Object Oriented Programming with Java (CSE1221)

1. What is Java?

Ans: Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

2. What is Java platform?

Ans: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

3. Write down the application of Java?

Ans: According to Sun, 3 billion devices run Java. There are many devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, javatpoint.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

4. What is JVM?

Ans: Java Virtual Machine (JVM) is a specification that provides **runtime environment** in which java bytecode(.class files) can be executed. The **JVM** is the platform. As the name implies, the JVM acts as a **"virtual"** machine or processor. Java's platform independence consists mostly of its Java Virtual Machine (JVM) . JVM makes this possible because it is aware of the specific instruction lengths and other particularities of the platform (Operating System).

5. Why Java is platform independent?

Ans: A programming language or technology is said to be platform independent if and only if which can run on all available operating systems with respect to its development and compilation. (Platform represents Operating System).

Java is a platform independent programming language, Because when you install jdk software on your system then automatically JVM are installed on your system. For every operating system separate JVM is available which is capable to read the **.class** file or **byte code**. When we compile your Java code then .class file is generated by javac compiler these codes are readable by JVM and every operating system have its own JVM so JVM is platform dependent but due to JVM java language is become platform independent.

6. What are the key features of Java?

Ans: Java is

1. **Object Oriented** − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
2. **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform-independent bytecode. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
3. **Simple** − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
4. **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
5. **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
6. **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
7. **Robust** − Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.
8. **Multithreaded** − With Java multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
9. **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
10. **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
11. **Distributed** − Java is designed for the distributed environment of the internet.
12. **Dynamic** − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

7. What is final Keyword?

Ans: The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

1. variable
2. method
3. class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only.

8. Why Java is called the language of internet?

Ans: Java was one of the most premiere languages on the web in the early days of the internet, as Java applets were in widespread use. Today, Java is still a very powerful language on the web, but doesn't have the dominance that it once had, with the widespread discontinuation of applets.

In fact, one of the reasons Java gained prominence was because of its portability. This made it very web friendly.

However, the adoption of languages more suitable for the web emerged, such as PHP, Python, Ruby, etc. that gained traction and largely displaced Java.

Today, Java is still used for the web, especially for server-side applications. In fact, Java has great performance and scalability for web applications. This makes it used widely in some of the biggest and most powerful sites. One such notable example is Amazon.com, which utilizes Java for backend applications.

Java is mostly used for large web projects, with smaller to medium sites not needing such tremendous power.

One of the reasons Java is used for larger websites than other languages such as PHP or Python or Ruby is because it's faster. Once a website grows to a certain point and it gets so many users and database requests and page views, it has to scale fast. The language on the server has to be very fast, processing all these user requests, without failing.

Java is the best for this. PHP, Python, Ruby are slower, so if a website gets to a certain point and there are constant server-side requests, it won't scale well.

An example of this is Twitter, which at first supported by the Ruby on Rails web framework. Once it got to a certain point of usage, it no longer scaled well and was switched over to Java's JVM. This made it quicker and scale much better.

So Java today on the web still has its place and is still tremendously powerful. However, for small to medium websites, it can be seen as unnecessary, being that a language such as PHP, Python, or Ruby can do the job well. Small or medium websites simply would not have the traffic and server-side requests that these languages wouldn't be able to manage.

But for extremely large websites such as Amazon.com, Java is powerful and allows for tremendous scaling that other languages would have difficulty achieving or could not achieve.

9. Write down the difference between Java and C++.
Ans: There are many differences and similarities between the C++ programming language and Java. A list of top differences between C++ and Java are given below:

Basic of Comparison	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.

10. Difference between JRE, JVM and JDK.

Ans: Following are the important differences between JDK, JRE, and JVM

JDK	JRE	JVM
JDK (Java Development Kit) is a software development kit to develop applications in Java. In addition to JRE, JDK also contains the number of development tools (compilers, JavaDoc, Java Debugger, etc.).	JRE (Java Runtime Environment) is the implementation of JVM and is defined as a software package that provides Java class libraries, along with Java Virtual Machine (JVM), and other components to run applications written in Java programming.	JVM (Java Virtual Machine) is an abstract machine that is platform-dependent and has three notions as a specification, a document that describes requirement of JVM implementation, implementation, a computer program that meets JVM requirements, and instance, an implementation that executes Java byte code provides a runtime environment for executing Java byte code.
JDK is primarily used for code execution and has prime functionality of development.	On the other hand, JRE is majorly responsible for creating environment for code execution.	JVM, on the other hand, specifies all the implementations and responsible to provide these implementations to JRE.
JDK is platform-dependent i.e for different platforms different JDK required.	Like of JDK JRE is also platform dependent.	JVM is platform-independent.
JDK is responsible for prime development so it contains tools for developing, debugging and monitoring java application.	On the other hand, JRE does not contain tools such as compiler or debugger etc. Rather it contains class libraries and other supporting files that JVM requires to run the program.	JVM does not include software development tools.
JDK = Java Runtime Environment (JRE) + Development tools	JRE = Java Virtual Machine (JVM) + Libraries to run the application	JVM = Only Runtime environment for executing the Java byte code.

11. How to create an ArrayList in Java?

Ans: In Java, we can create `ArrayList` by creating this simple statement:

ArrayList<String> arlist = new ArrayList<String>( );

In above syntax, list is of "String" type, so the elements are that going to be added to this list will be string type. The type decide which type of elements list will have.

ArrayList<String> arlist = new ArrayList<Integer>( );

Above syntax, is accepts int elements.

12. How to add and remove element in ArrayList?

Ans: To add an element in ArrayList, we can use add( ) method. This method has variations, which is use depends on requirements.

Syntax

arlist.add("JavaTpoint");

Add elements at the particular location, we can write method like this:

arlist.add(2, "Fahim");

To remove an element in ArrayList, we can use the remove( ) method. This method also has variations.

