TED (10)-3069 Reg. No. ………………………….

(REVISION-2010) Signature ……..……………………

FORTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/ TECHNOLIGY- MARCH, 2014

OOP THROUGH JAVA

(Common to CT, CM and IF)

[*Time:* 3 hours

(Maximum marks: 100)

Marks

PART –A

(Maximum marks: 10)

I. Answer all questions in a sentence

1. Write the way in which an object is created from a class

   Objects in Java are created using the keyword 'new'

   Syntax: ClassName objectName = new ClassName();

   Eg: MyClass m = new MyClass();

2. What is the use of constructor

   Constructor in java is a special type of method that is used to initialize the object. Java constructor is invoked at the time of object creation. It constructs the values i.e. provides data for the object that is why it is known as constructor.

3. List the visibility modifiers in java

   i. Private
   ii. Public
   iii. Default
   iv. protected

4. Write any two methods by which a thread may be blocked

www.madinpoly.com

i. sleep()
ii. suspend()

5. What is the use of streams in java?

   A java program uses a stream to either read data item from a source or to write data item to a destination. A stream is a path along which data flows.

PART – B

II. Answer *any five* questions. Each question carries 6 marks

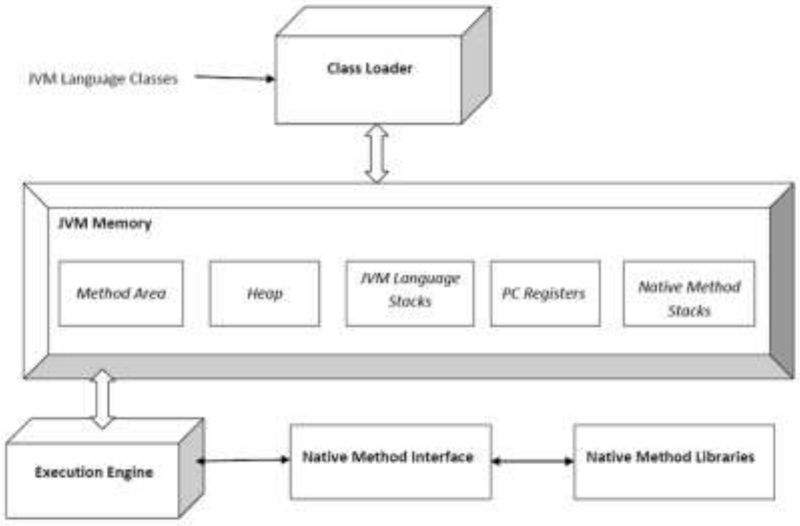
1. Compare POP and OOP

	Procedure Oriented Programming	**Object Oriented Programming**
Divided Into	In POP, program is divided into small parts calledfunctions.	In OOP, program is divided into parts called objects.
Importance	In POP,Importance is not given to data but to functions as well as sequence of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a real world.
Approach	POP follows Top Down approach.	OOP follows Bottom Up approach.
Access Specifiers	POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
Data Moving	In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
Expansion	To add new data and function in POP is not so easy.	OOP provides an easy way to add new data and function.
Data Access	In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system.	In OOP, data can not move easily from function to function,it can be kept public or private so we can control the access of data.

www.madinpoly.com

Data Hiding	POP does not have any proper way for hiding data so it is less secure.	OOP provides Data Hiding so provides more security.
Overloading	In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
Examples	Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET.

2. Describe JVM

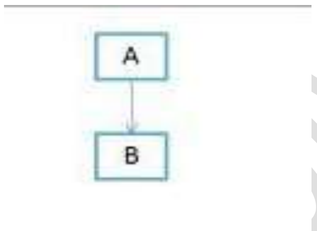


A Java virtual machine (JVM) is an abstract computing machine. There are three notions of the JVM: specification, implementation, and instance. The specification is a book that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A JVM implementation is a computer program that implements requirements of the JVM specification in a compliant and preferably per formant manner. An instance of the JVM is a process that executes a computer program compiled into Java byte code. JVM -- a machine within a machine -- mimics a real Java processor, enabling Java byte code to be executed as actions or operating system calls on any processor regardless of the operating system. For example, establishing a socket connection from a workstation to a remote machine involves an operating system call. Since different operating systems handle sockets in different ways, the JVM translates the programming code so that the two machines that may be on different platforms are able to connect.

