F.M: 60
P.M: 24

Advanced Java Programming - 2071

1) What is multithreading ? Why is it important to develop computer programs ? Discuss life cycle of thread in detail. (2+2+6)

→ Multithreading is a technique that allows a program or a process to execute many tasks concurrently. It allows a process to run its task in parallel mode on a single processor system. Multithreaded software treats each process as a separate program. It is the specialized form of multitasking.

A multithreaded program involves multiple threads of control in a single program. When a program contains multiple threads then the CPU can switch between the two threads to execute them at the same time. We can make the maximum use of CPU by keeping the idle time to a minimum. In order to avoid issues like deadlocks and starvation also, it is important to develop a multithreaded computer programs.

The life cycle of the thread in java is controlled by JVM. A thread can be in one of the five states which are as follows:

a) New

The thread is in new state if we create an instance of Thread class but before the invocation of start() method.

b) Runnable

The thread is in runnable state after invocation of start() method.

c) Running

The thread is in running state if the thread scheduler has selected it.

d) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

e) Terminated

A thread is in terminated or dead state when its run() method exits.

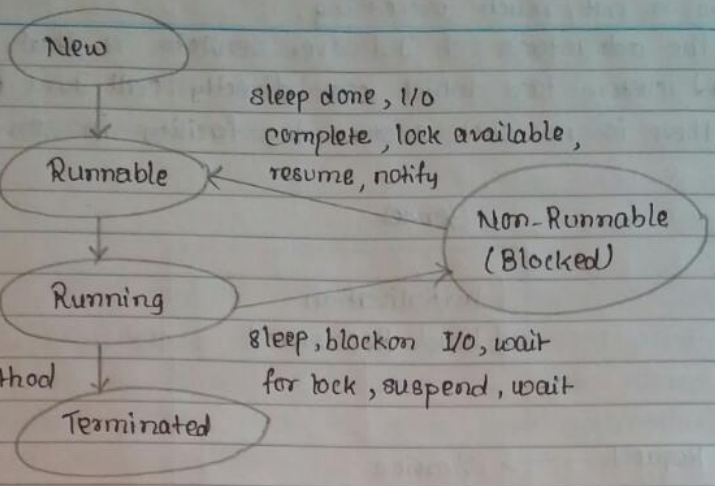


fig: The java thread states

3) What is servlet? Differentiate it with JSP. Discuss life cycle of servlet in detail. (2+3+5)

→ Servlets are small programs that execute on the server side of a web connection. It is a java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model. The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets.

The differences between JSP and servlet are as follows:

i) Servlet is a server-side program where as JSP is a webpage scripting language.

ii) It's easier to code in JSP but a bit difficult to write in servlet because it involves http request and response.

iii) JSP run slower compared to servlet as it takes compilation time to convert into java servlets. Servlets run faster than JSP.

iv) Servlets are best for use when there is more processing and manipulation involved where as JSP is preferred when

there is not much processing.

v) The advantage of JSP over servlets is that we can build custom tags which can directly call Java beans where as there is no such custom tag facility in servlets.

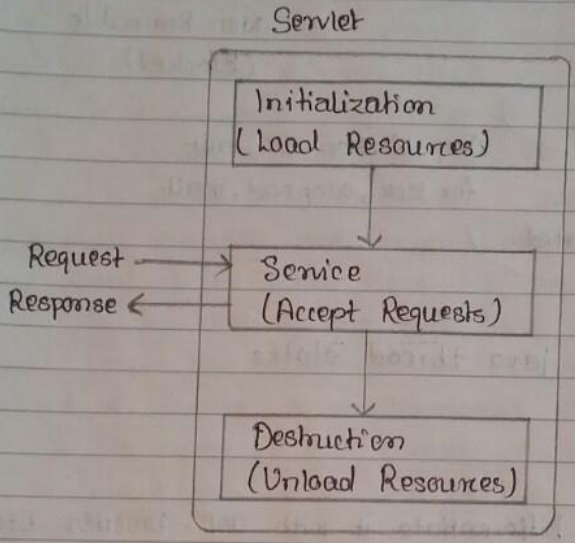


fig: Diagram of the Servlet Life Cycle

Servlets follow a three-phase life: initialization, service and destruction.

i) Initialization

is the first phase of the servlet life cycle and represents the creation and initialization of resources the servlet may need to service requests. All servlets must implement the javax.servlet.Servlet interface which defines the init() method.

ii) Service

This phase represents all interactions with requests until the servlet is destroyed. The service() method is invoked once per a request and is responsible for generating the response to that request.

iii) Destruction
represents when a servlet is being removed from use by a container. The Servlet interface defines the destroy() method to correspond to the destruction life cycle phase.