13. Write down the difference between Array and ArrayList.

Ans: The following table describes the key differences between array and ArrayList:

Basis	Array	ArrayList
Definition	An **array** is a dynamically-created object.	The **ArrayList** is a class of Java **Collections** framework.
Resizable	An array is a **fixed-length** data structure.	ArrayList is a **variable-length** data structure.
Single/ Multi-Dimensional	Array can be **multi-dimensional**.	ArrayList is always **single-dimensional**.
Adding Elements	We can add elements in an array by using the **assignment** operator.	Java provides the **add()** method to add elements in the ArrayList.
Generics	Generics are not compatible with Arrays.	ArrayLists allow the use of Generics.
Iterating Values	We use **for** loop or **for each** loop to iterate over an array.	We use an **iterator** to iterate over ArrayList.
Performance	It performs **fast** in comparison to ArrayList because of fixed size.	ArrayList is internally backed by the array in Java. The resize operation in ArrayList slows down the performance.

14. Write down the difference class and object in java?

Ans: There are many differences between object and class. A list of differences between object and class are given below:

Object	Class
Object is an **instance** of a class.	Class is a **blueprint or template** from which objects are created.
Object is a **physical** entity.	Class is a **logical** entity.
Object is created **many times** as per requirement.	Class is declared **once**.
Object **allocates memory when it is created**.	Class **doesn't allocated memory when it is created**.
Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.	Class is a **group of similar objects**.
Object is created through **new keyword** mainly e.g. Student s1=new Student();	Class is declared using **class keyword** e.g. class Student{}

15. What is static keyword?

Ans: The **static keyword** in Java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class.

16. What is static method?

Ans: Static method in Java is a method which belongs to the class and not to the object. A static method can access only static data. Static methods do not use any instance variables of any object of the class they are defined in. Static methods take all the data from parameters and compute something from those parameters, with no reference to variables.

Syntax: ***<class-name>.<method-name>***

17. Write the difference between method overloading and overriding.

Ans: There are many differences between method overloading and method overriding in java. A list of differences between method overloading and method overriding are given below:

Method Overloading	Method Overriding
Method overloading is a compile time polymorphism.	Method overriding is a run time polymorphism.
It is occur within the class.	It occurs in two class; sub class and super class
In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
Method overloading may or may not require inheritance.	While method overriding always needs inheritance.
It is performed at compile time.	It is performed at runtime.
Overloaded methods use static binding.	Overridden methods use dynamic binding.
You can perform overloading on static methods.	You cannot perform overriding on static methods.

18. Write about Java access modifier?

Ans: The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.

Let's understand the access modifiers in Java by a simple table.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

19. What are primitive and non-primitive data type?
Ans: A data type is a classification of data, which can store a specific type of information. Data types are primarily used in computer programming, in which variables are created to store data. Each variable is assigned a data type that determines what type of data the variable may contain.
The term "data type" and "primitive data type" are often used interchangeably. Primitive data types are predefined types of data, which are supported by the programming language. For example, integer, character, and string are all primitive data types. Programmers can use these data types when creating variables in their programs. For example, a programmer may create a variable called "lastname" and define it as a string data type. The variable will then store data as a string of characters.
Non-primitive data types are not defined by the programming language, but are instead created by the programmer. They are sometimes called "reference variables," or "object references," since they reference a memory location, which stores the data. In the Java programming language, non-primitive data types are simply called "objects" because they are created, rather than predefined. While an object may contain any type of data, the information referenced by the object may still be stored as a primitive data type.

20. What is local class?
Ans: A local class is declared locally within a block of Java code, rather than as a member of a class. Typically, a local class is defined within a method, but it can also be defined within a static initializer or instance initializer of a class.

21. What is Constructor?
Ans: In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.It is a special type of method which is used to initialize the object.Every time an object is created using the new() keyword, at least one constructor is called.

22. What is constructor overloading?

Ans: More than one constructor with different signature in a class is called constructor overloading. Signature of constructor includes:

- Number of arguments
- Type of arguments
- Sequence of arguments

When we create an object, the compiler determines the most appropriate definition to use by comparing the signature of constructor with the instantiation of object.

23. What are the 3 ways to initialize objects?
Ans: There are many different ways to create objects in Java. **Following are some ways in which you can create objects in Java:**

1) Using new Keyword : Using new keyword is the most basic way to create an object. This is the most common way to create an object in java. Almost 99% of objects are created in this way. By using this method we can call any constructor we want to call (no argument or parameterized constructors).

2) Using New Instance : If we know the name of the class & if it has a public default constructor we can create an object –**Class.forName**. We can use it to create the Object of a Class. Class.forName actually loads the Class in Java but doesn't create any Object. To Create an Object of the Class you have to use the new Instance Method of the Class.

3) Using clone() method: Whenever clone() is called on any object, the JVM actually creates a new object and copies all content of the previous object into it. Creating an object using the clone method does not invoke any constructor.
To use clone() method on an object we need to implement **Cloneable** and define the clone() method in it.

24. What is Local variable?
Ans: A local variable in Java is a variable that's declared within the body of a method. Then you can use the variable only within that method. Other methods in the class aren't even aware that the variable exists.

25. What about instance variable?
Ans: **Instance variables** are variables defined in a class, but outside the body of methods. The declaration of an instance variable is similar to the declaration of a local variable of a method, but:

1. the variable is defined inside the class, but outside all methods;
2. the variable is preceded by an access modifier (usually private);
3. the variable is *always initialized* when the object is created, either implicitly (to a default value), or explicitly by the constructor (see later).

Note: This is different from local variables, which are not necessarily initialized when the associated memory location is created.

26. What is static variable?
Ans: Static variables are created when the program starts and destroyed when the program stops. Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.

27. How to convert ArrayList to Array and Array to ArrayList in java?
Ans: Let's see a simple example to convert ArrayList to Array and Array to ArrayList in Java:

```
public class LengthVsSizeArrayList {
  public static void main(String[] args) {
    //creating Arraylist
    List<String> fruitList = new ArrayList<>();
    //adding String Objects to fruitsList ArrayList
    fruitList.add("Mango");
    fruitList.add("Banana");
    fruitList.add("Apple");
    fruitList.add("Strawberry");
    fruitList.add("Pineapple");
```

```
        System.out.println("Converting ArrayList to Array" );
        String[] item = fruitList.toArray(new String[fruitList.size()]);
        for(String s : item){
           System.out.println(s);
        }
        System.out.println("Converting Array to ArrayList" );
        List<String>l2 = new ArrayList<>();
        l2 =  Arrays.asList(item);
        System.out.println(l2);
    }
}
```

28. Write down the difference between constructor and method?

Ans: Following are the difference between constructor and method.