3. Describe single inheritance in java with the help of example

Single Inheritance

Single inheritance is damn easy to understand. When a class extends another one class only then we call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a **parent class** of B and B would be a **child class** of A.



(a) Single Inheritance

```
Class A
{
   public void methodA()
   {
     System.out.println("Base class method");
   }
}
Class B extends A
{
   public void methodB()
   {
     System.out.println("Child class method");
   }
   public static void main(String args[])
   {
     B obj = new B();
```

www.madinpoly.com

```
        obj.methodA(); //calling super class method
        obj.methodB(); //calling local method
    }
  }
```

4. Describe finalizer method

   Java supports a concept called finalization, which is just opposite to initialization. Sometimes an object will need to perform some action when it is destroyed. For example, if an object is holding some non-java resource such as a file handle or window character font, then you might want to make sure these resources are freed before an object is destroyed. To handle such situations, java provides a mechanism called finalization. By using finalization, you can define specific actions that will occur when an object is just about to be reclaimed by the garbage collector. To add a finalizer to a class, you simply define the **finalize()** method. The java run time calls that method whenever it is about to recycle an object of that class. Inside the **finalize()** method you will specify those actions that must be performed before an object is destroyed. The garbage collector runs periodically, checking for objects that are no longer referenced by any running state or indirectly through other referenced objects. Right before an asset is freed, the java run time calls the **finalize()**method on the object.

   The **finalize()** method has this general form:

```
    protected void finalize()
   {
      // finalization code here
   }
```

5. Illustrate the creation of a thread by extending the threading the thread class

   It includes the following steps:

   1) Declare the class the as extending the Thread class

```
    class A extends Thread
    {
         …………………
    }
```

   2) Implement the run() method

```
public void run()
{
      …………………
}
```

3) Create a thread object and call the start() method

```
A a1=new A();
a1.start();
```

6. Describe the benefits of organizing classes to packages
   - The classes contained in the package of other programs can be easily re-used
   - Two classes in two different packages can have the same name
   - Packages provide a way to hide classes
   - Packages provide a way for separating design from coding
   - It shows that the classes and interfaces in the package are related.
   - You know where to find the classes you want if they're in a specific package.
   - The names of your classes and interfaces won't be in conflict with those of other programmers.
   - You can restrict the access to your classes.
7. Describe exception handling

The **exception handling** in java is one of the powerful mechanism to handle the runtime *errors* so that normal flow of the application can be maintained. In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime. Exception Handling is a mechanism to handle runtime errors such as ClassNotFound, IO, SQL, Remote etc. The core advantage of exception handling is to maintain the normal flow of the application. Exception normally disrupts the normal flow of the application that is why we use exception handling

Advantages of Exception Handling

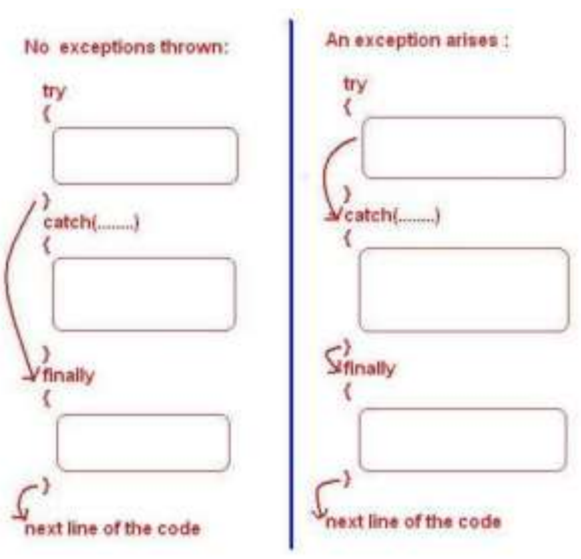
- Exception handling allows us to control the normal flow of the program by using exception handling in program.

