TED (10)-3069 Reg. No. …….

(REVISION-2010) Signature ……..……

FORTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/ TECHNOLIGY- MARCH, 2012

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 declaration statement of main method

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

2. State the use of final keyword

   1) To create constants

      Eg: final int size = 10;

   2) To prevent the sub-classes from overriding the members of the super class

```
final void show(])
{
   ……
}
```

   3) To make a class that cannot be sub-classed

```
finalclass A
{
   ……
}
```

3. List the visibility controls

1) private
2) public
3) default/friendly access
4) protected

4. List the two methods used to stop threads

1) suspend()
2) stop()

5. State the use of finally block

Can be used to handle any exception generated within a try block

Eg:
```
try
    {
        ……
    }
catch(……..)
    {
        ……
    }
finally
    {
        ……
    }
```

PART – B

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

1. Explain the creation of objects. Give example

As mentioned previously, a class provides the blueprints for objects. So basically an object is created from a class. In Java, the new key word is used to create new objects.

Syntax: ClassNameobjectName;

objectName = new ClassName();

There are three steps when creating an object from a class:

- Declaration: A variable declaration with a variable name with an object type.

  Eg: Complex c
- Instantiation: The 'new' key word is used to create the object.

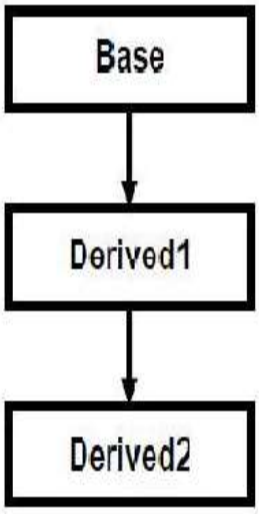
Eg: c = new Complex();

- Initialization: The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

2. List the benefits of OOP.
   - Through inheritance, we can eliminate redundant code and extend the use of existing classes.
   - We can build programs from standard working modules that communicate with one another rather than, having to start writing the code from scratch. This leads to saving of development time and higher productivity.
   - The principle of data hiding helps the programmers to built secure program that can't be invaded by code in other parts of the program.
   - It is possible to have multiple objects to coexist without any interference.
   - It is possible to map objects in the problem domain to those objects in the program.
   - It is easy to partition the work in a project based on objects.
   - The data-centered design approach enables us to capture more details of the model in an implementable form.
   - Object-oriented systems can be easily upgraded from small to large system

3. Illustrate multilevel inheritance with the help of an example

   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.



Eg: class A

```
{
    ……
    ……
}
class B extends A
{
    ……
    ……
}
class C extends B
{
    ……
    ……
}
```

4. List the similarities between interfaces and classes
   - An interface is basically a kind of class
   - There are both java basic object types
   - They both contain variable and methods but with a difference(classes contain fields and methods whereas interface contain abstract class and final)
   - The syntax for defining an interface is very similar to that for defining a class
   - Like classes interfaces can also be extended

- They can both be inherited(extend keyword for classes and implement keyword for inheritance)

5. Explain any 3 java packages
   a) Language support packages(java.lang)

      java.lang Provides classes that are fundamental to the design of the Java programming language such as String, Math, and basic runtime support for threads and processes.

      Example classes: String, Thread. It contain language support classes

   b) Networking Packages(java.net)

      The java.net package provides a powerful and flexible infrastructure for networking. Many of the classes in this package are part of the networking infrastructure and are not used by normal applications; these complicated classes can make the package a difficult one to understand.

      Example classes: InetAddresss, Socket

   c) i/o Package(java.io)

      Java.io package provides classes for system input and output through data streams, serialization and the file system. The Java I/O package, a.k.a. java.io, provides a set of input streams and a set of output streams used to read and write data to files or other input and output sources. There are three categories of classes in java.io: input streams, output streams and everything else.

      Example classes: InputStreamReader, BufferedReader

6. Explain thread synchronization with suitable example

   When we start two or more threads within a program, there may be a situation when multiple threads try to access the same resource and finally they can produce unforeseen result due to concurrency issue. For example if multiple threads try to write within a same file then they may corrupt the data because one of the threads can override data or while one thread is opening the same file at the same time another thread might be closing the same file.

