MCS-024/ Object Oriented Technologies and Java

1. (a) What is Object Oriented Programming? Explain basic components of Object Oriented Programming.

Answer: - **Object oriented programming: -** is the methodology that allows the association of data structures with operations similar to the way it is perceived in the human mind. The following are the important features of object-oriented programming: -

*. Improvement over the structured programming paradigm
*. Emphasis on data rather than algorithm
*. Data abstraction is introduced in addition to procedural abstraction.
*. Data and associated operations are unified into a single units, thus the objects are grouped with common attributes, operations and semantics.
*. Programs are designed around the data being operated rather than operations themselves.
*. Relationship can be created between similar, yet distinct data types.

Concept of OOP is as follows: -
The fundamental features of the OOPs are the following:
*. Encapsulation
*. Data abstraction
*. Inheritance
*. Polymorphism
*. Message passing
*. Extensibility
*. Persistence
*. Delegation
*. Genericity
*. Multiple inheritances

Encapsulation: - it is a mechanism that associates the code and the data it manipulates into a single unit and keeps them safe from external interference and misuse.
Data abstraction: - the technique of creating new data types that are well suited to an application to be programmed is known as data abstraction.
Inheritance: - it allows that extension and reuse of existing code without having to rewrite the code from scratch.
Multiple inheritances: - the mechanism by which a class is derived from more than one base class is known as multiple inheritances.
Polymorphism: - it allows a single name/operator to be associated with different operations depending on the types of data passed to it.
Message passing: - it is the process of invoking an operation on an object
Extensibility: - it is a feature, which allows the extension of the functionality of the existing s/w components.
Persistence: - the phenomenon where the object outlives the program execution time and exists between executions of a program is known as persistence.
Delegation: - it is an alternative to class inheritance. Delegation is a way of making object composition as powerful as inheritance.
Genericity: - it is a technique for defining s/w components that have more than one interpretation depending on the data type of parameters.

Q.1 (b) Explain abstraction and encapsulation with the help of Examples

Answer: - abstraction is the process of generalization unknown object oriented programming language such as C. there was no mechanism to show different local and global data member, hence accidently or internally programmer used to correct value of global member.

In OOPs, structure this problem is solved by confining the scope of data into logical unit called classes. Data of a class can be directly return by within a class. Outer class need class objects or public methods to refer these data members.

Q.1 (c) Explain concept of java virtual machine

Answer: - JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java byte code can be executed.

JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent.

A specification where working of Java Virtual Machine is specified But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.

An implementation its implementation is known as JRE (Java Runtime Environment).

Runtime Instance Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

The JVM performs following operation:

*. Loads code
*.Verifies code
*.Executes code
*. Provides runtime environment

JVM provides definitions for the:

Memory area
Class file format
Register set
Garbage-collected heap
Fatal error reporting etc

2. (a) Write a java program to demonstrate use of different operators available in java.

Answer: - The following program is a simple example that demonstrates the bitwise operators. Copy and paste the following Java program in Test.java file and compile and run this program –

```
public class Test
 {

   public static void main(String args[])
{
     int a = 60;                    /* 60 = 0011 1100 */
     int b = 13;                    /* 13 = 0000 1101 */
     int c = 0;

     c = a & b;                     /* 12 = 0000 1100 */
     System.out.println("a & b = " + c );

     c = a | b;                     /* 61 = 0011 1101 */
     System.out.println("a | b = " + c );

     c = a ^ b;                     /* 49 = 0011 0001 */
```

```
    System.out.println("a ^ b = " + c );

    c = ~a;                                  /*-61 = 1100 0011 */
    System.out.println("~a = " + c );

    c = a << 2;                              /* 240 = 1111 0000 */
    System.out.println("a << 2 = " + c );

    c = a >> 2;                               /* 15 = 1111 */
    System.out.println("a >> 2  = " + c );

    c = a >>> 2;                              /* 15 = 0000 1111 */
    System.out.println("a >>> 2 = " + c );
  }
}
```

This will produce the following result –

```
a & b = 12
a | b = 61
a ^ b = 49
~a = -61
a << 2 = 240
a >> 15
a >>> 15
```

Q.2 (b) Explain followings in context of java, with the help of example

(i) Access specifies and Inheritance

(ii) Application program and applet program

Answer: - (i) Access specifies and Inheritance

The access modifier in java specifies accessibility (scope) of a data member, method, constructor or class.