- Constructor is used to initialize an object whereas method is used to exhibits functionality of an object.
- Constructors are invoked implicitly whereas methods are invoked explicitly.
- Constructor does not return any value where the method may/may not return a value.
- In case constructor is not present, a default constructor is provided by java compiler. In the case of a method, no default method is provided.
- Constructor should be of the same name as that of class. Method name should not be of the same name as that of class.

29. Why main method is static?
Ans: Java **main()** method is always static, so that compiler can call it without the creation of an object or before the creation of an object of the class.

- In any Java program, the **main()** method is the starting point from where compiler starts program execution. So, the compiler needs to call the main() method.
- If the **main()** is allowed to be non-static, then while calling the **main()** method JVM has to instantiate its class.
- While instantiating it has to call the constructor of that class, There will be ambiguity if the constructor of that class takes an argument.
- Static method of a class can be called by using the class name only without creating an object of a class.
- The **main()** method in Java must be declared **public**, **static** and **void**. If any of these are missing, the Java program will compile but a runtime error will be thrown.

30. Is main() method compulsory in Java?

Ans: To compile a program, you doesn't really need a main method in your program. But, while execution JVM searches for the main method. In the Java the main method is the entry point Whenever you execute a program in Java JVM searches for the main method and starts executing from it.

The main method must be public, static, with return type void, and a String array as argument.

```
public static int main(String[] args){
}
```

You can write a program without defining a main it gets compiled without compilation errors. But when you execute it a run time error is generated saying "Main method not found".

Example: In the following Java program, we have two methods of same name (overloading) addition and, without main method. You can compile this program without compilation errors.

```
public class Calculator {

   int addition(int a , int b){

      int result = a+b;

      return result;

   }

   int addition(int a , int b, int c){

      int result = a+b+c;

      return result;

   }

}
```

Run time error

But, when you try to execute this program following error will be generated.

```
D:\>javac Calculator.java

D:\>java Calculator
Error: Main method not found in class Calculator, please define the main
method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

To resolve this you need to define main method in this program and call the methods of the class.

```
public class Calculator {

   int addition(int a , int b){

      int result = a+b;

      return result;

   }

   int addition(int a , int b, int c){

      int result = a+b+c;

      return result;

   }

   public static void main(String args[]){

      Calculator obj = new Calculator();
```

```
        System.out.println(obj.addition(12, 13));

        System.out.println(obj.addition(12, 13, 15));

    }

}
```

Output

```
25
40
```

31. Can we declare a constructor as final in java?
Ans: A *constructor* is used to initialize an object when it is created. It is syntactically similar to a method. The difference is that the constructors have the same name as their class and, have no return type.
There is no need to invoke constructors explicitly these are automatically invoked at the time of instantiation.

Example

In the following Java program, we are trying to override a final method.

```
class SuperClass{

   final public void display() {

      System.out.println("This is a method of the superclass");

   }

}

public class SubClass extends SuperClass{

   final public void display() {

      System.out.println("This is a method of the superclass");

   }

}
```

Compile time error

On compiling, the above program generates the following error.