So there is a need to synchronize the action of multiple threads and make sure that only one thread can access the resource at a given point in time. This is implemented using a concept called **monitors**. Each object in Java is associated with a monitor, which a thread can lock or unlock. Only one thread at a time may hold a lock on a monitor.

Java programming language provides a very handy way of creating threads and synchronizing their task by using **synchronized** blocks. You keep shared resources within this block.

Syntax:

```
synchronized (objectidentifier)
{
   // Access shared variables and other shared resources
}
```

Here, the **objectidentifier** is a reference to an object whose lock associates with the monitor that the synchronized statement represents. Now we are going to see two examples where we will print a counter using two different threads. When threads are not synchronized, they print counter value which is not in sequence, but when we print counter by putting inside synchronized() block, then it prints counter very much in sequence for both the threads.

7. Distinguish between input stream and Reader class

- An InputStream is the raw method of getting information from a resource. It grabs the data byte by byte without performing any kind of translation. If you are reading image data, or any binary file, this is the stream to use.

- A Reader is designed for character streams. If the information you are reading is all text, then the Reader will take care of the character decoding for you and give you Unicode characters from the raw input stream. If you are reading any type of text, this is the stream to use.

- Reader support only character and input/output stream support only bytes input stream and the java reader class has only one difference and that is input stream only

  support bytes and the reader class only supports the character.

- These are IO streams but differences their InputStream class is byte-oriented streams. This streams read and write in byte oriented. Input Stream are never support Unicode characters.
- Reader class are belongs to the character stream this streams read and write in character oriented Streams. Reader is never supports Unicode characters.

(5 x 6 = 30)

PART – C

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

UNIT – I

III.

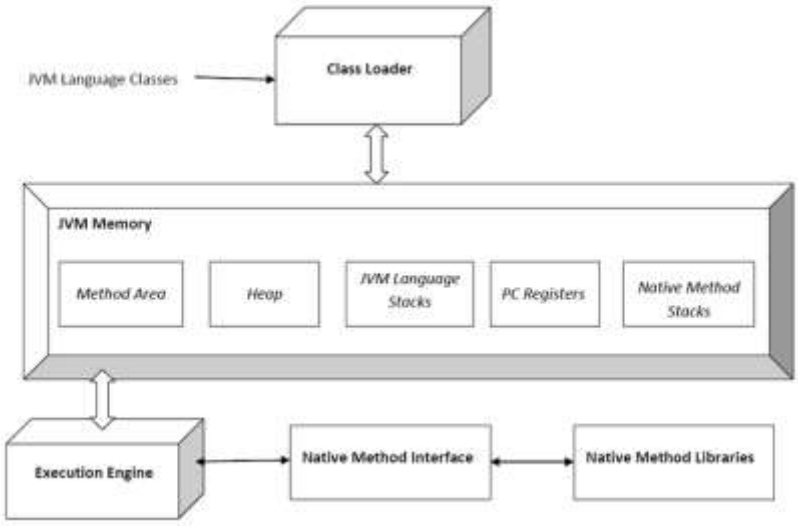
a) Distinguish between class method and instance methods. Give example to each

