CS-ELECTIVE -1

RCS-E12: WEB TECHNOLOGIES

Unit -1

Introduction: Introduction and Web Development Strategies, History of Web and Internet, Protocols Governing Web, Writing Web Projects, Connecting to Internet, Introduction to Internet services and tools, Introduction to client-server computing.

Core Java: Introduction, Operator, Data type, Variable, Arrays, Methods & Classes, Inheritance, Package and Interface, Exception Handling, Multithread programming, I/O, Java Applet, String handling, Event handling, Introduction to AWT,

AWT controls, Layout managers

INTERNET

The internet is a global network connecting millions of computer.

USES OF INTERNET

Communication

Using the internet is that we can communicate with the people living far away from us with extreme ease.

Research

Since the internet came into life, everything is available just a click away .

Education

There are lots and lots of websites which are related to different topic. You can visit them and can gain endless amount of knowledge that you wish to have.

Financial Transaction

This term is used when there is a exchange of mor
Your work has become a lot easier .

Real time Updates

It means that difference parts of the world, we come to know about it very easily and without any difficulty.

TABLE 7.3 MAJOR INTERNET SERVICES

CAPABILITY	FUNCTIONS SUPPORTED
E-mail	Person-to-person messaging; document sharing
Chatting and instant messaging	Interactive conversations
Newsgroups	Discussion groups on electronic bulletin boards
Telnet	Logging on to one computer system and doing work on another
File Transfer Protocol (FTP)	Transferring files from computer to computer
World Wide Web	Retrieving, formatting, and displaying information (including text, audio, graphics, and video) using hypertext links

Internet services and tools

CHAT

It is used for live discussions on the internet

E-MAIL

Exchanging electronic letters, messages, and small files.

FTP

It is a network protocol used to transfer files from one host to another.

HOSTING

The website needed files loading to server.

SEARCH ENGINE

It is a software system that is search for information on the world wide web.

WORLD WIDE WEB

It is a system of interlinked hypertext documents accessed via the internet.

URL
(UNIFORM RESOURCE LOCATOR)

- It is the global address of documents and other resources on the world wide web.
- Another name of URL is web address.

Example of URL:-

http::://www.teach-ict.com/index.html

Here,

http-Hyper Text Transfer Protocol
www-World Wide Web
www.teach-ict.com-Domain Name.
.com-Extension
index.html-Home page

CONNECTING TO INTERNET

If you can access the internet, there are three things you need:

1)Internet Service

It all depends on where you live and how much need you speed.

Eg:Cable,satellite,3G and 4G

2)Modem

The hardware of connecting to the internet is Modem.

It contains modulation and demodulation

Eg: DSL Modem, Cable Modem.....

3)Web Browser

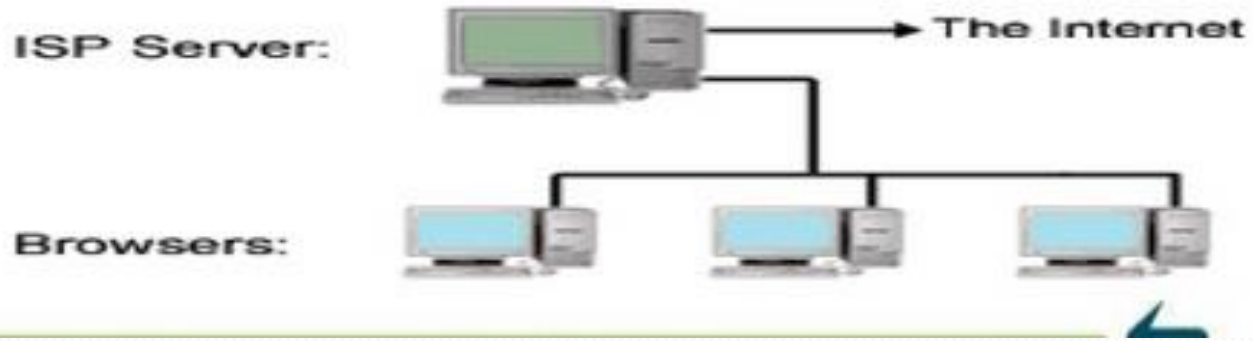
It is a tool to access the web. it's main job is display web pages.

Eg: Mozilla Firefox, Google Chrome, Opera.....