```
SubClass.java:10: error: display() in SubClass cannot override display() in
SuperClass
final public void display() {
                  ^
overridden method is final
1 error
```

32. What is object?

Ans: An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

An object has three characteristics:

- **State:** represents the data (value) of an object.
- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance (result) of a class.

Object Definitions:

- An object is *a real-world entity*.
- An object is *a runtime entity*.
- The object is *an entity which has state and behavior*.
- The object is *an instance of a class*.

33. What do you mean by OOP?

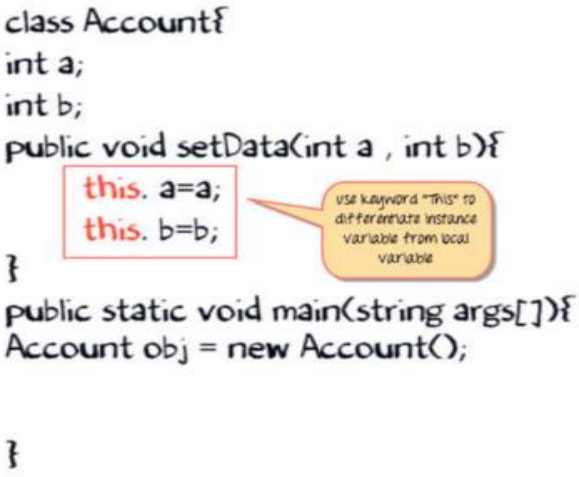
Ans: Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define the data type of a data structure, and also the types of operations (functions) that can be applied to the data structure.

In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects.

34. What is this keyword? Write it's issue.

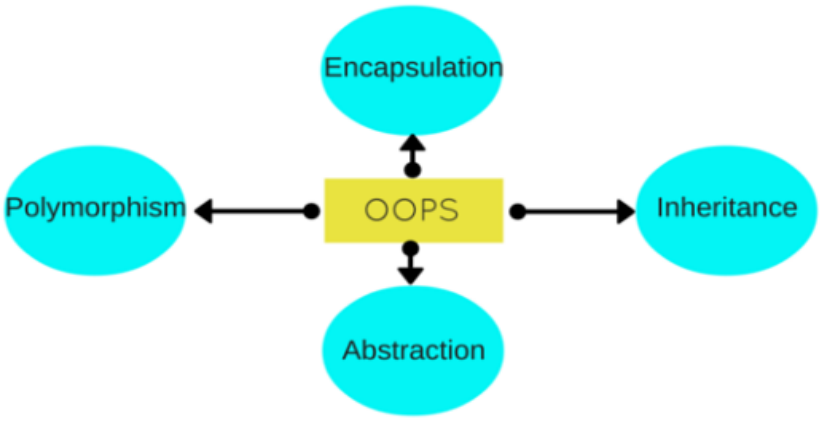
Ans: There are the following uses of **this** keyword.

- **this** can be used to refer to the current class instance variable.
- **this** can be used to invoke current class method (implicitly)
- **this()** can be used to invoke the current class constructor.
- **this** can be passed as an argument in the method call.
- **this** can be passed as an argument in the constructor call.
- **this** can be used to return the current class instance from the method.



35. Write and explain main OOP concepts.
Ans: There are four main OOP concepts in Java. These are:

- **Abstraction:** Abstraction means using simple things to represent complexity. We all know how to turn the TV on, but we don't need to know how it works in order to enjoy it. In Java, abstraction means simple things like **objects**, **classes**, and **variables** represent more complex underlying code and data. This is important because it lets avoid repeating the same work multiple times.
- **Encapsulation:** This is the practice of keeping fields within a class private, then providing access to them via public methods. It's a protective barrier that keeps the data and code safe within the class itself. This way, we can re-use objects like code components or variables without allowing open access to the data system-wide.
- **Inheritance:** This is a special feature of Object Oriented Programming in Java. It lets programmers create new classes that share some of the attributes of existing classes. This lets us build on previous work without reinventing the wheel.
- **Polymorphism:** This Java OOP concept lets programmers use the same word to mean different things in different contexts. One form of polymorphism in Java is **method overloading**. That's when different meanings are implied by the code itself. The other form is **method overriding**. That's when the different meanings are implied by the values of the supplied variables.



36.Point out characteristics of Package.

Ans: A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.
There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.
Advantage of Java Package
1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
2) Java package provides access protection.
3) Java package removes naming collision.

37. What are the advantages of OOP?
Ans: The advantages of OOP are mentioned below:

- OOP provides a clear modular structure for programs.
- It is good for defining abstract data types.
- Implementation details are hidden from other modules and other modules has a clearly defined interface.

- It is easy to maintain and modify existing code as new objects can be created with small differences to existing ones.
- objects, methods, instance, message passing, inheritance are some important properties provided by these particular languages
- encapsulation, polymorphism, abstraction are also counts in these fundamentals of programming language.
- It implements real life scenario.
- In OOP, programmer not only defines data types but also deals with operations applied for data structures.

38. What is interface?

Ans: An **interface in java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

39. Distinguish between interface between abstract class.

Ans: Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.

But there are many differences between abstract class and interface that are given below.

Abstract class	**Interface**
) Abstract class can **have abstract and non-abstract** methods.	ıterface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
) Abstract class **doesn't support multiple inheritance**.	ıterface **supports multiple inheritance**.
) Abstract class **can have final, non-final, static and non-static variables**.	ıterface has **only static and final variables**.
) Abstract class **can provide the implementation of interface**.	ıterface **can't provide the implementation of abstract class**.
) The **abstract keyword** is used to declare abstract class.	he **interface keyword** is used to declare interface.
) An **abstract class** can extend another Java class and implement multiple Java interfaces.	n **interface** can extend another Java interface only.
) An **abstract class** can be extended using keyword "extends".	n **interface** can be implemented using keyword "implements".
) A Java **abstract class** can have class members like private, protected, etc.	Iembers of a Java interface are public by default.

)**Example:**
public abstract class Shape{
public abstract void draw();
}
	xample:
	public interface Drawable{
	void draw();
	}

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

40. What are advantages of Encapsulation?

Ans: Encapsulation - The key advantage of using an *Object Oriented Programming* language like Java is that it provides your code - **security**, **flexibility** and its **easy maintainability** through *encapsulation*.

- **Encapsulation** helps us in **binding the data**(i*nstance variables*) and the **member functions**(that work on the instance variables) of a class.
- **Encapsulation** is also useful in **hiding** the **data**(*instance variables*) of a class from an *illegal direct access*.
- **Encapsulation** also helps us to make a **flexible code** which is easy to *change and maintain.*

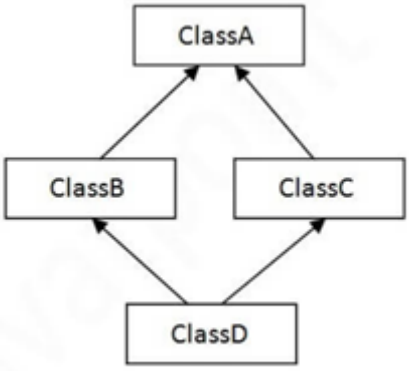
If proper encapsulation is not followed in a code, it leads to its ***bad designing***.

41. Why multiple inheritance not supported by Java Class?
Ans: To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

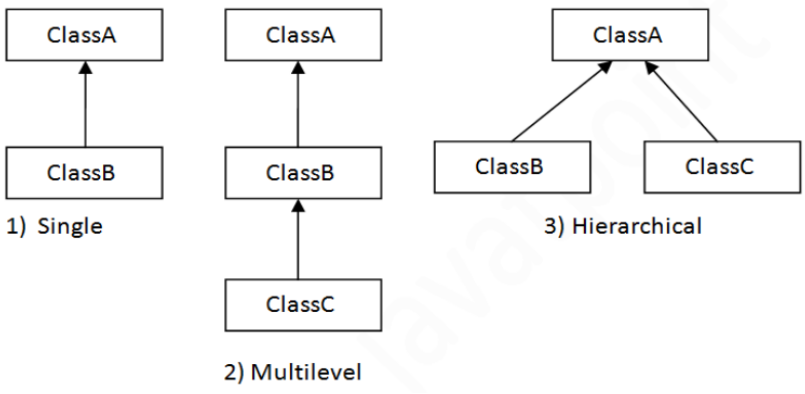
If a child inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.

The problem with multiple inheritance is that multiple parent classes may have the same method name, then at run time it becomes difficult for the compiler to decide which method to execute from the child class

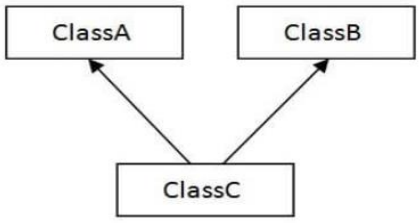
Therefore, Java does'nt support multiple inheritance for classes. This problem is commonly referred to Diamond problem.



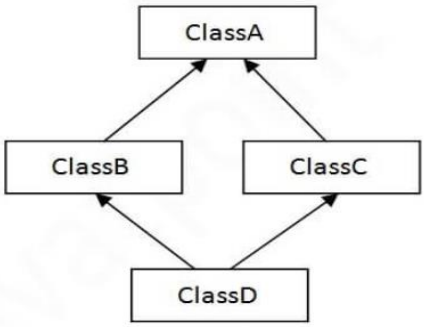
42. **What are different types of inheritance?**
On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.



When one class inherits multiple classes, it is known as multiple inheritance. For Example:



4) Multiple



5) Hybrid

43. What is static method?

Ans; The methods or variables defined as static are shared among all the objects of the class. The static is the part of the class and not of the object. The static variables are stored in the class area, and we do not need to create the object to access such variables. Therefore, static is used in the case, where we need to define variables or methods which are common to all the objects of the class.

For example, In the class simulating the collection of the students in a college, the name of the college is the common attribute to all the students. Therefore, the college name will be defined as **static**.

44. Why is main method static?

Ans: Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation.

45. Why Java doesn't support pointer?

Ans: Most studies agree that **pointers** are one of the primary features that enable developers to inject bugs into their code. When Java was created, the intention was to create a language that is **easy to learn** and not prone to the bugs that C++ is prone to. It's not like **c/c++** where we have to manage the memory management by destructors. In java **automatic Garbage Collector** works for memory management. Actually, Java references are pointers so everything in Java is accessed only through pointers.

46. What is super keyword in Java?

Ans: The **super** keyword in Java is a reference variable which is used to refer immediate parent class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

47. Can we declare constructor final?

Ans: No, because constructor is never inherited.

48. Can we declare constructor final?

Ans: In inheritance whenever you extend a class. The child class inherits all the members of the superclass except the constructors.

In other words, constructors cannot be inherited in Java, therefore, you cannot **override** constructors.

So, writing final before constructors make no sense. Therefore, java does not allow final keyword before a constructor.

If you try, make a constructor final a compile-time error will be generated saying “modifier final not allowed here”.

49. What is (.) operator?
Ans: The (.) operator is also known as member operator it is used to access the member of a package or a class.

50. Why multiple inheritances supported in interface?
Ans: Java supports multiple inheritance through interfaes only. A class can implement any number of interface but can extend only one class.

Multiple inheritance is supported by interface because there is no ambiguity as implementation is provided by the implemented class. The interface contains all abstract methods and a class can implement multiple methods just because all the abstract methods are overridden in the child class.