www.madinpoly.com

- It throws an exception whenever a calling method encounters an error providing that the calling method takes care of that error.
- It also gives us the scope of organizing and differentiating between different error types using a separate block of codes. This is done with the help of try-catch blocks.

Tasks incorporated with exception handling

1) Find the problem (Hit the exception)
2) Inform that an error has occurred(Throw the exception)
3) Receive the error information(Catch the exception)
4) Take corrective actions(Handle the exception)



(5 x 6 = 30)

PART – C

(Answer *one* full question from each unit. Each question carries 15 marks)

UNIT – I

III.

a) Distinguish between class and object with the help of example

1. Class is a group of a object that shares common property.

2. Class is a collection of methods and functionality.
3. Object is real time entity of class or it is a instance of class.
4. A Class is a blueprint of an object or we can say that its a template of an object.
5. Class –Classes defines object, Object -Object can't define classes.
6. Class -Object properties can access by class instance on which that object is residing.
7. Object-Should be belonging to any of the class.
8. Object - Can created and destroyed as your necessity
9. Class-One class can define any none of Object Class is temple for an object. Object is an instance of a class.
10. Class cannot be passed as a parameter or arguments, but object can be passed as argument or parameter.

b) Explain constructor overloading with example

7

Constructor overloading in java allows having *more than one constructor inside one Class*. in last article we have discussed about method overloading and overriding and constructor overloading is not much different than method overloading. Just like in case of method overloading you have multiple methods with same name but different signature; in Constructor overloading you have *multiple constructors with different signature* with only difference that Constructor doesn't have return type in Java. Those constructors will be called as overloaded constructor. Overloading is also another form of polymorphism in Java which allows having multiple constructors with different name in one Class in java.

```
class Example
{
        inta,b;
        Example()
        {
                a = 10; b = 20;
        }
        Example(int x)
        {
```

www.madinpoly.com

```
                a = x; b = x;
        }
        Example(intx,int y)
        {
                a = x; b = y;
        }
        void show()
        {
                System.out.println(a);
                System.out.println(b);

        }
}
class Test
{
        public static void main(String args[])
        {
                Example e1=new Example();
                e1.show();
                Example e2=new Example(50);
                e2.show();
                Example e3=new Example(100,30);
                e3.show();
        }
 }
```

OR

IV.

a) Explain terms:- Dynamic binding and Message passing

8

Message Passing is nothing but sending and receiving of information by the objects same as people exchange information. So this helps in building systems that simulate real life. Following are the basic steps in message passing. Message passing simply means that (at a very abstract level) the fundamental mechanism of program execution is objects sending each other message. The important point is that the name and structure of these messages is not necessarily fixed beforehand in the source code and can itself be additional information

- Creating classes that define objects and its behavior.

www.madinpoly.com

- Creating objects from class definitions
- Establishing communication among objects

In OOPs **Dynamic Binding** refers to linking a procedure call to the code that will be executed only at run time. The code associated with the procedure in not known until the program is executed, which is also known as late binding.

