NAME-PRIYANSH SALIAN

BATCH-C31

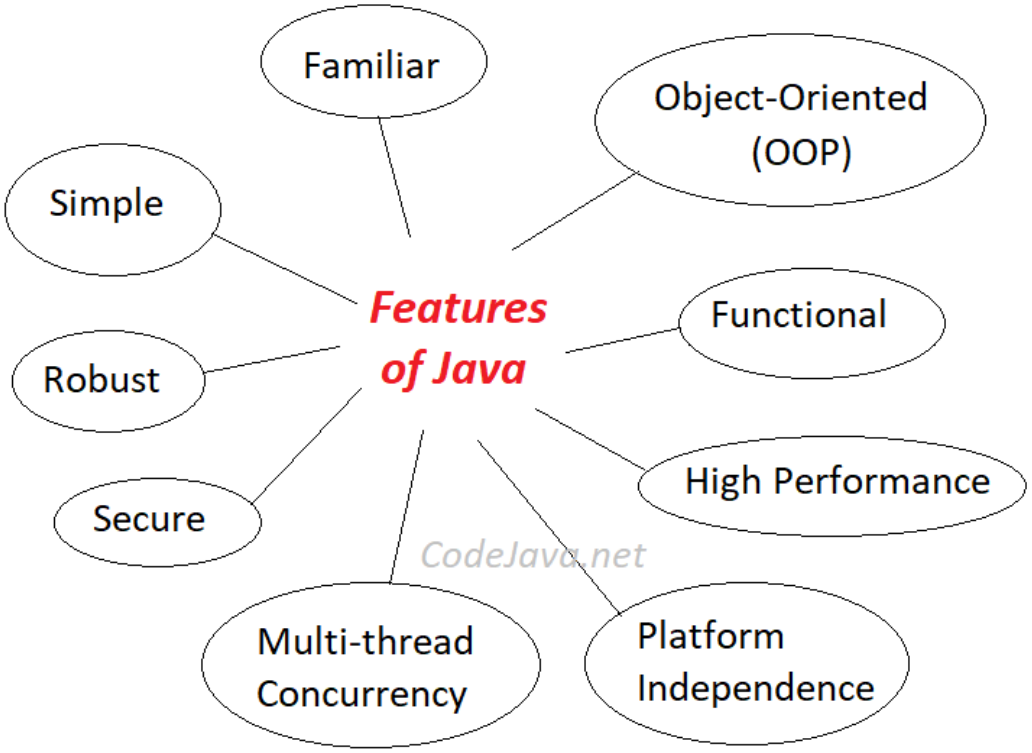
ROLL NO-2003148

ASSIGNMENT 1

Write Short Notes on

A.Features of Java

Generally, Java is a simple, robust and secure programming language. Here are the most important features of Java:



1. Java is Simple:

The Java programming language is easy to learn. Java code is easy to read and write.

2. Java is Familiar:

Java is similar to C/C++ but it removes the drawbacks and complexities of C/C++ like pointers and multiple inheritances. So if you have background in C/C++, you will find Java familiar and easy to learn.

3. Java is an Object-Oriented programming language:

Unlike C++ which is semi object-oriented, Java is a fully object-oriented programming language. It has all OOP features such as abstraction, encapsulation, inheritance and polymorphism.

4. Java supports Functional programming:

Since Java SE version 8 (JDK 8), Java is updated with functional programming feature like functional interfaces and Lambda Expressions. This increases the flexibility of Java.

5. Java is Robust:

With automatic garbage collection and simple memory management model (no pointers like C/C++), plus language features like generics, try-with-resources,… Java guides programmer toward reliable programming habits for creating highly reliable applications.

6. Java is Secure:

The Java platform is designed with security features built into the language and runtime system such as static type-checking at compile time and runtime checking (security manager), which let you creating applications that can't be invaded from outside. You never hear about viruses attacking Java applications.

7. Java is High Performance:

Java code is compiled into bytecode which is highly optimized by the Java compiler, so that the Java virtual machine (JVM) can execute Java applications at full speed. In addition, compute-intensive code can be re-written in native code and interfaced with Java platform via *Java Native Interface* (JNI) thus improve the performance.

8. Java is Multithreaded:

The Java platform is designed with multithreading capabilities built into the language. That means you can build applications with many concurrent threads of activity, resulting in highly interactive and responsive applications.

9. Java is Platform Independence:

A. Java code is compiled into intermediate format (bytecode), which can be executed on any systems for which Java virtual machine is ported. That means you can write a Java program once and run it on Windows, Mac, Linux or Solaris without re-compiling. Thus the slogan "*Write once, run anywhere*" of Java.

B.JVM

A Java virtual machine (JVM), an implementation of the Java Virtual Machine Specification, interprets compiled Java binary code (called bytecode) for a computer's processor (or "hardware platform") so that it can perform a Java program's instructions. Java was designed to allow application programs to be built that could be run on any platform without having to be rewritten or recompiled by the programmer for each separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform.

The Java Virtual Machine Specification defines an abstract -- rather than a real -- machine or processor. The Specification specifies an instruction set, a set of registers, a stack, a "garbage heap" and a method area. Once a Java virtual machine has been implemented for a given platform, any Java program (which, after compilation, is called bytecode) can run on that platform. A Java virtual machine can either interpret the bytecode one instruction at a time (mapping it to a real processor instruction) or the bytecode can be compiled further for the real processor using what is called a just-in-time compiler.

C.Wrapper Classes

Wrapper classes, as the name suggests, wraps around or encapsulates primitive datatypes in Java. We talked about this in one of our previous articles so be sure to check them out too.