```
InterfaceA{
InterfaceB{

public void hi();

}

        }
```

```
InterfaceB{

public void hi();

    }
```

```
public class myClass implements InterfaceA,

    hi(){

System.out.println(“Hello”);
```

51. Abstract class vs class in java.

Ans:

ABSTRACT CLASS	CONCRETE CLASS
An abstract class is declared using abstract modifier.	A concrete class is note declared using abstract modifier.
An abstract class cannot be directly instantiated using the new keyword.	A concrete class can be directly instantiated using the new keyword.
An abstract class may or may not contain abstract methods.	A concrete class cannot contain an abstract method.
An abstract class cannot be declared as final.	A concrete class can be declared as final.
Interface implementation is not possible	Interface implementation is possible.

52. Why is Inheritance used in Java?
Ans:

- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

53. What are the main uses of the super keyword?
Ans: There are the following uses of super keyword.

- super can be used to refer to the immediate parent class instance variable.
- super can be used to invoke the immediate parent class method.
- super() can be used to invoke immediate parent class constructor.

54. Can we change the scope of the overridden method in the subclass?

Ans: Yes, we can change the scope of the overridden method in the subclass. However, we must notice that we cannot decrease the accessibility of the method. The following point must be taken care of while changing the accessibility of the method.

- The private can be changed to protected, public, or default.
- The protected can be changed to public or default.
- The default can be changed to public.
- The public will always remain public.

55. Can we declare an interface as final?

Ans: No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.

56. What is the difference between the final method and abstract method?

Ans: The main difference between the final method and abstract method is that the abstract method cannot be final as we need to override them in the subclass to give its definition.

57. Difference between Abstraction and Encapsulation.

Ans:

Abstraction	Encapsulation
1. Abstraction solves the problem in the design level.	1. Encapsulation solves the problem in the implementation level.
2. Abstraction is used for hiding the unwanted data and giving relevant data.	2. Encapsulation means hiding the code and data into a single unit to protect the data from outside world.
3. Abstraction lets you focus on what the object does instead of how it does it	3. Encapsulation means hiding the internal details or mechanics of how an object does something.
4. **Abstraction**- Outer layout, used in terms of design. For Example:- Outer Look of a Mobile Phone, like it has a display screen and keypad buttons to dial a number.	4. **Encapsulation**- Inner layout, used in terms of implementation. For Example:- Inner Implementation detail of a Mobile Phone, how keypad button and Display Screen are connect with each other using circuits.

58. How to make a read-only class in Java?

Ans: A class can be made read-only by making all of the fields private. The read-only class will have only getter methods which return the private property of the class to the main method. We cannot modify this property because there is no setter method available in the class. Consider the following example.

```
  //A Java class which has only getter methods.
public class Student{
//private data member
private String college="AKG";
//getter method for college
public String getCollege(){
return college;
   }
}
```

59. What are the conditions for a class to become abstract?

Ans:

- Abstract classes may or may not contain *abstract methods*, i.e., methods without body (public void get();)
- But, if a class has at least one abstract method, then the class **must** be declared abstract.
- If a class is declared abstract, it cannot be instantiated.
- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

60. Difference between inheritance and polymorphism.

Ans:

BASIS FOR COMPARISON	INHERITANCE	POLYMORPHISM
Basic	Inheritance is creating a new class using the properties of the already existing class.	Polymorphism is basically a common interface for multiple form.
Implementation	Inheritance is basically implemented on classes.	Polymorphism is basically implemented on function/methods.
Use	To support the concept	Allows object to decide which form

61. What is thread in Java?

Ans: A thread is a single sequence of flow of control within a program.