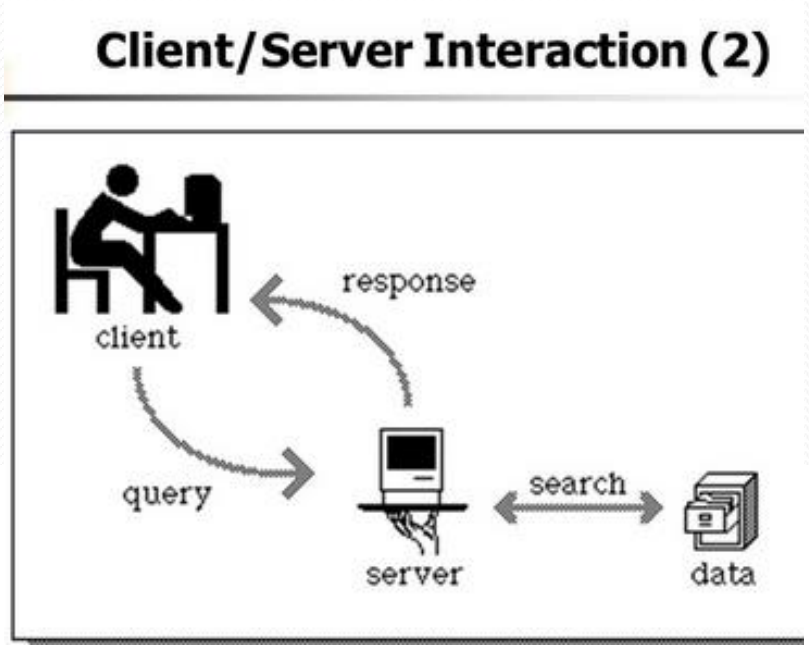
How does Internet work?

When we connect our computer to the internet, you are connecting to a special type of server which provided and operated by your internet service provider.

The picture shows that internet with several home computers connected to a server



Introduction to client-server computing



Client server computing

Introduction:

According to MIS terminology, Client/Server computing is new technology that yields solutions to many data management problems faced by modern organizations. The term Client/Server is used to describe a computing model for the development of computerized systems. This model is based on distribution of functions between two types of independent and autonomous processes: ***Server and Client***.

Client-Server Computing is divided into three components

- Client Process
- Server Process
- Middleware

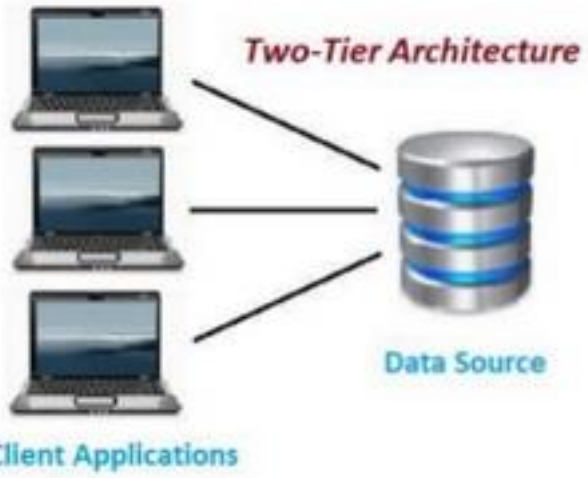
- A Client is any process that requests specific services from the server process. The main operations of the client system are to generates database request and transmits to server.

- Server :• A Server is a process that provides requested services for the Client. Accepts and processes database requests from client. Performs query/update processing and transmits responses to client.

- Middleware• Middleware is software that runs between client and server processes.• It makes it possible for them to communicate to each other.• Middleware is written in such a way that the user never notices its presence.• It delivers secure and transparent services to users.

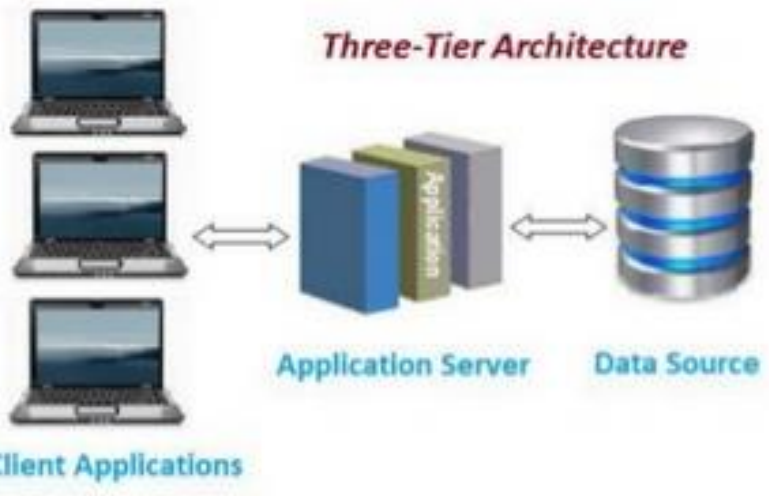
2-TIER ARCHITECTURE

- It is client-server architecture
- Direct communication
- Run faster(tight coupled)



3-TIER ARCHITECTURE

- Web based application
- Three layers:
  1) Client layer
  2) Business layer
  3) Data layer



Java

Java is a simple, object oriented, high performance language. It is distributed, portable, multi-threaded, and interpreted—mainly intended for the development of object oriented, network based software for Internet applications.