4) What is JDBC ? Discuss different driver types of JDBC. (1+4)
→ JDBC (Java Database Connectivity) is an API that can access any kind of tabular data, especially data stored in a relational database. JDBC helps to write java applications that manage these three programming activities
- connect to a database source like a database
- send queries and update statements to the database
- retrieve and process the result received from the database in answer to our query.

JDBC drivers are classified into the following types:
- A type 1 driver : translates JDBC to ODBC and relies on an ODBC driver to communicate with the database.
- A type 2 driver : communicates with the client API of a database. When we use such a driver, we must install some platform specific code in addition to java library.
- A type 3 driver : It is a pure java client library that uses a database-independent protocol to communicate database requests to a server component.
- A type 4 driver - is a pure java library that translates JDBC requests directly to a database specific protocol.

Most database vendors supply either a type 3 or type 4 driver with their database.

2) Write a program using swing components to find simple interest. Use text fields for inputs and output. Your program should display the result when the user presses a button. (10)

→

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SimpleInterest extends JFrame implements
                                    ActionListener
{
 JTextfield t1,t2,t3,t4;
 JLabel l1, l2, l3, l4;
 JButton b1;
 SimpleInterest()
  {
    setTitle(" SIMPLE INTEREST ");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    l1 = new JLabel ("Principal");
    l2 = new JLabel (" Time");
    l3 = new JLabel ("Rate");
    l4 = new JLabel ("Simple Interest");

    t1 = new JTextfield (10);
    t2 = new JTextfield (10);
    t3 = new JTextfield (10);
    t4 = new JTextfield (10);

    b1 = new JButton (" Compute");
    b1. addActionListener (this);

  setLayout (new FlowLayout(FlowLayout.LEFT ,150,10));
```

```
add(l1);
add(l2);
add(l3);
add(l4);

add(t1);
add(t2);
add(t3);
add(t4);

add(b1);

setVisible(true);
}

public void actionPerformed(ActionEvent ae)
{
  int p,t,r,SI;
  p = Integer.parseInt(t1.getText());
  t = Integer.parseInt(t2.getText());
  r = Integer.parseInt(t3.getText());
  SI = (p*t*r)/100;
  t4.setText(String.valueOf(SI));
  }

public static void main(String a[])
{
  new SimpleInterest();
 }
}
```

4) How do you achieve multiple inheritance in java ? Discuss. 6

→ Multiple inheritance is the ability of a single class to inherit from multiple classes. Java does not have this capability. Multiple inheritance can cause the diamond problem.

What a Java class does have is the ability to implement multiple interfaces - which is considered a reasonable substitute for multiple inheritance, but without the added complexity.

Example :

```
interface X
{
  public void myMethod();
}
interface Y
{
  public void myMethod();
}

class Demo implements X, Y
{
  public void myMethod()
  {
    System.out.println("Multiple Inheritance example using
    interfaces");
  }
}
```

As we can see that the class implemented two interfaces. A class can implement any number of interfaces. In this case, there is no ambiguity even though both the interfaces are having same method. Because methods in an abstract interface are always abstract by default, which doesn't let them to give their implementation in interface itself.

6) Explain the importance of exception handling with suitable example.

→ The basic purpose of exception handling is to maintain the normal flow of the application irrespective of errors/exceptions which might occur during execution. Even if some unexpected error occurs, the exception framework provides separate execution path to avoid application failure. So for any robust java application the exception handling framework must be designed properly.

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.

Example:

```
import java.util.Scanner;
class Division {
  public static void main (String args[]) {
    int a, b, result;

    Scanner input = new Scanner (System.in);
    System.out.println ("Input two integers");

    a = input.nextInt();
    b = input.nextInt();

    try {                          // try block
    result = a/b;
    System.out.println ("Result = " + result);
  }
```

```
// catch block
catch (ArithmeticException e) {
    System.out.println("Exception caught: Division by zero.");
}
}
}
```

13) Write a simple JSP program to display "Tribhuwan University" 10 times. (5)

→

```
<html>
<head>
<title> Simple JSP Program </title>
</head>
<body>
<%
    for(i=1;i<=10;i++)
        out.println(i+". Tribhuwan University <br>");
%>
</body>
</html>
```

Output: SimpleJSPProgram.jsp

1. Tribhuwan University
2. Tribhuwan University
3. Tribhuwan University
4. Tribhuwan University
5. Tribhuwan University
6. Tribhuwan University
7. Tribhuwan University
8. Tribhuwan University
9. Tribhuwan University
10. Tribhuwan University

8) Write a simple java program to read from and write to files.

→