There are 4 types of java access modifiers:

Private
Default
Protected
Public

```
class A
{
private int data=40;
private void msg()
{
System.out.println("Hello java");
}
}

public class Simple
{
 public static void main(String args[])
{
```

```
  A obj=new A();
  System.out.println(obj.data);          //Compile Time Error
  obj.msg();                             //Compile Time Error
  }
}
```

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as super class (base class, parent class).
Following is an example demonstrating Java inheritance. In this example, you can observe two classes namely Calculation and My Calculation.

Using extends keyword, the My Calculation inherits the methods addition () and Subtraction () of Calculation class.

```
Class Calculation {
  int z;

public void addition(int x, int y) {
  z = x + y;
System.out.println("The sum of the given numbers:"+z);
  }

public void Subtraction(int x, int y) {
z = x - y;
System.out.println("The difference between the given numbers:"+z);
}
}

public class My_Calculation extends Calculation {
 public void multiplication(int x, int y) {
  z = x * y;
  System.out.println("The product of the given numbers:"+z);
    }

public static void main(String args[]) {
 int a = 20, b = 10;
 My_Calculation demo = new My_Calculation();
demo.addition(a, b);
demo.Subtraction(a, b);
    demo.multiplication(a, b);
   }
   }
```

(ii) Application program and applet program: -

Java application program	Java applet
It is executed independently. application are executed at command line by java.exe	Applet cannot be executed independently. Applets can only be executed inside a java compatible container, such as a browser or applet viewer.
It contain main () method. It has a single point for execution	It does not contain main () method, and does not have a single point of entry for execution.
It has no inherent security restrictions, and can perform read/write to files in local system.	Applet cannot perform read/write to files in local system this is to provide security.
Applications have no special support in HTML for embedding or downloading	Applet can be embedded in HTML pages and downloaded over the internet

For example: -

```
<HTML>
        <HEAD>
       <TITLE> A simple applet</TITLE>
        </HEAD>
       <BODY>
       Here is the output of my program:
       <APPLET CODE="MY FIRST APPLET.CLASS" WIDTH=150
      HEIGHT=50>
     </APPLET>
       </BODY>
  </HTML>
```

3. (a) Create array of objects in a java program and pass it as an argument in a method.

```
package object;

import java.io.IOException;
import java.util.Scanner;


class Demo
{

   int l,w;

   void input(){
      Scanner rd=new Scanner(System.in);
      System.out.println("enter the length");
      l=rd.nextInt();
      System.out.println("enter the width");
      w=rd.nextInt();
   }
   void area(Demo obj[]){
      int a=obj[0].l*obj[0].w;
      System.out.println("area of ractangle:"+a);
   }
```

```
}

class Test
{
   public static void main(String arg[]) throws IOException
   {
      Demo obj[]=new Demo[2];
      System.out.println("Enter for first");
      obj[0].input();
      obj[0].area(obj);
       System.out.println("Enter for second");
      obj[1].input();
      obj[1].area(obj);
   }
}
```

Q.3 (b) Write a java program to demonstrate different use of final variable.

There is a final variable speed limit, we are going to change the value of this variable, but It can't be changed because final variable once assigned a value can never be changed. For example: -

```
class Bike9
{
 final int speed limit=90;                    //final variable
 void run()
{
  speed limit=400;
 }
 public static void main(String args[])
{
 Bike9 obj=new Bike9 ();
 obj.run();
 }
}                                  //end of class
```


Q.3 (c) Write a java program to create Ticket class with proper constructor, to create a railway ticket. Define a method to display ticket details.

```
package object;


import java.util.Scanner;



class Demo
{

   int number,fair;
  String source,destination;
```