'Internet' implies heterogeneous systems, different network features, different windows libraries, and different operating systems. Java guarantees identical program behavior on different platforms.

What is java?

- A general-purpose object-oriented language.
- Write Once Run Anywhere (WORA).
- Designed for easy Web/Internet applications.
- Widespread acceptance.

Why Java is Important

- Two reasons :
  - Trouble with C/C++ language is that they are not portable and are not platform independent languages.
  - Emergence of World Wide Web, which demanded portable programs
- Portability and security necessitated the invention of Java

Characteristics of Java

- Java is simple
- Java is object-oriented
- Java is distributed
- Java is interpreted
- Java is robust
- Java is architecture-neutral
- Java is portable
- Java's performance
- Java is multithreaded
- Java is dynamic
- Java is secure

History

- James Gosling - Sun Microsystems
- Co founder – Vinod Khosla
- Oak - Java, May 20, 1995, Sun World
- JDK Evolutions
  - JDK 1.0 (January 23, 1996)
  - JDK 1.1 (February 19, 1997)
  - J2SE 1.2 (December 8, 1998)
  - J2SE 1.3 (May 8, 2000)
  - J2SE 1.4 (February 6, 2002)
  - J2SE 5.0 (September 30, 2004)
  - Java SE 6 (December 11, 2006)
  - Java SE 7 (July 28, 2011)

Cont..

- Java Editions.
  - **J2SE**(Java 2 Standard Edition) - to develop client-side standalone applications or applets.
  - **J2ME**(Java 2 Micro Edition) - to develop applications for mobile devices such as cell phones.
  - **J2EE**(Java 2 Enterprise Edition) - to develop server-side applications such as Java servlets and Java ServerPages.

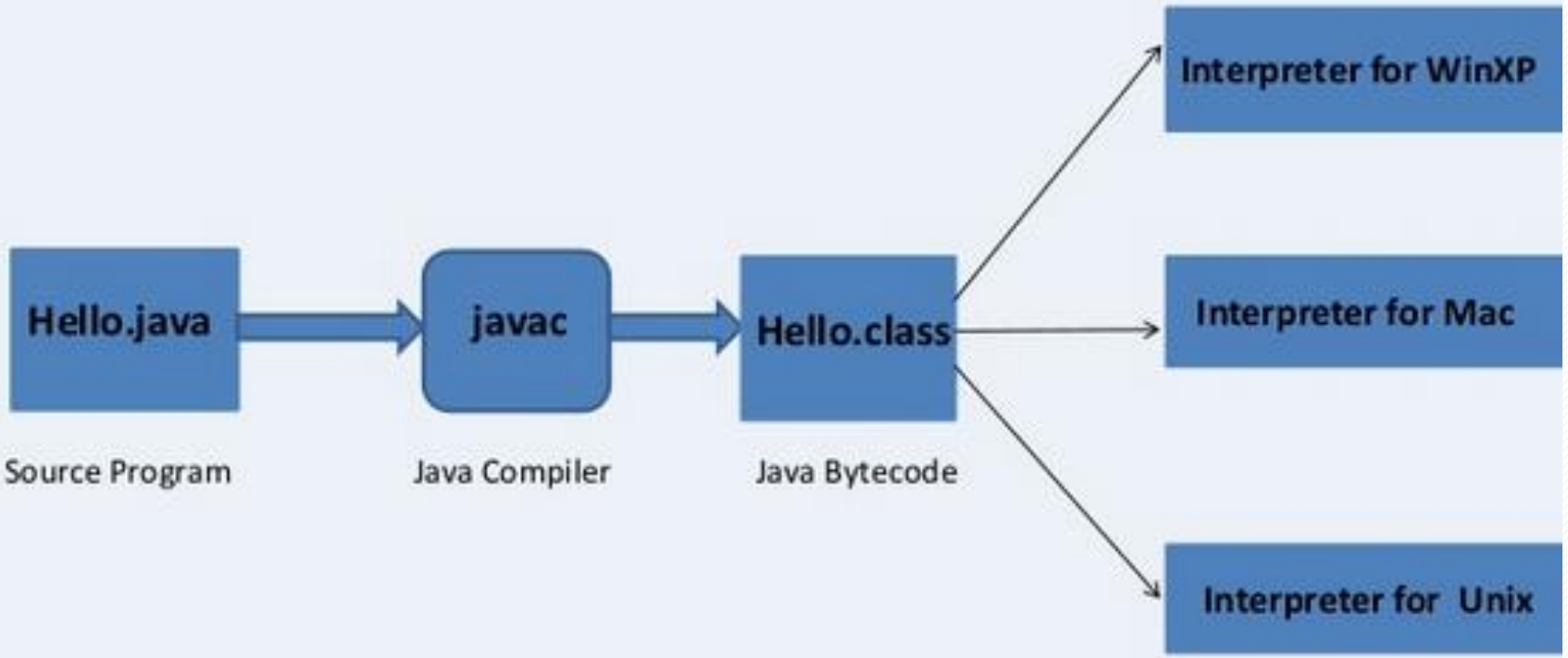
WHERE IS JAVA USED ?

