UNIT- III

Interfaces, Packages, and Multithreaded Programming: Interfaces: Multiple Inheritance: Introduction, Defining Interfaces, Extending Interfaces, Implementing Interfaces, Accessing Interface Variables. Packages: Putting Classes together: Introduction, Java API Packages, Using System Packages, Naming Conventions, Creating Packages, Accessing a Package, Using a Package, Adding a Class to a Package, Hiding Classes. Multithreaded Programming: Introduction, Creating Threads, Extending the Thread Class, Stopping and Blocking a thread, Life Cycle of a thread, Using Thread Methods, Thread Exceptions, Thread Priority, Synchronization, Implementing the 'Runnable' Interface. [12 Hours]

1. **What is interface? Explain interface with an example?**

Ans:

- **An interface is a keyword in java.**
- **An interface is pure abstract class in java**
- **For all interfaces we can't create object, because they are abstract**
- 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.

How to declare an interface?

Interface in Java

1. Interface
2. Example of Interface
3. Multiple inheritance by Interface
4. Why multiple inheritance is supported in Interface while it is not supported in case of class.
5. Marker Interface
6. Nested Interface

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.

Java Interface also **represents the IS-A relationship**.

It cannot be instantiated just like the abstract class.

Since Java 8, we can have **default and static methods** in an interface.

Since Java 9, we can have **private methods** in an interface.

Why use Java interface?

There are mainly three reasons to use interface. They are given below.

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction;

Syntax:

```
interface <interface_name>
{

   // declare constant fields
   // declare methods that abstract
   // by default.
}
```

Java Interface Example

```
interface A
{
void print();
}
Public class Today  implements A
{
public void print()
{
System.out.println("Hello");
}

public static void main(String args[]){
Today obj = new Today();
obj.print();
 }
}
```

2. What is an interface? Explain multiple inheritances using interface? VVVVIMP

Ans:

- **An interface is a keyword in java.**

- **An interface is pure abstract class in java**
- **For all interfaces we can't create object, because they are abstract**
- 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.

Syntax:

```
interface <interface_name>
{

   // declare constant fields
   // declare methods that abstract
   // by default.
}
```

Multiple Inheritances:

Multiple Inheritance in Java

```
interface A
{
void print();
}
interface Showable extends A
```

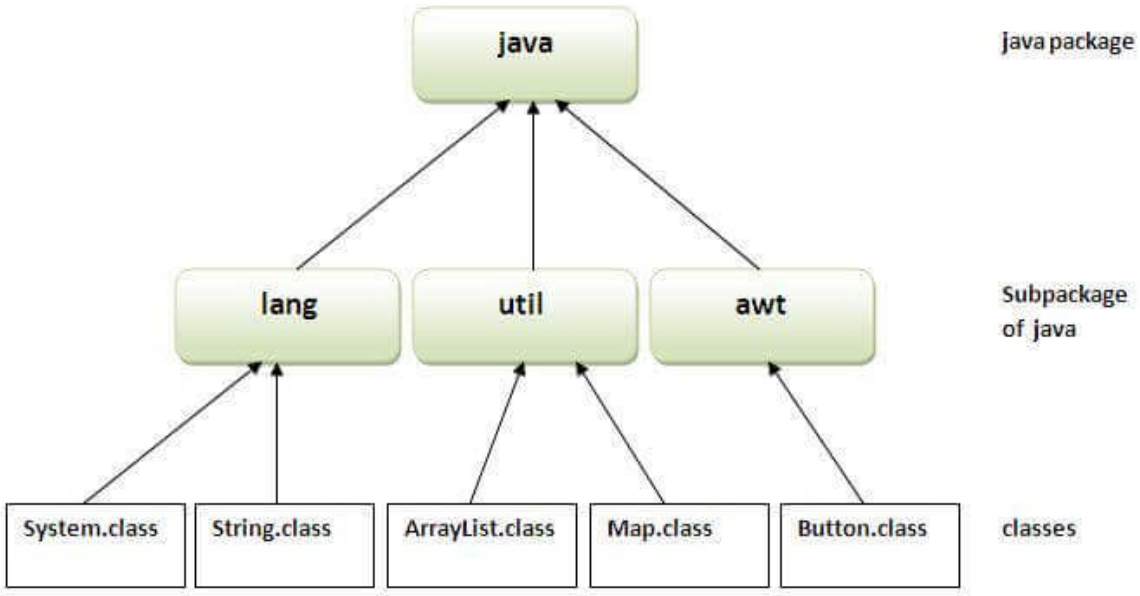
```
{
void show();
}
Public class Today implements Printable, Showable
{
public void print()
{
System.out.println("Hello");
}
public void show()
{
System.out.println("Welcome");
}
  public static void main(String args[])
{
Today obj = new Today();
obj.print();
obj.show();
 }
}
```

3. **What is package? Explain packages in detail? VVVVIMP**

   Ans: A package is a collection of classes & interfaces.

   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.