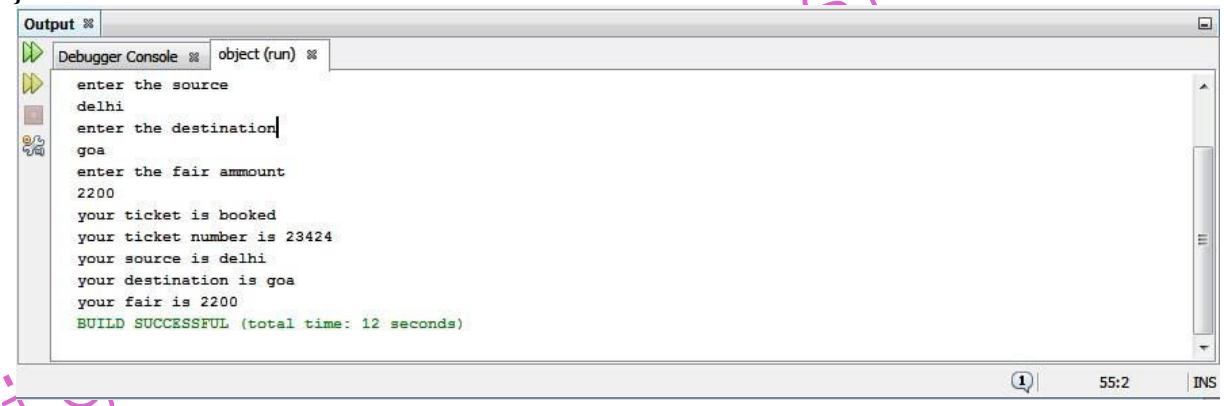
```
  Demo(int n,String s,int f,String d)
  {
     number=n;
     fair=f;
     source=s;
     destination=d;

  }

   Demo() {
      throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
   }
   void display(){
     System.out.println("your ticket is booked");
     System.out.println("your ticket number is "+number);
     System.out.println("your source is "+source);
     System.out.println("your destination is "+destination);
     System.out.println("your fair is "+fair);


   }


}
```

4. (a) What is polymorphism? It provides flexibility in application development? Yes or No? Justify your answer with the help of an example.

Answer: - it is natural phenomena which represent the thing are the found different shape and size in nature. In order to identify the thing they need to be named. Name can be given on the basis of shape a size or on the basis of utility shape and size multiple for a single utility. Hence, in real life to simplify the identification things are named on the basis of a utility.
In the contrast of programming concept of polymorphism state can be perform is multiple ways. Task can be named on the basis of the way it is performed.
Taking this due from real life methods is implemented. On the basis of task. i.e., if a task is to be performed in different way then we can have in different methods of the same way.
This approach simply task management.
Polymorphism has two aspect: -
*. An object performs a task in multiple ways. This aspect is implemented with the help of overloading.
*. Different object perform a task in different ways. This aspect is implemented with the help of overriding. i.e., overloading are the means of implementing polymorphism.

Q.4 (b) Explain the need of package in Java. Explain accessibility rules for package. Also explain how members of a package are imported. Write java program to create your own package for finding area of different shapes

Answer: - Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

A Package can be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

Some of the existing packages in Java are −

java.lang − bundles the fundamental classes

java.io − classes for input , output functions are bundled in this package

Programmers can define their own packages to bundle group of classes/interfaces, etc. It is a good practice to group related classes implemented by you so that a programmer can easily determine that the classes, interfaces, enumerations, and annotations are related.

Since the package creates a new namespace there won't be any name conflicts with names in other packages. Using packages, it is easier to provide access control and it is also easier to locate the related classes.

Creating a Package
While creating a package, you should choose a name for the package and include a package statement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that you want to include in the package.

The package statement should 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.

If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package. For example: -

Package animals;

```
/* File name : MammalInt.java */

public class MammalInt implements Animal
{

  public void eat()
 {
    System.out.println("Mammal eats");
  }

  public void travel()
{
    System.out.println("Mammal travels");
  }

  public int noOfLegs()
{
    return 0;
  }

  public static void main(String args[])
{
    MammalInt m = new MammalInt();
    m.eat();
    m.travel();
  }
}
```


5. (a) What is abstract class? Explain need of abstract class with the help of an example.