```
import java.io.*;

public class FileReadWrite
{
  public static void main(String a[]) throws IOException
  {
    FileWriter writer = new FileWriter("D:/hello.txt
    writer.write("This\n is\n an\n example");
    writer.close();
    FileReader fr= new FileReader("D:/hello.txt");
    char a[] = new char[50];
    fr.read(a);
    for(char c:a)
    System.out.println(c);
    fr.close();
    }
  }
```

Output :

This
is
an
example

9) What is TCP socket ? Differentiate it with UDP socket. (2+3)

→ A TCP socket is that Transmission Control Protocol endpoint suite of communication flow between two programs running over a network. It is used to create a new communication endpoint.

The differences between TCP socket and UDP socket can be illustrated below:

i) Connection oriented vs Connection less

TCP socket is a connection-oriented where connection is established between client and server, where as UDP socket is connection less.

ii) Reliability

TCP provides delivery guarantee, which means a message sent using TCP protocol is guaranteed to be delivered to client. UDP is unreliable.

iii) Ordering

TCP also guarantees order of message. Message will be delivered to client in the same order, server has sent than UDP.

iv) Data Boundary

TCP does not preserve data boundary, UDP does. On UDP, packets are sent individually and are checked for integrity only if they arrived.

v) Speed

TCP is slow and UDP is fast since UDP does not guarantee reliability.

vi) Heavy weight vs Light weight

TCP is considered as heavy weight as compared to light weight UDP protocol.

vii) Header size

TCP has bigger header than UDP. Usual header size of a TCP packet is 20 bytes which is more double of 8 bytes of UDP.

viii) Congestion or Flow Control

TCP does flow control where as UDP does not have an option for flow control.

A) Discuss group layout with suitable example. (5)

→ GroupLayout is a layout manager that was developed for use by GUI builder tools. GroupLayout works with the horizontal and vertical layouts separately. The layout is defined for each dimension independently. However, each component needs to be defined twice in layout, we do not need to worry about the vertical dimension when defining horizontal layout and vice versa.

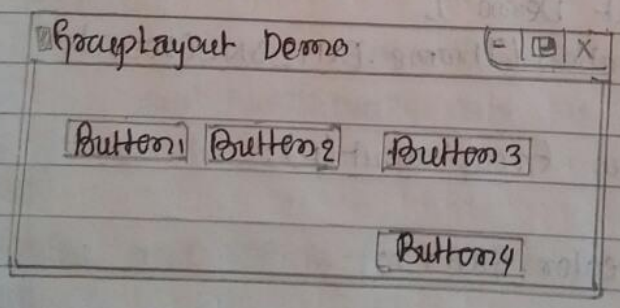
GroupLayout uses two types of arrangements - sequential and parallel.

Example:

```
import javax.swing.*;
import java.awt.*;
public class GroupLayoutDemo extends JFrame {
JButton b1, b2, b3, b4;
JPanel p;
GroupLayoutDemo() {
 setSize(400, 250);
 setTitle("Grouplayout Demo");
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 p = new JPanel();
 Grouplayout l = new Grouplayout(p);
 p.setLayout(l);
 p.setBackground(Color.GRAY);
 add(p);
 l.setAutoCreateGaps(true);
 l.setAutoCreateContainerGaps(true);
 b1 = new JButton("Button1");
 b2 = new JButton("Button2");
 b3 = new JButton("Button3");
 b4 = new JButton("Button4");
 l.setHorizontalGroup(l.createSequentialGroup().addComponent
                                                (b1)
                                              .addComponent
                                              (b2)
```

```
        .addGroup(l.createParallelGroup());
        .addComponent(b3);
        .addComponent(b4));
        l.setVerticalGroup(l.createSequenceGroup()
       {.addComponent(b1)
        .addGroup(l.createParallelGroup
        .addComponent(b2)
        .addComponent(b3))
        .addComponent(b4));
         setVisible(true);
       }
public static void main(String a[])
 {
  GroupLayoutDemo frame = new GroupLayoutDemo();
 }
}
```

Output:



10) Discuss any five event classes in java. (5)

→ The different event classes in java are:

Event Class	Description
ActionEvent	Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
AdjustmentEvent	Generated when a scroll bar is manipulated.
ComponentEvent	Generated when a component is hidden, moved, resized, or becomes visible.
ContainerEvent	Generated when a component is added to or removed from a container.
FocusEvent	Generated when a component gains or loses keyboard focus.

11) What is java beans? Differentiate it with java classes.

→ JavaBeans is an object-oriented programming interface from Sun Microsystems that lets you build re-usuable applications or program building blocks called components that can be deployed in a network on any major operating system platform.

- Java beans follow the Bean conventions:

i) Default, no-arg ctor,

ii) Serializable

iii) getX()/setX() or is isX()/setX() naming convention for read/write access to private data member X,

iv) can use java bean PropertyChangeEvent to notify interested parts when values change.

v) Can use java bean PropertyChangeListener to register for notification when a particular property changes.

- Java bean is also a class which normally does not have processing logic.

A java class is a class which normally has properties and methods through which these properties are modified. The class

can perform some functions specific to the data in the class.
Example: Creating a simple JavaBean