8

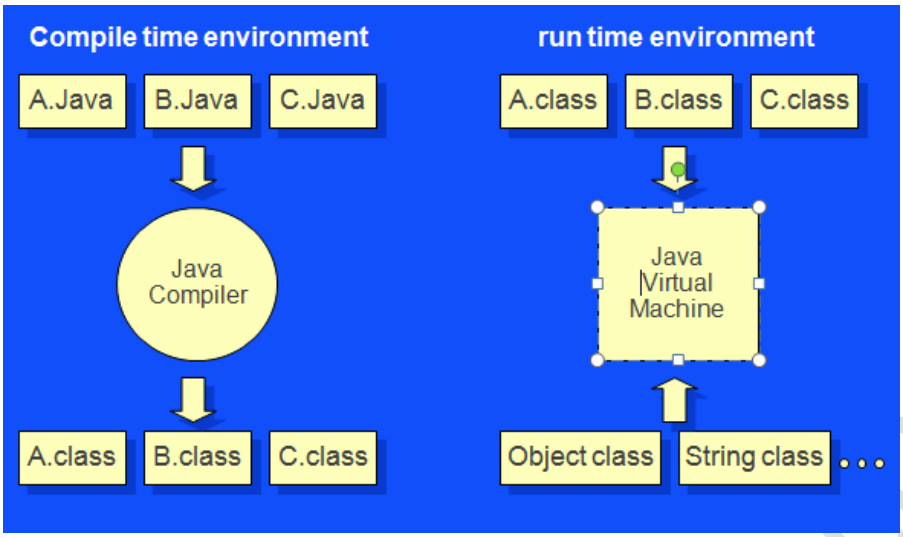
Instance method	Class method
They are non-static methods	They are declared using the keyword static
They are called using objects	They are called without using objects. They are called using class name
They can access both static and non-static members	They can only access other static members
They can refer to this or super keyword in any way	They cannot refer to this or super keyword in any way
Class method can be called from within an instance method	Instance methods cannot be called from within a class method

b) Describe java virtual machine

7



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.

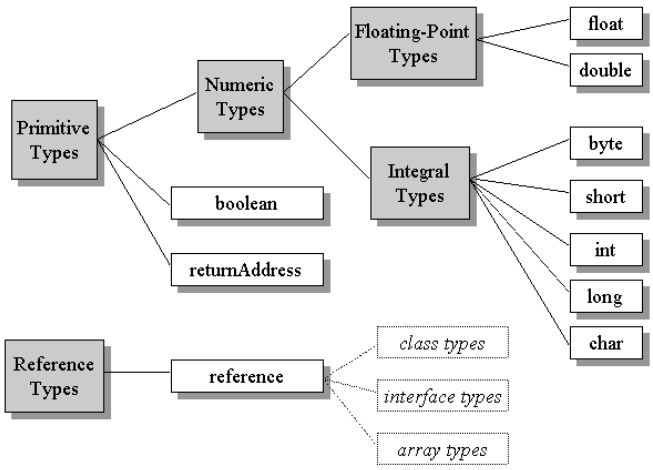


OR

IV.

a) Explain the data types in java

7



There are totally eight primitive data types in Java. They can be categorized as given below:

- Integer types (Does not allow decimal places)
  - byte
  - short
  - int
  - long
- Rational Numbers(Numbers with decimal places)
  - float

- double
- characters
  - char
- conditional
  - boolean

Type	Length	Minimum Value	Maximum Value
byte	8 bits	-128	127
short	16bits	-32768	32767
int	32 bits	-2,147,483,648	2,147,483,647
long	64bits	-9,223,372,036,854,775,808	9,223,372,036,854,775,80
float		-3.4E+38	+3.4E+38
double		-1.7E+308	+1.7E+308
boolean	**Possible values are two - true or false**		

- char
  - char is 16 bit type and used to represent Unicode characters.
  - Range of char is 0 to 65,536.
  - Example . char letter ='A' .
- Reference Data Types :
  - Reference data types or objects are created using class constructors. It's type can't be changed after declaration.
  - Default value of any reference variable is null.
  - A reference variable can be used to refer to any object of the declared type or any compatible type.
  - Example : Mainclass mc = new Mainclass("Ankit") .

b) Write a java program to find the area of circle, rectangle and obtuse angle using method overloading


```
class MethodOverloading
{
  void area()
  {
                    float a=3.14F*r*r;
                    System.out.println("Area of circle="+a);
```

```
  }
  void area(int l,int b)
  {
                    int a= l*b;
                    System.out.println("Area of rectangle="+a);
  }
  void area()
  {
                    float a=(b*h)/2;
                    System.out.println("Area of triangle="+a);
  }
}
class Test
{
  public static void main(string args[])
  {
                    MethodOverloading o = new MethodOverloading();
                    o.area(5);
                    o.area(10,8);
                    o.area(4.5,5.5);
  }
```