Answer: - abstraction is the process of hiding the implementation details and showing only funcanality to the user.
There are two ways to achieve abstraction in java: -

*. Abstract class (0-100%)
*. Interface (100%)

Abstract class: - a class p.c declared as abstract is known as abstract class. It need to be extended and its can be instanceiated.
Abstract method that is declared as abstract and does not have implementation then it is known as abstract class.

For example: -

```
Abstract class shape
{
Abstract void show ();   // abstract method
}
Class rectangle extends shape
{
Void show ()
{
System.out.println ("this is rectangle class");
}
Class circle extend shape
```

```
{
Void show()
{
System.out.println (“this is circle class”);
}
}
class abstract test
{
Public static void main (string args[])
{
Rectangle obj = new rectangle ();
Obj. show ();
Circle obj1 = new circle ();
Obj1.show ();
}
}
```

Q.5 (b) what is an exception? Explain haw an exceptions are handled in Java. Write a java program to handle different arithmetic exceptions while managing a medical store billing process.

Answer: - An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.
when an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception.

After a method throws an exception, the runtime system attempts to find something to handle it. The set of possible "something" to handle the exception is the ordered list of methods that had been called to get to the method where the error occurred. The list of methods is known as the call stack (see the next figure).

The runtime system searches the call stack for a method that contains a block of code that can handle the exception. This block of code is called an exception handler.


```
import java.util.Scanner;

class Billing{
   String name;
  int Price,total,Quantity,d;
   public void input(){
       Scanner rd=new Scanner(System.in);
       System.out.println("enter the name of medicine");
       name=rd.next();
       System.out.println("enter the price of medicine");
       Price=rd.nextInt();
       System.out.println("enter the Quantity of medicine");
       Quantity=rd.nextInt();
       System.out.println("enter the Discount in percent");
       d=rd.nextInt();
   }
```

```
    void process()
    {

        try{

        total=Price*Quantity;
        total=total/d;
        output();
        }
        catch(ArithmeticException e){
          System.out.println("Arithmetic Exception");
      }

    }
    void output(){
        System.out.println("Your medicine "+name);
        System.out.println("Price "+Price);
        System.out.println("Quantity "+Quantity);
        System.out.println("Discount "+d+"%");
        System.out.println("total ammount "+total);
    }

}


public class ProgramRun {


    public static void main(String[] args) {
     Billing obj=new Billing();

     obj.input();

     obj.process();



    }

}
```

```
Output - JavaApplication4 (run)
enter the name of medicine
crocin
enter the price of medicine
10
enter the Quantity of medicine
34
enter the Discount in percent
10
Your medicine crocin
Price 10
Quantity 34
Discount 10%
total ammount 34
BUILD SUCCESSFUL (total time: 14 seconds)
```

6 (a) What is I/O stream in java? Write a program in java to create a file and count the number of words in it.

Answer: - The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc.

Stream
A stream can be defined as a sequence of data. There are two kinds of Streams −

Input Stream − The Input Stream is used to read data from a source.

Output Stream − The Output Stream is used for writing data to a destination.


```
public class Count Words
{
   public static void main (String[] args)
   {

         System.out.println("Simple Java Word Count Program");

         String str1 = "Today is Holiday Day";

         int word Count = 1;

         for (int i = 0; i < str1.length(); i++)
         {
            if (str1.charAt(i) == ' ')
            {
               wordCount++;
            }
         }

         System.out.println("word :"+str1+": Word count is = " + wordCount);
   }
}
```

```
Output - JavaApplication5 (run)
run:
Simple Java Word Count Program
word :Today is Holdiay Day Word count is = 4
BUILD SUCCESSFUL (total time: 0 seconds)
31:48 INS
```

Q.6 (b) Create an Applet program to display details of a quiz competition. Insert a background image in this program.