According to the Sun , 3 billion devices run java.

There are many devices where Java is currently used.

- **Desktop Applications** - Acrobat reader, Media player, Antiviruses etc.
- **Web Applications** - irctc.co.in , javatpoint.com etc.
- **Enterprise Application** – Banking Application, Business Application.
- Mobile.
- Embedded System.
- Games.
- Robotics.

JAVA PROGRAM TRANSLATION

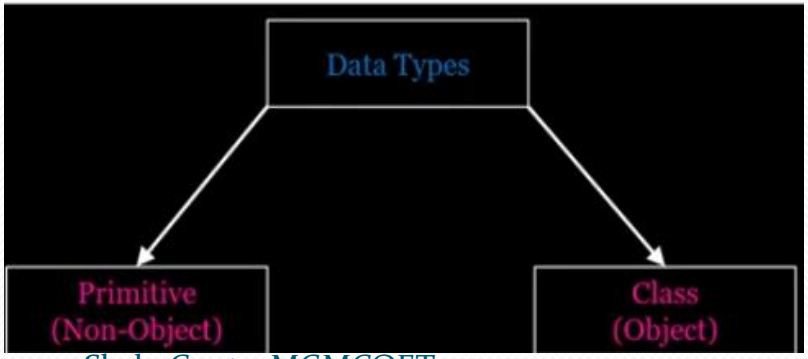


Data type

- Data type specifies the size and type of values that can be stored in an identifier. The Java language is rich in its data types. Different data types allow you to select the type appropriate to the needs of the application.

Data types in Java are classified into two types:

- Primitive—which include Integer, Character, Boolean, and Floating Point.
- Non-primitive—which include Classes, Interfaces, and Arrays.



Primitive Data type

A primitive data type can be of eight types :

Primitive Data types							
`char`	`boolean`	`byte`	`short`	`int`	`long`	`float`	`double`

Once a primitive data type has been declared its type can never change, although in most cases its value can change. These eight primitive type can be put into four groups

Integer

This group includes `byte` , `short` , `int` , `long`

byte : It is 1 byte(8-bits) integer data type. Value range from -128 to 127. Default value zero. example: `byte b=10;`

short : It is 2 bytes(16-bits) integer data type. Value range from -32768 to 32767. Default value zero. example: `short s=11;`

int : It is 4 bytes(32-bits) integer data type. Value range from -2147483648 to 2147483647. Default value zero. example: `int i=10;`

long : It is 8 bytes(64-bits) integer data type. Value range from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807. Default value zero. example: `long l=100012;`

Floating-Point Number

This group includes `float`, `double`

float : It is 4 bytes(32-bits) float data type. Default value 0.0f. example: `float ff=10.3f;`

double : It is 8 bytes(64-bits) float data type. Default value 0.0d. example: `double db=11.123;`

Characters

This group represent `char`, which represent symbols in a character set, like letters and numbers.

char : It is 2 bytes(16-bits) unsigned unicode character. Range 0 to 65,535. example: `char c='a';`

Boolean

This group represent `boolean`, which is a special type for representing true/false values. They are defined constant of the language. example: `boolean b=true;`

2) Non-Primitive(Reference) Data type

A reference data type is used to refer to an object. A reference variable is declare to be of specific and that type can never be change. We will talk a lot more about reference data type later in Classes and Object lesson.

Identifiers in Java

All Java components require names. Name used for classes, methods, interfaces and variables are called **Identifier**. Identifier must follow some rules. Here are the rules:

- All identifiers must start with either a letter(a to z or A to Z) or currency character($) or an underscore.
- After the first character, an identifier can have any combination of characters.
- A Java **keyword** cannot be used as an identifier.
- Identifiers in Java are case sensitive, foo and Foo are two different identifiers.

Variable in Java

- When we want to store any information, we store it in an address of the computer. Instead of remembering the complex address where we have stored our information, we name that address.The naming of an address is known as variable. Variable is the name of memory location.
- Java Programming language defines mainly three kind of variables.
- Instance variables
- Static Variables
- Local Variables

Instance variables in Java

Instance variables are variables that are declare inside a class but outside any method,constructor or block. Instance variable are also variable of object commonly known as field or property. They are referred as object variable. Each object has its c copy of each variable and thus, it doesn't effect the instance variable if one object changes the value of the variable.

```
class Student
{
    String name;
    int age;
}
```

Here **name** and **age** are instance variable of Student class.