UNIT – II

V.

a) Explain method overriding with 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)
  {
printnum ;
  }
}
```

b) Explain the implementation of interface with suitable 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{

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

}
```

OR

VI.

a) Describe extending of interface with example

7

An interface can extend another interface, similarly to the way that a class can extend another class. The **extends** keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

Syntax:

```
interface ChildInterface extends ParentInterface
{
  Body;
}

interface A
{
  in code =101;
  String name=”oop”;
}
```

```
interface B extends A
{
  void show();
}
```

A Java class can only extend one parent class. Multiple inheritances is not allowed. Interfaces are not classes, however, and an interface can extend more than one parent interface.

The extends keyword is used once, and the parent interfaces are declared in a comma-separated list

```
Syntax:
interface C extends A,B
{
  Body;
}
```

b) Write a java program to demonstrate abstract classes and methods

8

```
abstract class A
{
  abstract void show();
}
class B extends A
{
  void print()
  {
        System.out.println("class B");
  }
  void show()
  {
        System.out.println("Abstract class A");
  }
}
class Test
{
  public static void main(String args[])
  {
        B b1=new B();
        b1.show();
        b1.print();
  }
}
```

UNIT – III

VII.

a) Explain the creation of packages

7

To create a package, you choose a name for the package (naming conventions are discussed in the next section) and put a package statement with that name at the top of *every source file* that contains the types (classes, interfaces, enumerations, and annotation types) that you want to include in the package.

The package statement (for example, package graphics;) must be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

- package <package_name>;

If you put multiple types in a single source file, only one can be public, and it must have the same name as the source file. For example, you can define public class Circle in the file Circle.java, define public interface Draggable in the file Draggable.java, define public enum Day in the file Day.java, and so forth.

You can include non-public types in the same file as a public type (this is strongly discouraged, unless the non-public types are small and closely related to the public type), but only the public type will be accessible from outside of the package. All the top-level, non-public types will be *package private*.

```
Package test;
    public class A
    {
      public void show()
      {
          System.out.println("class A in package test");
      }
    }
```

b) Write a java program to illustrate the use of multithreads

8

```
class NewThread implements Runnable
{
```

```
String name;
Thread t;
NewThread(String threadname)
{
        name=threadname;
        t=new Thread(this,name);
        System.out.println("New Thread:"+t);
        t.start();
}
public void run()
{
        try
        {
                for(int i=5;i>0;i--)
                {
                        System.out.println(name+ ":"+i);
                        Thread.sleep(1000);
                }
        }
        catch(InterruptedException e)
        {
                System.out.println(name+" Interrupted");
        }
        System.out.println(name+" exiting");
}
}
class MultiThreadDemo
{

public static void main(String[] args)
{
        new NewThread("One");
        new NewThread("Two");
        new NewThread("Three");
        try
        {
                Thread.sleep(10000);
        }
        catch(InterruptedException e)
        {
                System.out.println("Main Thread Interrupted");
        }
        System.out.println("Main Thread Exiting");
```

```
    }
}
```

OR

VIII.

a) Illustrate hiding of classes from outside the package with example

7

One of the most useful features of Java packages is the ability to grant access to classes, interfaces, methods, or fields exclusively to other members of the same package. This feature gives the package an internal implementation and an external interface. It provides the usual advantages of a hidden internal implementation: robustness and ease of modification. The robustness arises from the inability of types declared in other packages to incorrectly manipulate the internal implementation of the package. Types in other packages must go through the external interface of the package, and the package maintains control of its internal implementation. Ease of modification comes from the ability to change the internal implementation of the package without affecting the code of other packages, which is tied only to the external interface.