Answer: -

```
 import java.applet.*;
import java.awt.*;
import java.net.*;
import java.io.IOException.*;

public class BackgroundApplet extends Applet {
    Image backGround;

    public void init() {

        // set the size of the applet to the size of the background image.
        // Resizing the applet may cause distortion of the image.
        setSize(300, 300);

        // Set the image name to the background you want. Assumes the image
        // is in the same directory as the class file is
        backGround = getImage(getCodeBase(), "save.GIF");
        BackGroundPanel bgp = new BackGroundPanel();
        bgp.setLayout(new FlowLayout());
        bgp.setBackGroundImage(backGround);

        // Add the components you want in the Applet to the Panel
        bgp.add(new TextField("The competition is open to Macquarie University undergraduate  and
co-curricular students only"));
bgp.add(new TextField("Only photos entered in the correct session timeframe for the unit will be
accepted"));
bgp.add(new TextField("Only photos entered through the online entry form will be accepted in the
competition.");
bgp.add(new TextField("Entries must be taken during your PACE activity in one of the sessions from
March 2016 - March 2017.");


        // set the layout of the applet to Border Layout
        setLayout(new BorderLayout());

        // now adding the panel, adds to the center
        // (by default in Border Layout) of the applet
        add(bgp);
```

```
    }
}

class BackGroundPanel extends Panel {
    Image backGround;

    BackGroundPanel() {
        super();
    }

    public void paint(Graphics g) {

        // get the size of this panel (which is the size of the applet),
        // and draw the image
        g.drawImage(getBackGroundImage(), 0, 0,
            (int)getBounds().getWidth(), (int)getBounds().getHeight(), this);
    }

    public void setBackGroundImage(Image backGround) {
        this.backGround = backGround;
    }

    private Image getBackGroundImage() {
        return backGround;
    }
}
```

Q.6 (c) Write and explain different constructors of String class.

Answer: - Strings, which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects.
The Java platform provides the String class to create and manipulate strings.
By the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.
Java String is a powerful concept because everything is treated as a string if you submit any form in window based, web based or mobile application.

Class constructors: -

1. String()

This initializes a newly created String object so that it represents an empty character sequence.

2
String(byte[] bytes)

This constructs a new String by decoding the specified array of bytes using the platform's default char set.

3
String(byte[] bytes, Charset charset)

This constructs a new String by decoding the specified array of bytes using the specified charset.

4
String(byte[] bytes, int offset, int length)

This constructs a new String by decoding the specified subarray of bytes using the platform's default charset

5
String(byte[] bytes, int offset, int length, Charset charset)

This constructs a new String by decoding the specified subarray of bytes using the specified charset.

6
String(byte[] bytes, int offset, int length, String charsetName)

This constructs a new String by decoding the specified subarray of bytes using the specified charset.

7
String(byte[] bytes, String charsetName)

This constructs a new String by decoding the specified array of bytes using the specified charset.

8
String(char[] value)

This allocates a new String so that it represents the sequence of characters currently contained in the character array argument.

9
String(char[] value, int offset, int count)

This allocates a new String that contains characters from a subarray of the character array argument.

10
String(int[] codePoints, int offset, int count)

This allocates a new String that contains characters from a subarray of the Unicode code point array argument.

11
String(String original)

This initializes a newly created String object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

12
String(StringBuffer buffer)

This allocates a new string that contains the sequence of characters currently contained in the string buffer argument.

13
String(StringBuilder builder)

This allocates a new string that contains the sequence of characters currently contained in the string builder argument.

7. (A) what is layout manager? Explain different layouts available in java for GUI programming. What is default layout of an applet? Explain how to set the layout of an applet.

Answer: - Layout Manager is an interface that is implemented by all the classes of layout managers. There are following classes that represent the layout managers: java.awt.BorderLayout. java.awt.FlowLayout. java.awt.GridLayout

There are following classes that represent the layout managers:

***. Border Layout:**

The Border Layout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window.

Grid Layout: -

The Grid Layout is used to arrange the components in rectangular grid. One component is displayed in each rectangle

Flow Layout: -

The Flow Layout is used to arrange the components in a line, one after another (in a flow). It is the default layout of applet or panel.