Static variables in Java

Static are class variables declared with static keyword. Static variables are initialized only once. Static variables are also used in declaring constant along with final keyword.

```
class Student
{
    String name;
    int age;
    static int instituteCode=1101;
}
```

Here **instituteCode** is a static variable. Each object of Student class will share instituteCode property.

Additional points on static variable:

- static variable are also known as class variable.
- static means to remain constant.
- In Java, it means that it will be constant for all the instances created for that class.
- static variable need not be called from object.
- It is called by *classname.static variable name*

Note: A static variable can never be defined inside a method i.e it can never be a local variable.

Example:

Suppose you make 2 objects of class Student and you change the value of static variable from one object. Now when you print it from other object, it will display the changed value. This is because it was declared static i.e it is constant for every object created.

```
package studytonight;

class Student{
    int a;
    static int id = 35;

    void change(){

        System.out.println(id);
    }
}

public class StudyTonight {
    public static void main(String[] args) {

        Student o1 = new Student();
        Student o2 = new Student();

        o1.change();

        Student.id = 1;
        o2.change();
    }
}
```

OUTPUT:

```
35
1
```

Local variables in Java

Local variables are declared in method, constructor or block. Local variables are initialized when method, constructor or block start and will be destroyed once its end. Local variable reside in stack. Access modifiers are not used for local variable.

```
float getDiscount(int price)
{
 float discount;
 discount=price*(20/100);
 return discount;
}
```

Here **discount** is a local variable.

Concept of Array in Java

An array is a collection of similar data types. Array is a container object that hold values of homogenous type. It is also known as static data structure because size of an array must be specified at the time of its declaration.

An array can be either primitive or reference type. It gets memory in heap area. Index of array starts from zero to size-1.

Features of Array

- It is always indexed. Index begins from 0.
- It is a collection of similar data types.
- It occupies a contiguous memory location.

Array Declaration

Syntax :

```
datatype[] identifier;
or
datatype identifier[];
```

Both are valid syntax for array declaration. But the former is more readable.

Example :

```
int[ ] arr;
char[ ] arr;
short[ ] arr;
long[ ] arr;
int[ ][ ] arr;   // two dimensional array
```

Initialization of Array

`new` operator is used to initialize an array.

Example :

```
int[] arr = new int[10];    //this creates an empty array named arr of integer type whos
or
int[] arr = {10,20,30,40,50};  //this creates an array named arr whose elements are give
```

Accessing array element

As mention ealier array index starts from 0. To access nth element of an array. Syntax

```
arrayname[n-1];
```

Example : To access 4th element of a given array

```
int[ ] arr = {10,20,30,40};
System.out.println("Element at 4th place" + arr[3]);
```

The above code will print the 4th element of array `arr` on console.

Note: To find the length of an array, we can use the following syntax: `array_name.length`. There are no braces infront of length. Its not length().

Multi-Dimensional Array

A multi-dimensional array is very much similar to a single dimensional array. It can have multiple rows and multiple columns unlike single dimensional array, which can have only one full row or one full column.

Array Declaration

Syntax:

```
datatype[ ][ ] identifier;
or
datatype identifier[ ][ ];
```

Initialization of Array

new operator is used to initialize an array.

Example:

```
int[ ][ ] arr = new int[10][10];    //10 by 10 is the size of array.
or
int[ ][ ] arr = {{1,2,3,4,5},{6,7,8,9,10},{11,12,13,14,15}};
// 3 by 5 is the size of the array.
```

Java Operators

Java provides a rich set of operators environment. Java operators can be divided into following categories:

- Arithmetic operators
- Relation operators
- Logical operators
- Bitwise operators
- Assignment operators
- Conditional operators
- Misc operators

Arithmetic operators

Arithmetic operators are used in mathematical expression in the same way that are used in algebra.

Operator	Description
+	adds two operands
-	subtract second operands from first
*	multiply two operand
/	divide numerator by denumerator
%	remainder of division
++	Increment operator increases integer value by one
--	Decrement operator decreases integer value by one

Relation operators

The following table shows all relation operators supported by Java.