- in the context of OOP typically refers to the binding of methods to messages
- methods varying dynamically entails much of the power of the OO approach
- main source of power in an OO language
- search for method (code body) to bind to a message starts from the class to which the receiver currently (i.e., at run-time) is an instance of, and and proceeds up the class inheritance hierarchy from there (static binding initiates the search from the class to which the reference variable pointing to the receiver was declared to be an instance of)
- if no method found anywhere, a run-time error (method not found) is reported and this is typically the only error in a Smalltalk program ever detected and reported

  Eg:

```
          Mammal m;
          Cow c;

          if (user input)
            m = new Cow;   // if static binding used, run method in class
          Mammal
                                   bound to run message here
                         // if dynamic binding used, run method in class
          Cow
                                   bound to run message here
          m.run
          else
            c = new Cow;
      c.run
```

b) Explain method overloading with example

7

Method overloading

www.madinpoly.com

Writing two or more methods in the same class with same name but different parameters list, is known as method overloading. If we have to perform only one operation, having same name of the methods increases the readability of the program. Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs. So, we perform method overloading to figure out the program quickly.

Eg:

```
class Complex
{
            intreal,image;
            void assign()
            {
                real=10;
                image=3;
            }
            void assign(int x, int y)
            {
                real = x;
                image = y;
            }
            void show()
            {
                System.out.println(real+"+"+image+"i");
            }
}
class Test
{
            public static void main(String args[])
            {
                Complex c = new Complex();
                c.assign(6);
                c.show();
            }
}
```

UNIT – II

V.

www.madinpoly.com

a) Explain implementation of interface with example

8

Interface in java is core part of Java programming language and one of the way to achieve abstraction in Java along with abstract class. Even though interface is fundamental object oriented concept ; Many Java programmers thinks Interface in Java as advanced concept and refrain using interface from early in programming career. At very basic level interface in java is a keyword but same time it is an object oriented term to define contracts and abstraction , This contract is followed by any implementation of Interface in Java. Since multiple inheritance is not allowed in Java, interface is only way to implement multiple inheritance at Type level.

1) Interface in java is declared using keyword interface and it represent a Type like any Class in Java. a reference variable of type interface can point to any implementation of that interface in Java
2) All variables declared inside interface is implicitly public final variable or constants. which brings a useful case of using Interface for declaring Constants. We have used both Class and interface for storing application wide constants and advantage of using Interface was that you can implement interface and can directly access constants without referring them with class name which was the case earlier when Class is used for storing Constants.
3) All methods declared inside Java Interfaces are implicitly public and abstract, even if you don't use public or abstract keyword. you can not define any concrete method in interface.
4) In Java its legal for an interface to extend multiple interface. for example following code will run without any compilation error: interface Session extends Serializable, Clonnable{ }

```
interface SessionIDCreator extends Serializable, Cloneable{
      String TYPE = "AUTOMATIC";
      int createSessionId();
  }
class SerialSessionIDCreator implements SessionIDCreator{
```

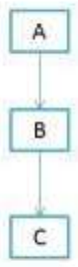
www.manishpoly.com

```
        private int lastSessionId;
        @Override
        public int createSessionId() {
            return lastSessionId++;
        }
    }
```

b) Explain multilevel inheritance with the help of example

7

In multilevel, one-to-one ladder increases. Multiple classes are involved in inheritance, but one class extends only one. The lowermost subclass can make use of all its super classes' members. Multilevel inheritance is an indirect way of implementing multiple inheritances.



Multilevel Inheritance

If we take the example of above diagram then class C inherits class B and class B inherits class A which means B is a parent class of C and A is a parent class of B. So in this case class C is implicitly inheriting the properties and method of class A along with B that's what is called multilevel inheritance.

```
Syntax : class A
         {
              …………………
              …………………
         }
         class B extends A
         {
              …………………
              …………………
         }
```

www.ma...y.com

```
class C extends B
{
      ....................
      ....................
}
```

Example:

```
Class A
{
  public void methodA()
  {
    System.out.println("Base class method");
  }
}

Class B extends A
{
  public void methodB()
  {
    System.out.println("Child class method");
  }
  public static void main(String args[])
  {
    B obj = new B();
    obj.methodA(); //calling super class method
    obj.methodB(); //calling local method
  }
}
```

OR

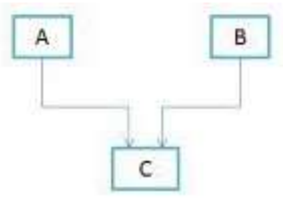
VI.