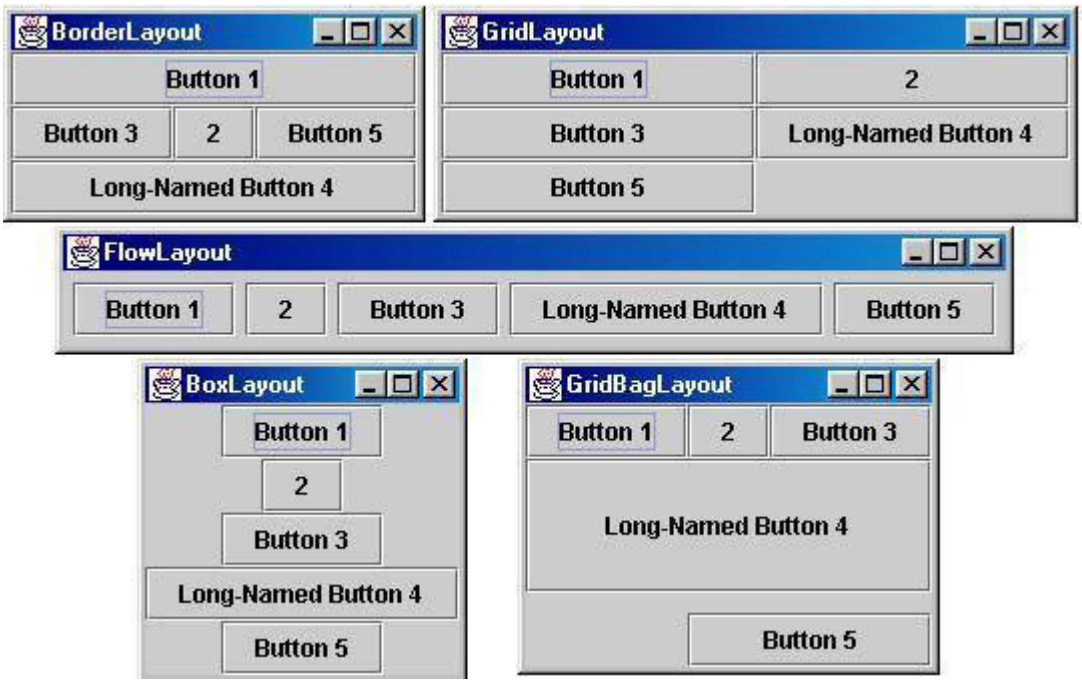
Box Layout class:

The Box Layout is used to arrange the components either vertically or horizontally. For this purpose, Box Layout provides four constants.

Card Layout class

The Card Layout class manages the components in such a manner that only one component is visible at a time. It treats each component as a card that is why it is known as Card Layout.

Layout Managers are provided to arrange GUI components on a container for presentation purposes. This allows the programmer to concentrate on the basic "look and feel" and lets the layout managers process most of the layout details.



Q.7 (b) what is multithreading? Explain how threads are synchronized in java.

Answer: - 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 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. Following is the general form of the synchronized statement –

```
Synchronized (object identifier)
{
  // Access shared variables and other shared resources
}
```

Here, the object identifier is a reference to an object whose lock associates with the monitor that the synchronized statement represents.

Q.7 (c) Explain the need of JDBC? Explain steps involved in connecting a databases using JDBC.

Answer: -

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

Making a connection to a database.

Creating SQL or MySQL statements.

Executing SQL or MySQL queries in the database.

Viewing & Modifying the resulting records.

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as −

Java Applications

Java Applets

Java Servlets

Java ServerPages (JSPs)

Enterprise JavaBeans (EJBs).

All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

8.(a) What is socket? Explain stream sockets and datagram sockets.

Answer: - Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.
To a programmer, a socket looks and behaves much like a low-level file descriptor. This is because commands such as read() and write() work with sockets in the same way they do with files and pipes.

Stream Sockets − Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order − "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.

Datagram Sockets − Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets − you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).

Q.8 (b) what is RMI? Explain RMI architecture.

Answer: - The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects stub and skeleton.