Operator	Description
`==`	Check if two operand are equal
`!=`	Check if two operand are not equal.
`>`	Check if operand on the left is greater than operand on the right
`<`	Check operand on the left is smaller than right operand
`>=`	check left operand is greater than or equal to right operand
`<=`	Check if operand on left is smaller than or equal to right operand

Logical operators

Java supports following 3 logical operator. Suppose a=1 and b=0;

Operator	Description	Example
&&	Logical AND	(a && b) is false
||	Logical OR	(a || b) is true
!	Logical NOT	(!a) is false

Bitwise operators

Java defines several bitwise operators that can be applied to the integer types long, int, short, char and byte

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
<<	left shift
>>	right shift

Now lets see truth table for bitwise `&`, `|` and `^`

a	b	a & b	a | b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The bitwise shift operators shifts the bit value. The left operand specifies the value to be shifted and the right operand specifies the number of positions that the bits in the value are to be shifted. Both operands have the same precedence.

Example:

```
a = 0001000
b = 2
a << b = 0100000
a >> b = 0000010
```

Assignment Operators

Assignment operator supported by Java are as follows:

Operator	Description	Example
`=`	assigns values from right side operands to left side operand	`a = b`
`+=`	adds right operand to the left operand and assign the result to left	`a+=b` is same as `a=a+b`
`-=`	subtracts right operand from the left operand and assign the result to left operand	`a-=b` is same as a `=a-b`
`*=`	mutiply left operand with the right operand and assign the result to left operand	`a*=b` is same as `a=a*b`
`/=`	divides left operand with the right operand and assign the result to left operand	`a/=b` is same as `a=a/b`
`%=`	calculate modulus using two operands and assign the result to left operand	`a%=b` is same as `a=a%b`

Misc. operator

There are few other operator supported by java language.

Conditional operator

It is also known as ternary operator and used to evaluate Boolean expression,

```
epr1 ? expr2 : expr3
```

If `epr1` Condition is true? Then value `expr2` : Otherwise value `expr3`

`instanceOf` operator

This operator is used for object reference variables. The operator checks whether the object is of particular type (class type or interface type)

Java - Object and Classes

Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts –

- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Passing

Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

- **Real life example of polymorphism:** A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.
- Polymorphism is considered as one of the important features of Object Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word "poly" means many and "morphs" means forms, So it means many forms.

- **In Java polymorphism is mainly divided into two types:**
- Compile time Polymorphism
- Runtime Polymorphism

Inheritance

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

Encapsulation

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.

To achieve encapsulation in Java –

- Declare the variables of a class as private.
- Provide public setter and getter methods to modify and view the variables values.

Benefits of Encapsulation

- The fields of a class can be made read-only or write-only.
- A class can have total control over what is stored in its fields.

Abstraction

Abstraction is the quality of dealing with ideas rather than events. For example, when you consider the case of e-mail, complex details such as what happens as soon as you send an e-mail, the protocol your e-mail server uses are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver, and click send.

Likewise in Object-oriented programming, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

Classes

- A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type. In general, class declarations can include these components, in order:
- **Modifiers** : A class can be public or has default access.
- **Class name:** The name should begin with a initial letter (capitalized by convention).
- **Superclass(if any):** The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
- **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
- **Body:** The class body surrounded by braces, { }.

Objects

- It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of :

- **State** : It is represented by attributes of an object. It also reflects the properties of an object. Example: A dog has states - color, name, breed

- **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other objects. Example – wagging the tail, barking, eating.

- **Identity** : It gives a unique name to an object and enables one object to interact with other objects.

Instance

Instance variable in **Java** is used by Objects to store their states. Variables which are defined without the STATIC keyword and are Outside any method declaration are Object specific and are known as **instance** variables. They are called so because their values are **instance** specific and are not shared among **instances**.

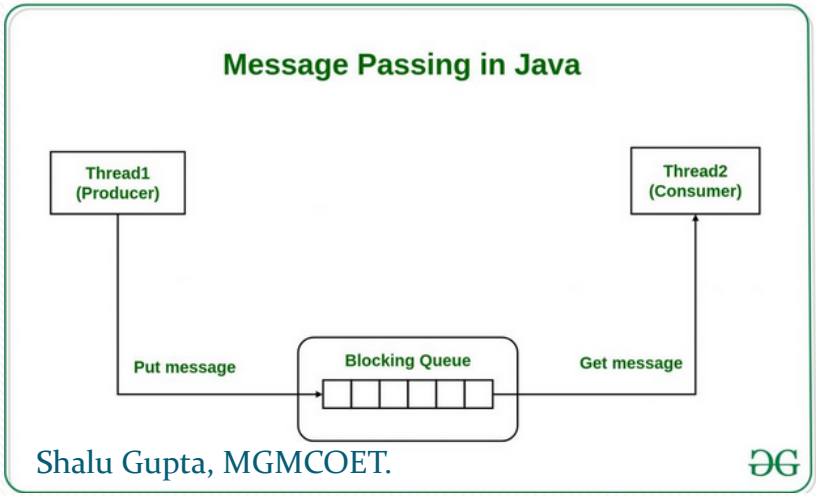
Method

- A Java method is a collection of statements that are grouped together to perform an operation. When you call the System.out.**println()** method, for example, the system actually executes several statements in order to display a message on the console.