a) Explain with example how interface is used to implement multiple inheritance in java

8

"**Multiple Inheritance**" refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. The problem with "multiple inheritance" is that the derived class will have to manage the dependency on two base classes.

www.madinpoly.com



(b) Multiple Inheritance

The Java programming language supports *multiple inheritance of type*, which is the ability of a class to implement more than one interface. An object can have multiple types: the type of its own class and the types of all the interfaces that the class implements. This means that if a variable is declared to be the type of an interface, then its value can reference any object that is instantiated from any class that implements the interface. As with multiple inheritance of implementation, a class can inherit different implementations of a method defined (as default or static) in the interfaces that it extends. In this case, the compiler or the user must decide which one to use.

Eg:

```
class A
{
                void showA()
                {
                    System.out.println("class A");
                }
}
Interface B
{
                void showB();
}
class C extends A implements B
{
                void showC()
                {
                    System.out.println("class C");
                }
                public void showB()
                {
                    System.out.println("interface B");
                }
}
class Multi
```

www.madinpoly.com

```
    {
                    public static void main(String args[])
                    {
                        C c1=new C();
                        c1.showA();
                        c1.showB();
                        c1.showC();
                    }
  }
```

b) Explain method overriding with the help of example

7

- In Method Overriding, sub class has the same method with same name and exactly the same number and type of parameters and same return type as a super class.
- Method Overriding means method of base class is re-defined in the derived class having same signature.
- Method Overriding is to "Change" existing behavior of method.
- It is a **run time polymorphism.**
- It always requires inheritance in Method Overriding.
- In Method Overriding, methods must have **same signature.**
- In Method Overriding, relationship is there between methods of super class and sub class.
- In Method Overriding, methods have same name and same signature but in the different class.Method Overriding requires at least two classes for overriding.

```
Eg: Class A              // Super Class
    {
     void display(intnum)
     {
        printnum ;
      }
    }                    //Class B inherits Class A
    Class B              //Sub Class
    {
     void display(intnum)
```

www.msbte.com

```
    {
       printnum ;
    }
   }
```

UNIT – III

VII.

a) What is mean by multithreading. What happened when a method is declared as synchronized

Multithreading in java is a process of executing multiple threads simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. But we use multithreading than multiprocessing because threads share a common memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process. Java Multithreading is mostly used in games, animation etc. Java is a *multithreaded programming language* which means we can develop multithreaded program using Java. A multithreaded program contains two or more parts that can run concurrently and each part can handle different task at the same time making optimal use of the available resources especially when your computer has multiple CPUs.

Advantage of Java Multithreading

1) It doesn't block the user because threads are independent and you can perform multiple operations at same time.

2) You can perform many operations together so it saves time.

3) Threads are independent so it doesn't affect other threads if exceptions occur in a single thread.

When we declare a method synchronized, java creates a monitor and hand it over to the thread that calls the method first time. As long as the thread holds the monitor, no other thread can enter the synchronized section of code. Whenever a thread has completed its work, it will hand over the monitor to the next thread that is ready to use the same resource.

Eg:

```
synchronized void update()
{
    ………………………
    ………………………
}
```

b) Explain naming convention of java packages

6

Package names should be unique and consist of lowercase letters. With small projects that only have a few packages it's okay to just give them simple (but meaningful!) names.

```
package pokeranalyzer
package mycalculator
```

Underscores may be used if necessary:

```
package com.oreilly.fish_finder;
```

Publicly available packages should be the reversed Internet domain name of the organization, beginning with a single-word top-level domain name (e.g., *com*, *net*, *org*, or *edu*), followed by the name of the organization and the project or division. (Internal packages are typically named according to the project.)

Package names that begin with java and javax are restricted and can be used only to provide conforming implementations to the Java class libraries.