```
import java.awt.*;
import java.io.Serializable;

public class SimpleBean extends Canvas implements Serializable{
// Constructor sets inherited properties
 public SimpleBean(){
  setSize(60,40);
  setBackground(Color.red);
 }
}
```

12) What is RMI? Differentiate it with CORBA.

→ Remote Method Invocation (RMI) is a java API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage collection.

It allows Java developers to invoke object methods, and have them execute on remote JVMs.

i) RMI is a Java-specific technology. CORBA has implementations for many languages. You can use CORBA to share objects between programs written in different languages.

ii) CORBA uses IDL (Interface Definition Language) to separate interface from implementation. RMI just uses Java interfaces.

iii) Because CORBA is not tied to a particular language, the data types do not always map exactly to the types used by our programming language.

iv) RMI programs can download new classes from remote JVMs. CORBA doesn't have this code sharing mechanism.

2) Write a program using swing component to multiply two numbers. Use text fields for input and output. Your programs should display the result when the user press a button.

→

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class MultiplyEvent extends JFrame implements ActionListener{
 JTextField t1,t2,t3;
 JLabel l1,l2,l3;
 JButton b1;
 MultiplyEvent()
  {
    setTitle("MULTIPLY TWO NUMBERS");
    setSize(400,300);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    l1 = new JLabel("First Value:");
    l2 = new JLabel("Second Value:");
    l3 = new JLabel("Third Value:");
    t1 = new JTextField(10);
    t2 = new JTextField(10);
    t3 = new JTextField(10);

    b1 = new JButton("MULTIPLY");
    b1.addActionListener(this);

    setLayout(new FlowLayout(FlowLayout.LEFT,150,10));
    add(l1);
    add(l2);
    add(l3);
    add(t1);
```

```
    add(t3);  -> add(b1);
    setVisible(true);
  }

public void actionPerformed (ActionEvent ae)
 {
  int x,y,z;
  x = Integer.parseFloat(t1.getText());
  y = Integer.parseFloat(t2.getText());
  z = x*y;
  t3.setText(String.valueOf(z));
 }

public static void main(String a[])
 {
   new MultiplyEvent();
  }
}
```

3) How can you use RMI to develop a program that runs in different machine? Discuss with suitable example.

→ There are six steps to write the RMI program.

i) Create the remote interface

We extend the Remote interface and declare the RemoteException with all the methods of the remote interface.

```
import java.rmi.*;
public interface Adder extends Remote
 {
  public int add(int x, int y) throws RemoteException;
 }
```

ii) Provide the implementation of the Remote Interface

For providing the implementation of the Remote inter

need to either extend the UnicastRemoteObject class, or use the exportObject() method of the UnicastRemoteObject class

```
import java.rmi.*;
import java.rmi.server.*;

public class AdderRemote extends UnicastRemoteObject
implements Adder
{
 AdderRemote() throws RemoteException
 {
  super();
 }
 public int add(int x, int y)
 {
  return x+y;
 }
}
```

iii) Create the stub and skeleton objects using the rmic Tool
The rmic tool invokes the RMI compiler and creates stub and skeleton objects.

```
rmic AdderRemote
```

iv) Start the Registry Service by the rmiregistry Tool
The port number (default) that we use is 5000.

```
rmiregistry 5000
```

v) Create the and Run the Server Application
Now rmi services need to be hosted in a server process.

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer
{
```

```
try {
    Adder adder object = new AdderRemote();
    Naming.rebind("rmi://localhost:5000/rmiserver", adderobject);
  }
  catch (Exception e)
  {
    System.out.println(e);
  }
}
```

vi) Create and run the client application

At the client we are getting the adderobject object by the lookup() method of the Naming class and invoking the method on this object.