- Now you will learn how to create your own methods with or without return values, invoke a method with or without parameters, and apply method abstraction in the program design.

Message Passing

- Message Passing in terms of computers is communication between processes. It is a form of communication used in object-oriented programming as well as parallel programming. Message passing in Java is like sending an object i.e. message from one thread to another thread. It is used when threads do not have shared memory and are unable to share monitors or semaphores or any other shared variables to communicate. Suppose we consider an example of producer and consumer, likewise what produce will produce, the consumer will be able to consume that only. We mostly use Queue to implement communication between threads.



Exceptions in Java

- **Exceptions in Java**
- **What is an Exception?**
- An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.
- **Error vs Exception**
- **Error:** An Error indicates serious problem that a reasonable application should not try to catch.
  Exception: Exception indicates conditions that a reasonable application might try to catch.

- **How Programmer handles an exception?**
- **Customized Exception Handling :** Java exception handling is managed via five keywords: **try**, **catch**, throw, throws, and **finally**. Briefly, here is how they work. Program statements that you think can raise exceptions are contained within a try block. If an exception occurs within the try block, it is thrown. Your code can catch this exception (using catch block) and handle it in some rational manner.

Java - Files and I/O

- The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc.
- **Stream**
- A stream can be defined as a sequence of data. There are two kinds of Streams –
- **InPutStream** – The InputStream is used to read data from a source.
- **OutPutStream** – The OutputStream is used for writing data to a destination.

- Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionality related to streams and I/O.

Multithreading in Java

- Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

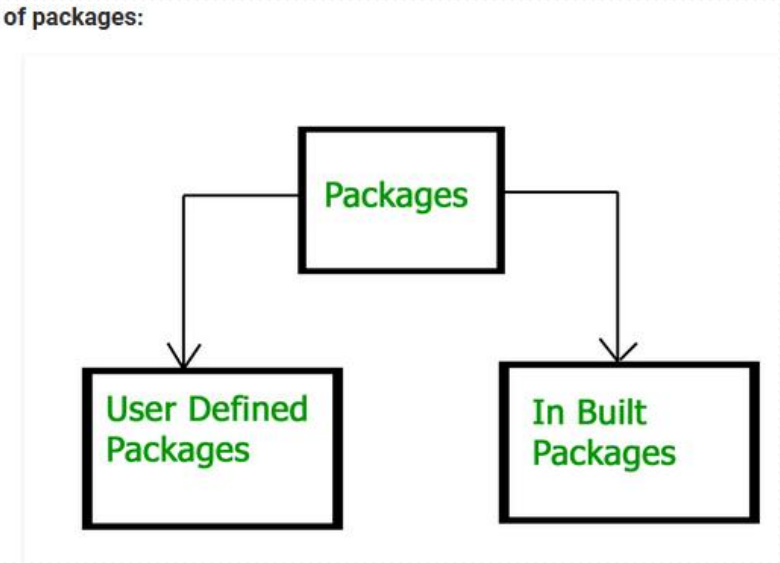
- Java is a *multi-threaded programming language* which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

- Threads can be created by using two mechanisms :
  1. Extending the Thread class
  2. Implementing the Runnable Interface

Package and Interface

- Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces. In other words, a package in Java refers to a collection of classes, interfaces, abstract classes, and exceptions that will help in a module in Java programming.

Types of packages:



Differences Between **Packages and Interfaces in Java**

- A **package** is a group of classes and **interfaces** together whereas, an **interface** is a group of abstract methods. **Package** is created using a keyword **package** whereas, an **interface** is created using a keyword **interface**.

Applet

- **What is Applet?**
  An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.
- Applets are used to make the web site more dynamic and entertaining.
- **Important points :**
- All applets are sub-classes (either directly or indirectly) of *java.applet.Applet* class.
- Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
- In general, execution of an applet does not begin at main() method.
- Output of an applet window is not performed by *System.out.println()*. Rather it is handled with various AWT methods, such as *drawString()*.

String handling

Strings, which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings.

The most direct way to create a string is to write –

```
String greeting = "Hello world!";
```

What is an Event?

- Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page are the activities that causes an event to happen.
- **Types of Event**
- The events can be broadly classified into two categories:
- **Foreground Events** - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the page etc.
- **Background Events** - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

Event Handling

- **Event Handling** is the mechanism that controls the **event** and decides what should happen if an **event** occurs. This mechanism have the code which is known as **event handler** that is executed when an **event** occurs. **Java** Uses the Delegation **Event** Model to **handle** the **events**. ... **Java** provide as with classes for source object.