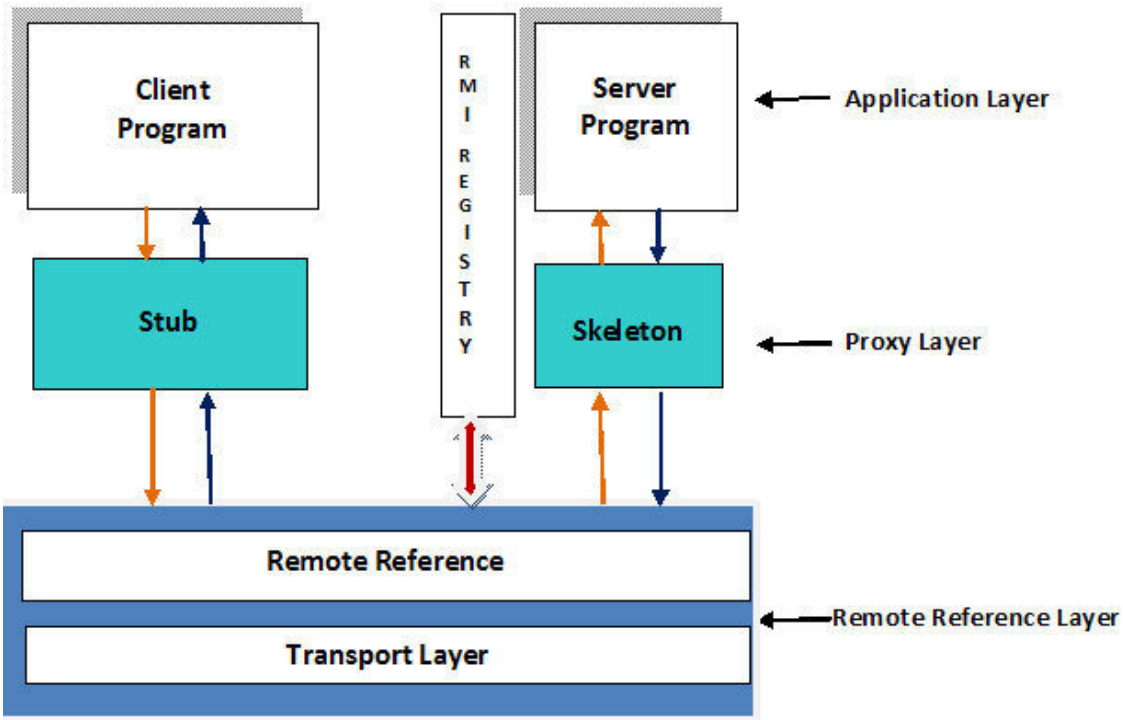


Fig: RMI Architecture

As the figure illustrates, there comes three layers in the RMI Architecture – Application layer, Proxy layer and Remote reference layer (Transport layer is part of Remote reference layer).

1. Application Layer

This layer is nothing but the actual systems (client and server) involved in communication. A client Java program communicates with the other Java program on the server side. RMI is nothing but a communication between two JVMs placed on different systems.

2. Proxy Layer

The proxy layer consists of proxies (named as stub and skeleton by designers) for client and server. Stub is client side proxy and Skeleton is server side proxy. Stub and skeleton, as many people confuse, are not separate systems but they are after all programs placed on client and server. The stub and skeleton are not hard coded by the Programmer but they are generated by RMI compiler (this is shown in execution part later). Stub is placed on client side and skeleton is placed on server side. The client server communication goes through these proxies. Client sends its request of method invocation (to be executed on remote server) to stub. Stub inturn sends the request to skeleton. Skeleton passes the request to the server program. Server executes the method and sends the return value to the skeleton (to route to client). Skeleton sends to stub and stub to client program.

3. Remote Reference Layer

Proxies are implicitly connected to RMI mechanism through Remote reference layer, the layer responsible for object communication and transfer of objects between client and server. It is

responsible for dealing with semantics of remote invocations and implementation – specific tasks with remote objects. In this layer, actual implementation of communication protocols is handled.

Q.8 (c) What is servlet ? Explain servlet life cycle.

Answer: - Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.
A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet

The servlet is initialized by calling the init () method.

The servlet calls service() method to process a client's request.

The servlet is terminated by calling the destroy() method.

Finally, servlet is garbage collected by the garbage collector of the JVM.

The init() method :
The init method is designed to be called only once. It is called when the servlet is first created, and not called again for each user request. So, it is used for one-time initializations, just as with the init method of applets.

```
public void init() throws ServletException
{
  // Initialization code...
}
```

The service() method :

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

```
public void service(ServletRequest request,
                    ServletResponse response)
      throws ServletException, IOException
{
}
```

The destroy() method :

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

```
public void destroy()
 {
   // Finalization code...
  }
```