```
class Multi extends Thread{
        public void run(){
        System.out.println("thread is running...");
                }

public static void main(String args[]){
        Multi t1=new Multi();
                t1.start();
                 }
                }
```

62. What is multithreading?

Ans: Multithreading is a type of execution model that allows multiple threads to exist within the context of a process such that they execute independently but share their process resources.

63. Write down the difference between thread and process?

Ans: The major differences between a process and a thread are given as follows:

Comparison Basis	Process	Thread
Definition	A process is a program under execution i.e an active program.	A thread is a lightweight process that can be managed independently by a scheduler.
Context switching time	Processes require more time for context switching as they are more heavy.	Threads require less time for context switching as they are lighter than processes.
Memory Sharing	Processes are totally independent and don't share memory.	A thread may share some memory with its peer threads.
Communication	Communication between processes requires more time than between threads.	Communication between threads requires less time than between processes .
Blocked	If a process gets blocked, remaining processes can continue execution.	If a user level thread gets blocked, all of its peer threads also get blocked.
Resource Consumption	Processes require more resources than threads.	Threads generally need less resources than processes.
Dependency	Individual processes are independent of each other.	Threads are parts of a process and so are dependent.
Data and Code sharing	Processes have independent data and code segments.	A thread shares the data segment, code segment, files etc. with its peer threads.
Treatment by OS	All the different processes are treated separately by the operating system.	All user level peer threads are treated as a single task by the operating system.
Time for creation	Processes require more time for creation.	Threads require less time for creation.
Time for termination	Processes require more time for termination.	Threads require less time for termination.

64. What are the common advantage and disadvantage of multithreading?

Ans: The primary function of multithreading is to simultaneously run or execute multiple tasks. These tasks are represented as threads in a Java program and have a separate execution path. Also, handling of multithreaded Java programs is easy because you can decide the sequence in which execution of Java threads take place.

Following are some of the common advantages of multithreading:

- Enhanced performance by decreased development time
- Simplified and streamlined program coding
- Improvised GUI responsiveness
- Simultaneous and parallelized occurrence of tasks
- Better use of cache storage by utilization of resources
- Decreased cost of maintenance
- Better use of CPU resource

Multithreading does not only provide you with benefits, it has its disadvantages too. Let us go through some common disadvantages:

- Complex debugging and testing processes
- Overhead switching of context
- Increased potential for deadlock occurrence
- Increased difficulty level in writing a program
- Unpredictable results

65. How to create thread in java?

Ans: There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

Thread class provide constructors and methods to create and perform operations on a thread.Thread class extends Object class and implements Runnable interface.

Java Thread Example by extending Thread class

```
class Multi extends Thread{
    public void run(){
    System.out.println("thread is running...");
        }
 public static void main(String args[]){
    Multi t1=new Multi();
   t1.start();
     }
    }
Output:thread is running...
```

Java Thread Example by implementing Runnable interface

```
class Multi3 implements Runnable{
        public void run(){
        System.out.println("thread is running...");
       }
```