First step you can take to hide the internal implementation of a package is to declare as public only those types that are needed by other packages. When you declare a class or interface, it is by definition contained in a package. If you want a class or interface to be accessible to types declared in other packages, you must declare it public. If you do not declare it public, it will only be accessible to types in the same package. Therefore, you can denote some types (the public ones) as part of the external interface of the package. The other types (the ones that aren't public) are part of the internal implementation of the package.

```
package p1
public class X
{
   body;
}
class Y
{
   body;
```

}

Here class X is public. So it is available outside the package. But class Y is not public, so it is hidden. Consider the following code which imports package p1, that contains the classes X and Y

```
import p1.*;
X object;          // Right.class X is not available
Y object;          //Wrong.class is not available
```

b) Explain with an example how java performs thread synchronization 8

The Java synchronized keyword is an essential tool in concurrent programming in Java. Its overall purpose is to only allow one thread at a time into a particular section of code thus allowing us to protect, for example, variables or data from being corrupted by simultaneous modifications from different threads. This article looks at how to use synchronized in Java to produce correctly functioning multithreaded programs. Other articles in this section look at other Java 5 concurrency facilities which have in fact superseded synchronized for certain tasks.

At its simplest level, a block of code that is marked as synchronized in Java tells the JVM: *"only let one thread in here at a time"*.

Imagine, for example, that we have a counter that needs to be incremented at random points in time by different threads. Ordinarily, there would be a risk that two threads could simultaneously try and update the counter at the same time, and in so doing currpt the value of the counter (or at least, miss an increment, because one thread reads the present value unaware that another thread is just about to write a new, incremented value). But by wrapping the update code in a synchronized block, we avoid this risk:

```
public class Counter
{
  private int count = 0;
  public void increment()
  {
    synchronized (this)
   {
      count++;
```

```
    }
  }
  public int getCount()
  {
    synchronized (this)
    {
      return count;
    }
  }
}
```

UNIT – IV

IX.

a) Explain exception handling and the tasks incorporated with exception handling

7

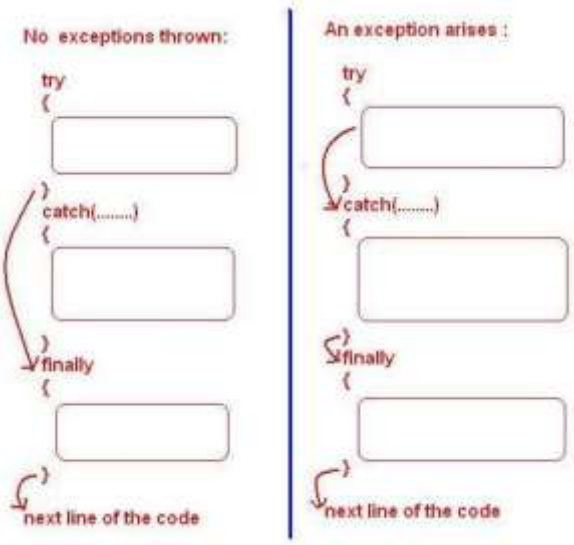
The **exception handling** in java is one of the powerful *mechanism to handle the runtime errors* so that normal flow of the application can be maintained. In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime. Exception Handling is a 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.
- 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)



b) Explain the Applet life cycle

8

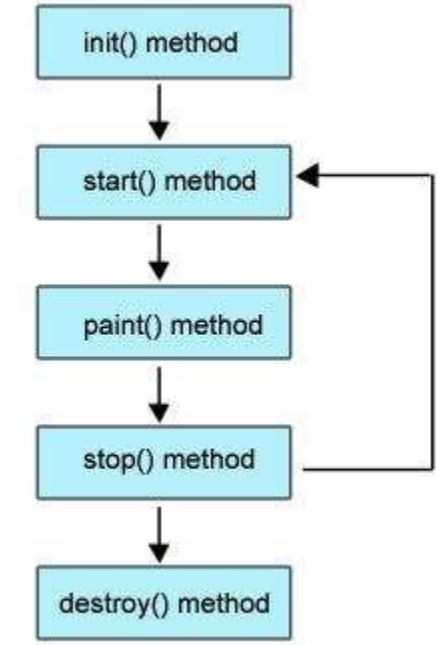
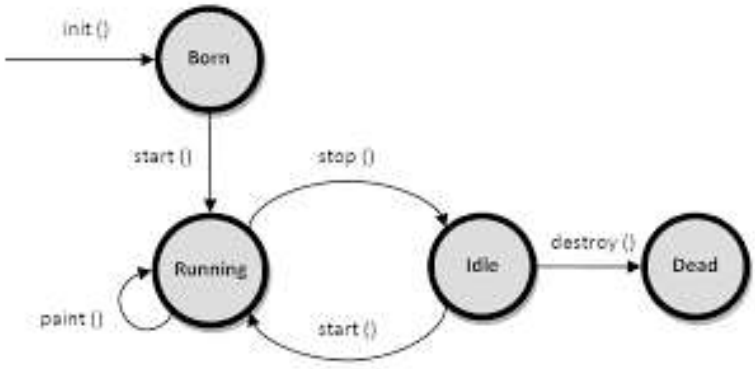