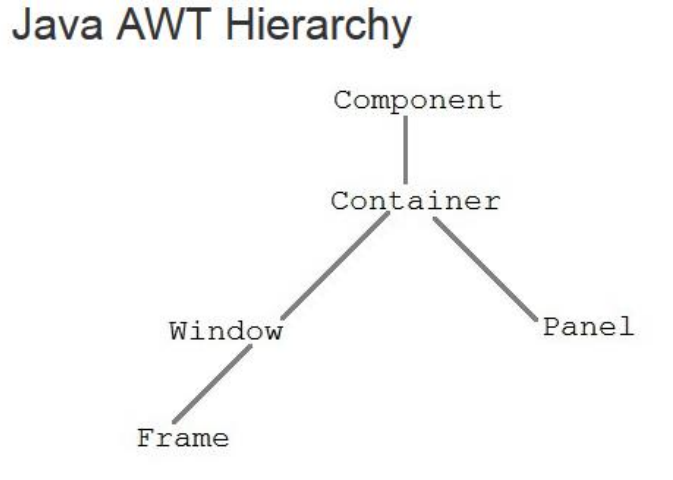
Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener

AWT (Abstract Window Toolkit)

- AWT contains large number of classes and methods that allows you to create and manage graphical user interface (GUI) applications, such as windows, buttons, scroll bars,etc. The AWT was designed to provide a common set of tools for GUI design that could work on a variety of platforms.

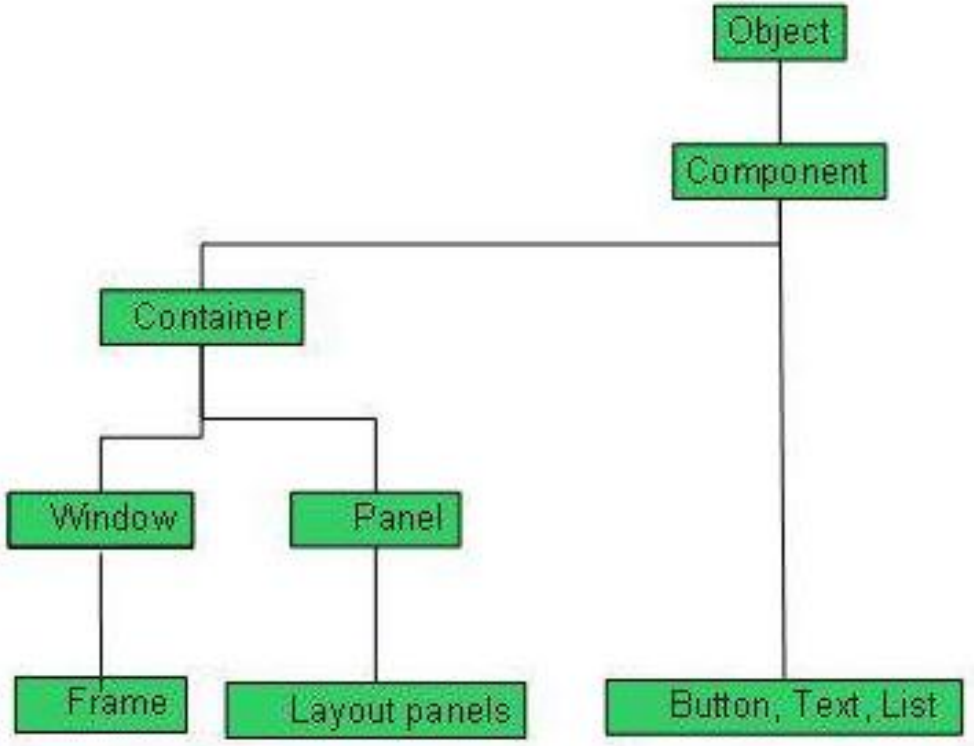
The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.



AWT Controls in Java with Examples

The AWT supports the following types of controls:

- Labels
- Push buttons
- Check boxes
- Choice lists
- Lists
- Scroll bars
- Text Area
- Text Field



Java LayoutManagers

The LayoutManagers are used to arrange components in a particular manner. LayoutManager is an interface that is implemented by all the classes of layout managers. The layout manager automatically positions all the components within the container. If we do not use layout manager then also the components are positioned by the default layout manager. It is possible to layout the controls by hand but it becomes very difficult because of the following two reasons.

- It is very tedious to handle a large number of controls within the container.
- Oftenly the width and height information of a component is not given when we need to arrange them.

There are following classes that represents the layout managers:

- java.awt.BorderLayout
- java.awt.FlowLayout
- java.awt.GridLayout
- java.awt.CardLayout
- java.awt.GridBagLayout
- javax.swing.BoxLayout
- javax.swing.GroupLayout
- javax.swing.ScrollPaneLayout
- javax.swing.SpringLayout etc.