```
    public static void main(String args[]){
        Multi3 m1=new Multi3();
        Thread t1 =new Thread(m1);
        t1.start();
            }
           }
Output:thread is running...
```

If you are not extending the Thread class,your class object would not be treated as a thread object.So you n explicitely create Thread class object.We are passing the object of your class that implements Runnable so t run() method may execute.

67. Describe java thread model.

Ans: The Java language and its run-time system was designed keeping in mind about multithreading. The run-time system depend upon multithreading. Java provides asynchronous thread environment, this helps to increase the utilization of CPU.

Multithreading is best in all cases in contrast with single-thread model. Single-thread system uses an approach of event loop with polling. According to this approach a single thread in the system runs in an infinite loop. Polling the mechanism, that selects a single event from the event queue to choose what to do next. As the event is selected, then event loop forwards the control to the corresponding required event handler. Nothing else can be happened, until the event handler returns. Because of this CPU time is wasted. Here, only one part of the complete program is dominating the whole system, and preventing the system to execute or start any other process. In single-thread model one thread blocks all other threads until its execution completes. On other waiting or idle thread can start and acquire the resource which is not in use by the current thread. This causes the wastage of resources.

Java's multithreading provides benefit in this area by eliminating the loop and polling mechanism, one thread can be paused without stopping the other parts of the program. If any thread is paused or blocked, still other threads continue to run.

As the process has several states, similarly a thread exists in several states. A thread can be in the following states:

Ready to run (New): First time as soon as it gets CPU time.

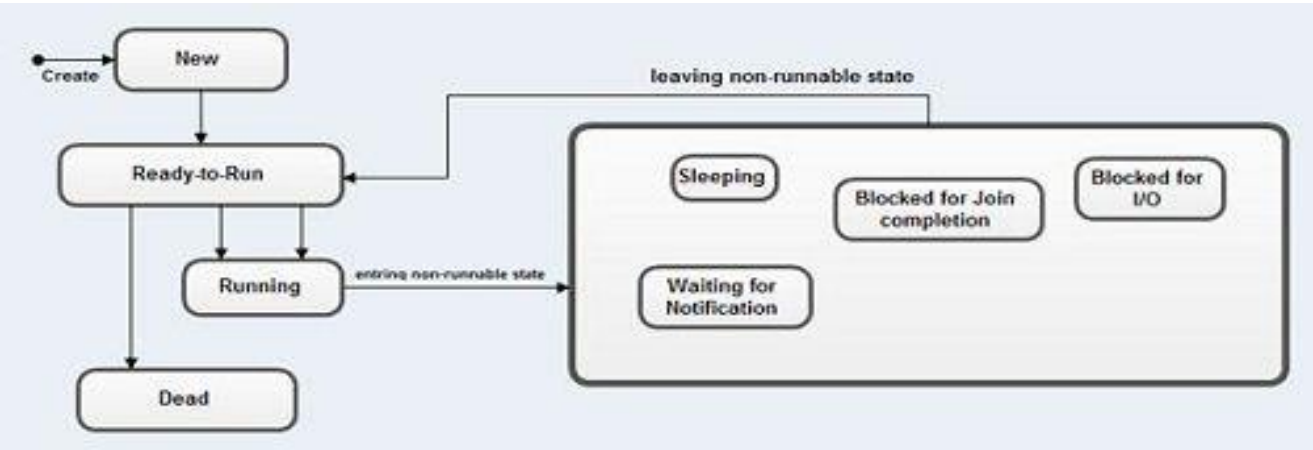
Running: Under execution.

Suspended: Temporarily not active or under execution.

Blocked: Waiting for resources.

Resumed: Suspended thread resumed, and start from where it left off.

Terminated: Halts the execution immediately and never resumes.



68. What is thread synchronization?

Ans: Java supports an asynchronous multithreading, any number of thread can run simultaneously without disturbing other to access individual resources at different instant of time or shareable resources. But some time it may be possible that shareable resources are used by at least two threads or more than two threads, one has to write at the same time, or one has to write and other thread is in the middle of reading it. For such type of situations and circumstances Java implements synchronization model called *monitor*. The monitor was first defined by C.A.R. Hoare. You can consider the monitor as a box, in which only one thread can reside. As a thread enter in monitor, all other threads have to wait until that thread exits from the monitor. In such a way, a monitor protects the shareable resources used by it being manipulated by other waiting threads at the same instant of time. Java provides a simple methodology to implement synchronization.

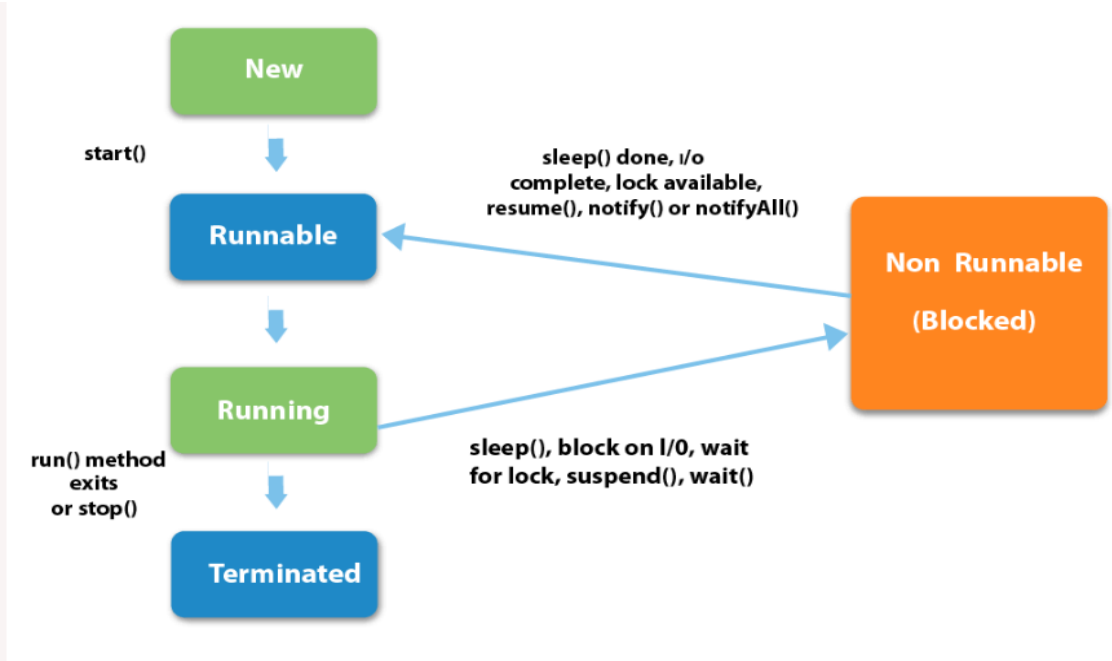
69. Describe life cycle of java thread model.

Ans: A thread can be in one of the five states. According to sun, there is only 4 states in **thread life cycle in java** new, runnable, non-runnable and terminated. There is no running state.

But for better understanding the threads, we are explaining it in the 5 states.

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable
3. Running
4. Non-Runnable (Blocked)
5. Terminated



1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable
The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running
The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)
This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated
A thread is in terminated or dead state when its run() method exits.

70. Write down about delegation event model.

Ans: The Change in state of an object is called as an event i.e. the event describes a particular change in the state of a source.

The Events are generated as a result of the user interaction with Graphical User Interface (GUI) components.

For example- Clicking on a button, movement of the mouse, inserting a character through the keyboard, selection of an item from the given list, scrolling of the page are all of the activities which causes an event to be happen.

Types of Event

The events can be generally classified into following two categories:

1) The Foreground Events:-

- The event which requires the direct interaction of the user are called as the Foreground events.
- They are generated as a consequence of a user interacting with the graphical components in GUI. For example- Clicking on a button, movement of the mouse, inserting a character through the keyboard, selection of an item from the given list, scrolling of the page.

- **The Background Events** :-

- The event that requires the interaction of an end user is called as the Background events.
- For Example:- an Operating system interrupts, failure of hardware or software, a timer expires, an operation completion etc. are the examples of Background events.

Now **Event Handling** is the mechanism that has control over the event and decides what should be happen if an event occurs.

This mechanism has the separate code which is known as event handler which is executed when an event happens.

The Java uses a Delegation Event Model to handle all the events which defines the standard mechanism to generate & handle the events.

Let’s have a look on the brief introduction to this model.

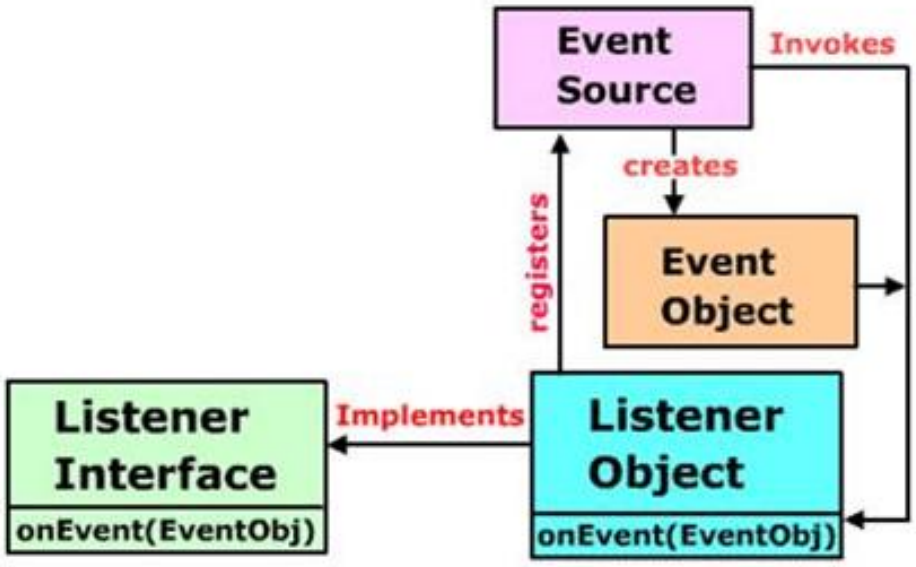
The **Delegation Event Model** has following key participants as:

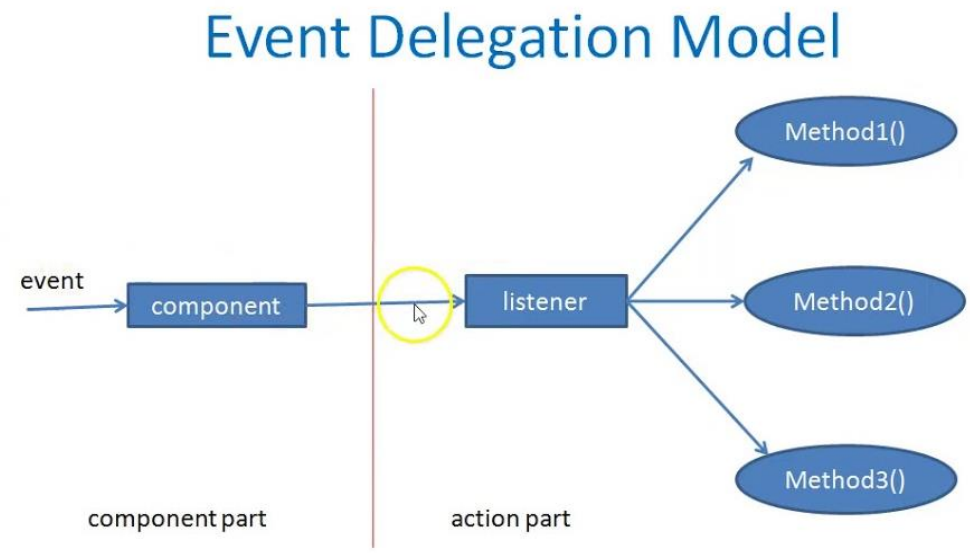
- **Source** – The source is an object on which an event occurs and the Source is responsible for providing information of the event which has occurred to it’s handler. Java provide it as with classes for the source object.

- **Listener** – It is also called as event handler. The Listener is responsible for generating a response to an event. From the java implementation point of view the listener is an object too. The Listener waits until it has received an event and Once the event is received , the listener process the event & then returns the result.

Following are the Steps to Handle Events

- To Declare and instantiate the event sources (or components) as buttons, menus, choices etc.
- To Implement an interface (listener) for providing the event handler that responds to event source activity.
- To Register this event handler with event source.
- To Add the event source with the container like frame, panel etc.





71. Write the difference between runnable and callable.

Ans: Difference Between Callable And Runnable

Runnable	**Callable**
Available in java.lang package.	Available in java.util.concurrent package.
Method: void run()	Method: V call() throws Exception
It cannot return any value.	It can return value.
It cannot throw checked exception.	It can throw checked exception.

class TestThread implements Runnable {
@override
public void run(){
//do something
}
}
	class TestThread implements Callable<String> {
	@override
	public String call() throws Exception {
	//do something
	return “codesjava.com”;
	}
	}

72. Discuss about exception handling.

Ans: The Java programming language provides a mechanism known as *exceptions* to help programs report and handle errors. When an error occurs, the program throws an exception.

Three statements play a part in handling exceptions:

- The *try* statement identifies a block of statements within which an exception might be thrown.
- The *catch* statement must be associated with a try statement and identifies a block of statements that can handle a particular type of exception. The statements are executed if an exception of a particular type occurs within the try block.
- The *finally* statement must be associated with a try statement and identifies a block of statements that are executed regardless of whether or not an error occurs within the try block.

Here's the general form of these statements:

```
try {
   statement(s)
} catch (exceptiontype name) {
   statement(s)
} finally {
   statement(s)
}
```

73. write down the advantage and disadvantage of inheritance ?

Ans:

Advantages of Inheritance in OOPS

1. The main advantage of the inheritance is that it helps in reusability of the code. The codes are defined only once and can be used multiple times. In java we define the super class or base class in which we define the functionalities and any number of child classes can use the functionalities at any time.
2. Through inheritance a lot of time and efforts are being saved.
3. It improves the program structure which can be readable.
4. The program structure is short and concise which is more reliable.
5. The codes are easy to debug. Inheritance allows the program to capture the bugs easily
6. Inheritance makes the application code more flexible to change.
7. Inheritance results in better organisation of codes into smaller, simpler and simpler compilation units.

Disadvantages of Inheritance in OOPS

1. The main disadvantage of the inheritance is that the two classes(base class and super class) are tightly coupled that is the classes are dependent on each other.
2. If the functionality of the base class is changed then the changes have to be done on the child classes also.
3. If the methods in the super class are deleted then it is very difficult to maintain the functionality of the child class which has implemented the super class's method.
4. It increases the time and efforts take to jump through different levels of the inheritance.

Prepared by-

Fahim Ahammed Firoz
Part-1/2
Dept. of CSE
Imperial College of Engineering

Teresa Jency Bala
Part-1/2
Dept. of CSE
Imperial College of Engineering