In software companies and large projects where the packages might be imported into other classes, the names will normally be subdivided. Typically this will start with the company domain before being split into layers or features:

```
package com.mycompany.utilities
package org.bobscompany.application.userinterface
```

OR

VIII.

a) Explain the steps for creating a user defined package. What is the use of import statement

9

Following steps are to be followed in creating and using packages.

1. Create a package with a .class file
2. set the classpath from the directory from which you would like to access. It may be in a different drive and directory. Let us call it as a target directory.
3. Write a program and use the file from the package.

Let us create a package called **forest** and place a class called **Tiger** in it. Access the package from a different drive and directory.

1st Step: Create a package (forest) and place Tiger.class in it.

```
package forest; import java.util.*;
public class Tiger
 {
        public void getDetails(String nickName, int weight)
        {
                System.out.println("Tiger nick name is " + nickName);
                System.out.println("Tiger weight is " + weight);
        }
}
```

Order of Package Statement

The above program codingwise is very simple but is important to know the steps of package creation.

```
package forest;
import java.util.*;
public class Tiger
```

Package is a keyword of Java followed by the package name. Just writing the package statement followed by the name creates a new package; see how simple Java is to practice. For this reason, Java is known as a production language.

While creating packages, the order of statements is very important. The order must be like this, else, compilation error.

1. Package statement
2. Import statement
3. Class declaration

www.madinpoly.com

If exists, the package statement must be first one in the program. If exists, the import statement must be the second one. Our class declaration is the third. Any order changes, it is a compilation error.

When the code is ready, the next job is compilation. We must compile with package notation

b) Write short note on following thread methods

a) Sleep()

b) Suspend()

c) Wait()

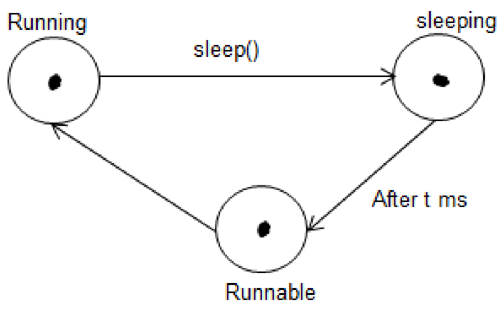
6

Blocking of a thread

A thread can also be temporarily blocked from entering into the runnable and subsequently running state by using the following methods:

1) sleep()

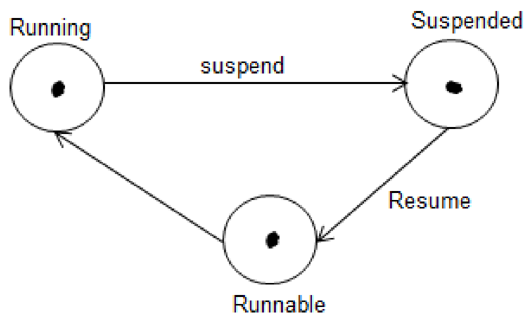
sleep() is a static method that is used to send the calling thread into a non-runnable state for the given duration of time. The important part for this is recognizing the "calling thread", which is actually the thread in which the sleep() method is invoked rather than the thread object which may (which is essentially a violation of Java standards) invoke this method



2) suspend()

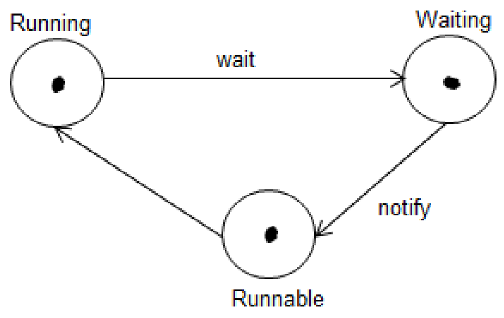
It suspends the thread on which it is invoked. If the target thread holds a lock on the monitor protecting a critical system resource when it is suspended, no thread can access this resource until the target thread is resumed. If the thread that would resume the target thread attempts to lock this monitor prior to calling resume, deadlock results.

www.diplomapoly.com



3) wait()

wait() Causes current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.