Wrapper classes, simply put, is basically a class for converting a primitive datatype, to an object for specific functions. This is useful because primitive datatypes are generally immutable. Also, in Java, everything is object-oriented in nature. So if you want to have a deeper understanding of how this work, read along.

Some of the main reasons why wrapper classes are essential in programming are as follows:

1. When we are working with methods, it is essential that the arguments we pass to them are objects and not primitive values. Hence wrapper classes help by converting the primitive datatype into its specific Wrapper class.
2. All classes in java.util package work with objects. Hence if we need to use these classes we need to convert primitive data types to Wrapper class objects.
3. All collections, such as ArrayLists and Queues work with Objects as input. It is easy if the objects are of a user-defined datatype. But if they are primitive datatypes they cannot directly be stored in the collection. Hence we need to box them into objects and then use them accordingly.

4. Objects are essential for supporting synchronization in multithreading.
5. If we want to implement serialization in Java, we can only do so with the help of objects. That is why primitive data types need to be converted to objects.

D. Thread Life Cycle:

A thread life cycle is always in one of these five states. It can move from one state to another state. In Java, the life cycle of thread has five states.

1. Newborn State
2. Runnable State
3. Running State
4. Blocked State
5. Dead State

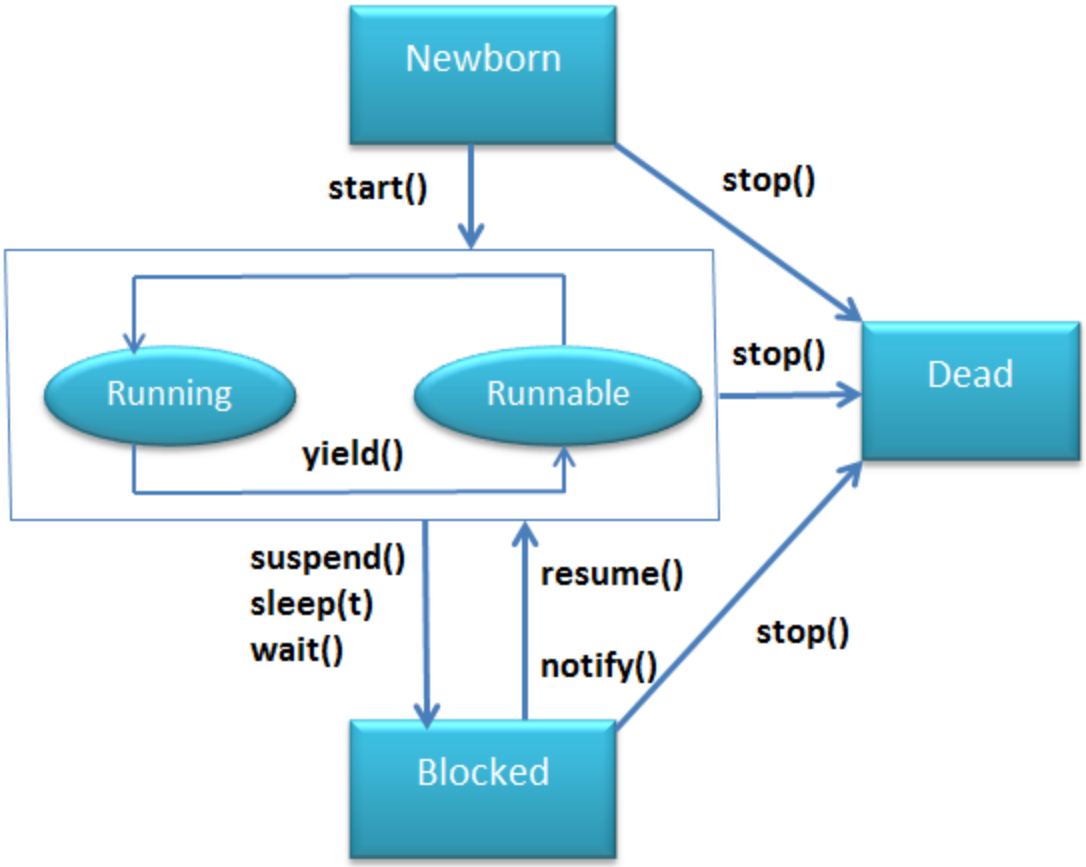


Fig: Life Cycle of Thread

Life Cycle of Thread in Java:

Newborn State:

When we want to create a thread object, the thread is born then it is said to be Newborn State. At this State, we can do by using **start()** method.

Runnable State:

It means that the thread is ready for execution and it is waiting for the availability of the processor. If all threads have equal priority, then they are given time slots for execution FCFS manner (**First Come First Serve**).

Then the thread that relinquishes control joins the queue at the end and again waits for its turn. This process of assigning time to threads that are known as **Time-Slicing.** At this State, we can do by using **yield()** method.

Runninsg State:

It means that the processor has given its time to the thread for its execution. The thread runs until it relinquishes control on its own. When a running thread may relinquish its control, then the following situations occur:

- i. It is suspend using **suspend**() method. A suspended thread can be revived by using **resume**() method.
-
- ii. It has been made to sleep. So, we can put to sleep a thread for a specified time period by using the **sleep**(time) method, where time is in milliseconds. It means that the thread is out of the queue during this time period.
-
- iii. It has been told to wait until some event occurs. We can do by using **wait**() method. Then the thread can schedule to run again using **notify**() method.

Blocked State:

A thread is said to be blocked when it is prevented from entering into the runnable state and subsequently the running state. This happens when the thread is suspending, sleeping or waiting to satisfy certain requirements. A blocked thread is "**not runnable**" but not dead.

Dead State:

A running thread ends its life when it has completed executing its **run()** method. It is a natural death. We can kill the thread by using **stop()** method.