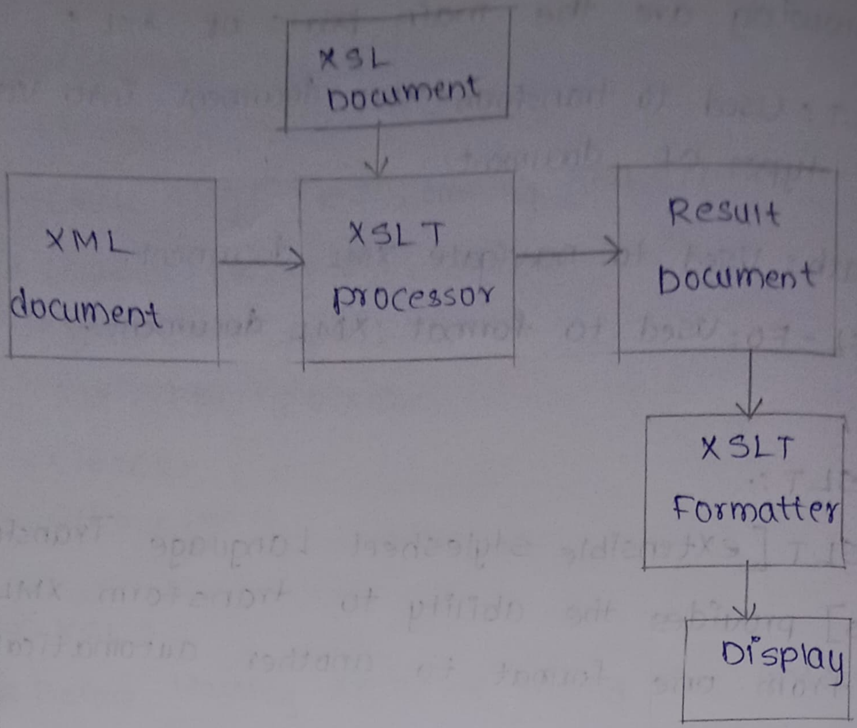
1. XSLT : Used to transform XML document into various other types of document.
2. Xpath : Used to navigate XML document.
3. XSL-FO : Used to format XML document.

⇒ XSLT :-

* XSLT [extensible stylesheet Language Transformations] provides the ability to transform XML data from one format to another automatically.

XSLT Working :

* An XSLT Stylesheet is used to define the transformation rules to be applied on the tag target XML document.
* XSLT Stylesheet is written in XML format.
* XSLT processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted XML document in the form of XML, HTML (or) text document in the format.
* This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.



⇒ WEB SERVICES:

* A Web Service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system.

* XML is used to encode all Communications to a web service.

Example:

* A Client invokes a web service by sending an XML message, then waits for a correspon-ding XML response.

* As all communication is in XML, web services are not tied to any one operating system (or) proprogramming language. java can talk with perl.

* Windows applications can talk with Unix applications.

⇒ UDDI :-

* UDDI stands for Universal Description Discovery and Integration.

* UDDI is a directory for storing information about web services.

* UDDI is a directory of web services interfaces described by WSDL.

* UDDI Communicates via SOAP.

* UDDI Based on:

* UDDI users World wide web Consortium (W3C) and internet engineering Task force (IETF) Internet Standards such as XML, HTTP and DNS protocols.

* UDDI uses WSDL to describe interfaces to web services.

UDDI Benefits:

* Any industry (or) businesses of all sizes can benefit from UDDI.

* Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and partners with information about their products and services.

-* Problems the UDDI specification can help to solve :-

* Making it possible to discover the right business from the millions currently online.

* Defining how to enable commerce once the preferred business is discovered.

How can UDDI be used :-

* Travel agencies could then search the UDDI directory to find the airline's reservation interface (i.e WSDL file)

* When the interface is found, the travel agency can communicate with the service using SOAP and WSDL.

=> WSDL :-

* WSDL stands for Web Services Description Language.

* WSDL is used to describe web Services.

* WSDL is written in XML.

* WSDL is a W3C recommendation from 26 June 2007.

-* WSDL Documents:

* An WSDL document describes a web service.

* It Specifies the location of the service, and the methods of the service, using these major elements.

* Elements of WSDL Document:

1. Types: Defines the (XML schema) for user defined data types used by the web service.

2. Message: Message defines a one-way request (or) response message consist of one (or) more parts.

3. ~~part~~ portType: Defines the operations that can be performed, and the messages that are involved.

4. binding: Binding provides details on how a port Type operation is transmitted over the wire.

* Service: defines a Collection of end points in terms of nested port elements.

The main structure of a WSDL document looks like this:-

```
<definitions>
<types>
data type definitions....
</types>
<message>
 definition of the data being
Communicated.....
<Imessage>
<port Type>
   Set of Operations....
</portType>
<binding>
 Protocol and data format
```

```
specification.....
</binding>
</definitions>
```

* WSDL Example :-

```
<message name = "get Term Request">
    <port name = "term"
      type = "xs:string"/>
        </message>
      <message name = "getTermResponse">
        <part name = "value"
         type = "xs:string"/>
        </message>
   <port Type name = "glossary Terms">
     <operation name = "getTerm">
        <input
      message = "get Term Request"/>
         <output
       message = "get TermResponse"/>
           </operation>
           </ portType>
```

⇒ JAVA WEB SERVICES:-

1. The Services that are accessible across various networks are java web services.

2. Since Java EE6, it has two API's defined by Java for creating web services Applications.

1. JAX - WS for SOAP web Services :-

* The jakarta EE API, known as JAX-WS, is used to develop and create web services, especially for SOAP Service users.

* There are mainly two ways to write JAX-WS application code. They are :-

i. RPC style

ii. Document style.

2. JAX- RS for RESTful web Services:

1. JAX-RS is a framework for building RESTfull web applications.

2. There are mainly two implementations for building JAX-RS. They are:

i. Jersey

ii. RESTeasy

* Implementation of Java web Services:

1. Implementing SOAP Web Services with JAX-WS:-

* First, you need to define service end point interfaces (SEI) which specify, the methods to expose as web service.

* Next, you need to implement SEI with a Java class.

* Then you need to Annotate the SEI and its implementation class with JAX-WX annotations to specify the web service.

* Package web service classes to the WAR file and deploy it to a web server.

2. Implementing RESTful web services with JAX-RS.

* First you need to define the resources that represents the web services and its methods.

* Then you need to annote the resource class and its methods with JAX-RS annotations to specify the web service package.

* At last we need to package the web service classes to the WAR file and deploy it to a web server.

=> WEB RESOURCES :-

* Web Resources are the resources which are required for proper rendering in the web application, it includes images, script files, and any user-created component libraries.

* The following are the some important points to store the resources.

i. It must be stored in a subdirectory of a resources directory at the web application root: resources/resource_identifier.

ii. A resource packaged in the web applications classpath must be in a subdirectory of the META-INF/resources directory within a web application: META-INF/resources/resource_identifier.

iii. you can use this file structure to package resources in a JAR file bundled in the web application.