These methods cause the thread to go into the blocked state. The thread will return to the runnable state when specified time is elapsed in the case of sleep(), the resume() method is invoked in the case of suspend(), and the notify() method is called in the case of wait().

UNIT – IV

IX.

a) What is meant by finally statement? How it is used in java. Give example

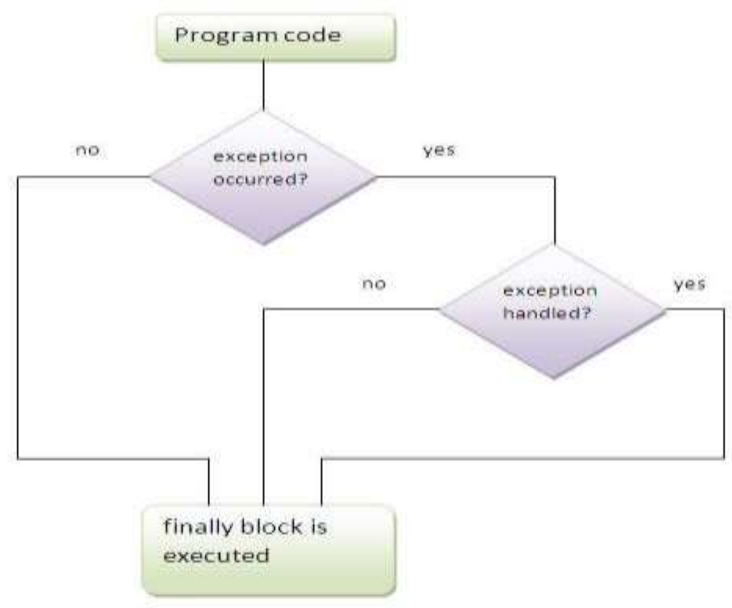
Java finally block is a block that is used *to execute important code* such as closing connection, stream etc.

Java finally block is always executed whether exception is handled or not.

www.mc...v.com

Java finally block must be followed by try or catch block.

- The finally block always executes immediately after try-catch block exits.
- The finally block is executed incase even if an unexpected exception occurs.
- The main usage of finally block is to do clean up job. Keeping cleanup code in a finally block is always a good practice, even when no exceptions are occurred.
- The runtime system always executes the code within the finally block regardless of what happens in the try block. So it is the ideal place to keep cleanup code



Eg(1):

```
try
{
   ………………
   ………………
}
finally
{
   ………………
   ………………
}
```

Eg(2):

```
try
{
   ………………
   ………………
}
catch(……)
{
```

www.m...y.com

```
    .................
    .................
}
catch(......)
{
    .................
    .................
}
finally
{
    .................
    .................
}
```

b) Write the basic stream type in java

7

A stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination. InputStream and OutputStream are the basic stream classes in Java.

All the other streams just add capabilities to the basics, like the ability to read a whole chunk of data at once for performance reasons (BufferedInputStream) or convert from one kind of character set to Java's native unicode (Reader), or say where the data is coming from (FileInputStream, SocketInputStream and ByteArrayInputStream, etc.)

Input streams

The class `java.io.InputStream` is the base class for all Java IO input streams. If you are writing a component that needs to read input from a stream, try to make our component depend on an `InputStream`, rather than any of its subclasses (e.g. `FileInputStream`). Doing so makes your code able to work with all types of input streams, instead of only the concrete subclass

Following is the declaration for **Java.io.InputStream** class:

```
public abstract class InputStream
 extends Object
   implements Closeable
```

www.madinpoly.com

OutputStream

The class java.io.OutputStream is the base class of all Java IO output streams. If you are writing a component that needs to write output to a stream, try to make sure that component depends on an OutputStream and not one of its subclasses.