Example:

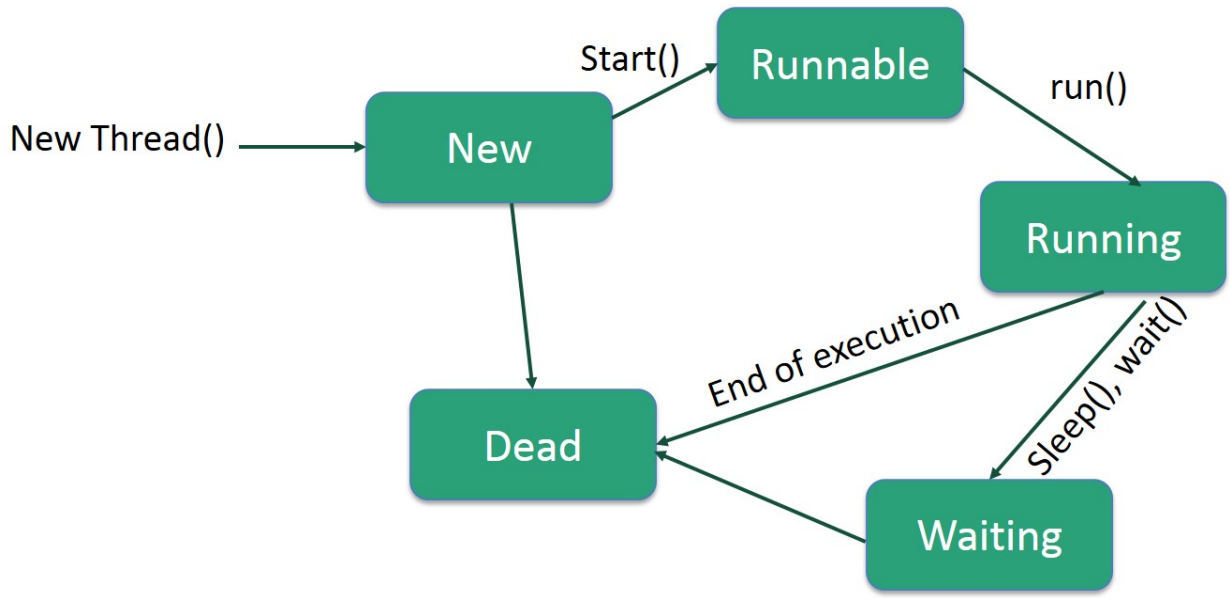
```
package mypack;
public class Simple
{
 public static void main(String args[])
{
   System.out.println("Welcome to package");
  }
}
```

4. **What is a thread? Explain thread life cycle with a neat diagram? VVVVVIMP**

   Ans:

   Thread: A **thread** is a single sequential flow of control within a program.

   Thread Life cycle:



- **New** – A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a **born thread**.
- **Runnable** – After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.
- **Waiting** – Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
- **Timed Waiting** – A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.
- **Terminated (Dead)** – A runnable thread enters the terminated state when it completes its task or otherwise terminates.

5. What is a thread? Explain different ways of creating threads in java? VVVVIMP

Ans:

Thread: A **thread** is a single sequential flow of control within a program.

Different ways of creating a threads in java.

In java threads can create in two ways they are:

A. Single Thread
B. Multi Thread

Single Thread:- single thread is a thread, which contains only one parent thread and followed by only one child thread.

Single threads can be create by using extends keyword.

Syntax:

```
class A extends Thread
{
  public void run()
{
  for(int i=0;i<=10;i++)
{
  System.out.println(i);
}
}
public class Today
{
  public static void main(String[] args)
{
 A a1=new A();
    a1.start();
}
}
```

6. What is thread? Explain multi thread with an example? VVVVVIMP
   Or
   Explain Multi Thread with an example?

   Ans: Thread: A **thread** is a single sequential flow of control within a program.

   Multi Thread:

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.

Multi threads can be created by using Runnable interface.

```
// Java code for thread creation by implementing
// the Runnable Interface
public class MultithreadingDemo implements Runnable
{
	public void run()
	{
		try
		{
			// Displaying the thread that is running
			System.out.println ("Thread " +

	Thread.currentThread().getId() + " is running");

		}
		catch (Exception e)
		{
			// Throwing an exception
			System.out.println ("Exception is caught");
		}
	}
}

// Main Class
public class Multithread
{
	public static void main(String[] args)
	{
```

```
                int n = 8; // Number of threads
                for (int i=0; i<n; i++)
                {
                        Thread object = new Thread(new MultithreadingDemo());
                        object.start();
                }
        }
}
```

7. **Explain thread properties? IMP FOR 3 MARKS**

   Ans:

   Each and every thread will have three properties they are:

   1. **Thread name**
   2. **Thread Id**
   3. **Thread priority**

- **Thread Name:- Each and every thread will have a thread name.**
- **Thread name can be editable.**
- **To know the name of a thread we will use getName()**
- **To change name of a thread we will use setName()**

Syntax:

```
Thread t1=new Thread();
System.out.println(t1.getName());
```

Example:

```
Public class A extends Thread
{
 Public static void main(String[] args)
{
  A a1=new A();
System.out.println(a1.getName());
//to change name of a thread
```

a1.setName("Hello");

System.out.println(a1.getName());

}

}

Thread Id: - Each and every thread will have a thread Id.

- **Thread Id can't be editable.**
- **To know the Id of a thread we will use getName()**

Syntax:

- **Thread t1=new Thread();**
- **System.out.println(t1.getId());**

Example:

Public class A extends Thread

{

Public static void main(String[] args)

{

A a1=new A();

System.out.println(a1.getId());

}

}

Thread priority:-

- **Each and every thread will have a thread priority.**
- **Thread priority can be editable.**
- **To know the priority of a thread we will use getPriority() method**
- **To change priority of a thread we will use setPriority() method. Thread priority should be set in the range b/w 1 to 10**

Each & every thread will have three priorities they are:

1. **MIN_PRIORITY=1**
2. **NORM_PRIORITY=5**
3. **MAX_PRIORITY=10**

Syntax:

- **Thread t1=new Thread();**
- **System.out.println(t1.getPriority());**

Example:

```
Public class A extends Thread
{
 Public static void main(String[] args)
{
  A a1=new A();
System.out.println(a1.getPriority());
System.out.println(a1.MIN_PRIORITY());
System.out.println(a1.NORM_PRIORITY());
System.out.println(a1.MAX_PRIORITY());
// to set the thread priority
 a1.setPriority(6);
System.out.println(a1.getPriority());

}
}
```