Figure: Life cycle of Applet

- **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.
- **Display state (paint()):** This method takes a java.awt.Graphics object as parameter. This class includes many methods of drawing necessary to draw on the applet window. This is the place where the programmer can write his code of what he expects from applet like animation etc. This is equivalent to **runnable state**of thread.
- **Idle state (stop()):** In this method the applet becomes temporarily inactive. An applet can come any number of times into this method in its life cycle and can go back to the active state (paint() method) whenever would like. It is the best place to have cleanup code. It is equivalent to the **blocked state** of the thread.
- **Dead or destroy state (destroy()):** This method is called just before an applet object is garbage collected. This is the end of the life cycle of applet. It is the best place to have cleanup code. It is equivalent to the **dead state** of the thread.



OR

X.

a) Describe the creation and running of applets

7

Creating Applets

1) Import the necessary packages

   Eg: import java.applet.*;

Import java.awt.*;

2) Define a class which extends the Applet class and declare it public.

Eg: public class MyApplet extends Applet
{
……
……
}

3) Save this file with the name of the public class. Ie MyApplet.java

4) Compile the file

Running Applet

For running applets, we must design a web page that include the above created applet

1) Design a webpage

Eg: <html>
……..
……..
</html>

2) Include the applet tag

<applet code=MyApplet.class>
Width=200
Height=200>
</applet>

3) Run the applet(ie. Webpage) using java enabled web browser or java applet viewer

Applet to display a welcome message

```
import java.awt.*;
import java.applet.*;
public class Hello extends Applet
{
  public void paint(Graphics g)
  {
    g.drawString("Welcome",100,100);
  }
}
```

Save this file as Hello.java and compile it

Design web page

```
<html>
    <applet code=Hello.class
```

```
                Width=200
                Height=200>
            </applet>
        </html>
```

b) Demonstrate reading and writing in file

1) Reading or writing characters
2) Reading or writing Bytes

Reading or writing characters

We can read or write characters to a file. The classes used for handling characters are FileReader(for reading characters) and FileWriter (for writing characters)

Program to write characters to a file

```
Import java.io.*;
Class WriteCharacters
{
    public static void main(String args[]) throw IOException
    {
        char b[]={'a','k','b','a','r'};
        FileWriter f = new FileWriter("D:/wc.txt");
        f.write(b);
        f.close();
    }
}
```

Program to read characters from a file

```
import java.io.*;
class ReaderCharacters
{
    public static void main(string args[]) throw IOException
    {
        int b;
        FileWriter f = new FileWriter("D:/wc.txt");
        While((b=f.read())!=-1)
                System.out.print((char)b);
        f.close();
    }
}
```

Program to read bytes to a file

```
Import java.io.*;
Class WriteBytes
{
    public static void main(String args[]) throw IOException
    {
        byte b[]={'a','k','b','a','r'};
        FileOutputStream f = new FileOutputStream("wb.txt");
        f.write(b);
        f.close();
   }
}
```

Program to read bytes from a file

```
        import java.io.*;
        class ReaderBytes
        {
            public static void main(string args[]) throw IOException
            {
                int b;
                FileInputStream f = new FileInputStream("wb.txt");
                While((b=f.read())!=-1)
                        System.out.print((char)b);
                f.close();
            }
        }
```