Following is the declaration for **Java.io.OutputStream** class:

```
public abstract class OutputStream
  extends Object
    implements Closeable, Flushable
```

OR

X.

a) Illustrate the use of FileInputStream class

9

Java FileInputStream class obtains input bytes from a file.It is used for reading streams of raw bytes such as image data. For reading streams of characters, consider using FileReader. The FileInputStream class makes it possible to read the contents of a file as a stream of bytes. TheFileInputStream class is a subclass of **InputStream**. This means that you use the FileInputStream as an InputStream (FileInputStream behaves like an InputStream).

It should be used to read byte-oriented data for example to read image, audio, video etc.

Example of FileInputStream class

```
import java.io.*;
class SimpleRead{
 public static void main(String args[]){
  try{
    FileInputStream fin=new FileInputStream("abc.txt");
    int i=0;
    while((i=fin.read())!=-1){
     System.out.println((char)i);
    }
    fin.close();
```

www.ktunotes.com

```
        }catch(Exception e){system.out.println(e);}
       }
      }
```

b) Distinguish between the init() and start() method of applet

6

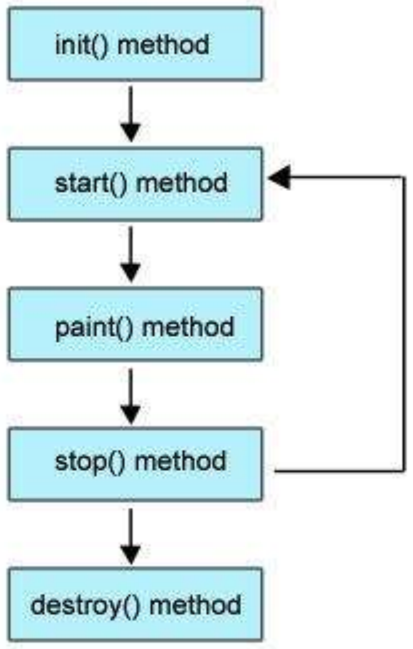
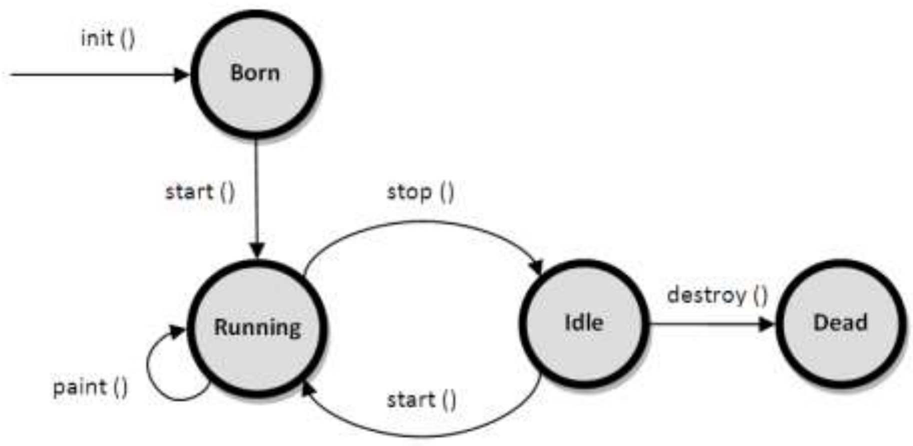


Figure: Life cycle of Applet

- **Born or initialization state (init()):** The applet's voyage starts here. In this method, the applet object is created by the browser. Because this method is called before all the other methods, programmer can utilize this method to instantiate objects, initialize variables, setting background and foreground colors in GUI etc.; the place of a constructor in an application. It is equivalent to born state of a thread.
- **Running state (start()):** In init() method, even though applet object is created, it is ininactive state. An inactive applet is not eligible for microprocessor time even though the microprocessor is idle. To make the applet active, the init() method calls start() method. In start() method, applet becomes active and thereby eligible for processor time.



www.madinpoly.com