```
import java.rmi.*;
public class MyClient{
public static void main(String args[]) throws Exception
{
    Adder a = (Adder)Naming.lookup("rmi://localhost:5000/as");
    int s = a.add(34,4);
    System.out.println("sum = "+s);
  }
}
```

a) How do you execute SQL queries in JDBC ?

→ The Statement object's executeQuery method is used to submit a query that selects all the author information from table Authors. This method returns an object that implements interface ResultSet and contains the query results. The ResultSet methods enable the program to manipulate the query result.

To execute a query, call an execute method from Statement such as the following :

. execute : Returns true if the first object that the query returns is a ResultSet object. Use this method if the query could return one or more ResultSet objects. Retrieve the ResultSet objects returned from the query by repeatedly calling Statement.getResultSet.

. executeQuery : Returns one ResultSet object

. executeUpdate : Returns an integer representing the number of rows affected by the SQL statement.

for Example,

CoffeesTables.viewTable executed a Statement object with the following code :

```
ResultSet rs = stmt.executeQuery(query);
```

8) Discuss borderlayout with suitable example.

→ A BorderLayout has five areas : north, south, east, and west and center.

Example :

```
setLayout(new BorderLayout());
setFont(new Font("Arial", Font.PLAIN, 14));

add(new JButton("North"), BorderLayout.NORTH);
add(new JButton("South"), BorderLayout.SOUTH);
add(new JButton("East"), BorderLayout.EAST);
add(new JButton("West"), BorderLayout.West);
add(new JButton("Center"), BorderLayout.Center);
```

To add a gap between the components you can use the constructor :

```
public BorderLayout(int horizontalGap, int verticalGap)
```

10) Discuss any 5 exception classes in java.

→ Java defines several other types of exceptions that relate to its various class libraries. Following is the list of Java Unchecked RuntimeException.

Exception	Description
1) ArithmeticException	Arithmetic error, such as divide by zero
2) ArrayIndexOutOfBounds-Exception	Array index is out-of-bounds
3) ArrayStoreException	Assignment to an array element of an incompatible type.
4) ClassCastException	Invalid cast
5) IndexOutOfBoundsException	Some type of index is out-of-bounds.
6) NullPointerException	Invalid use of a null reference.
7) SecurityException	Attempt to violate security
8) StringIndexOutOfBounds-Exception	Attempt to index outside the bounds of a string
9) InterruptedException	One thread has been interrupted by another thread.

11) Discuss the role of event listeners to handle events with suitable example.

→ Events represent changes or actions that occur within objects. For example, modification of class data, execution of a method, destruction of an object.

An object that would like to be notified of and respond to an event is an event listener. Some of the event listeners and their roles are discussed below:

i) ActionListener

are interfaces that extend java.util.EventListener. When the user

clicks on a JButton, the actionPerformed() method is invoked after creating an ActionEvent.

ii) EventListener

doesn't define any methods but the interface defines one or more methods that an event source may invoke when a particular type of event occurs.

iii) MouseListener

defines methods including button press events and button release events.

10)

→ 5 exception classes in java

i) Arithmetic Exception

This is a built-in-class present in java.lang package. This exception occurs when an integer is divided by zero.

```
class ExceptionDemo1 {
  public static void main(String args[]) {
    try {
      int num1=30, num2=0;
      int output = num1/num2;
      System.out.println("Result =");
    }
    catch(ArithmeticException e) {
      System.out.println("Arithmetic Exception");
    }
  }
}
```

ii) ArrayIndexOutOfBounds

class: java.lang.ArrayIndexOutOfBounds

This exception occurs when the referenced element does not exi in the array. For e.g. If array is having only 5 elements and we are trying to display 7th element then it would throw this

exception.

```
class ExceptionDemo2 {
  public static void main (String args[]) {
    try {
      int a[] = new int[10];
      a[11] = 9;
    }
    catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Array Index Out of Bounds Exception");
    }
  }
}
```

iii) NumberFormatException

The object of the ~~ab~~.java.lang.NumberFormatException class gets created whenever a string is parsed to any numeric variable. for e.g. The statement int num = Integer.parseInt("XYZ"); would throw NumberFormatException because String "XYZ" cannot be parse to int.

```
class ExceptionDemo3 {
  public static void main(String args[]) {
    try {
      int num = Integer.parseInt("XYZ");
      System.out.println(num);
    }
    catch (NumberFormatException e) {
      System.out.println("Number format is wrong");
    }
  }
}
```

Output : Number format is wrong.

iv) StringIndexOutOfBound Exception

- An object of this class gets created whenever an index of is invoked of a string, which is not in the range.
- Each character of a string object is stored in a particular index starting from 0.
- To get a character present in a particular index of a string we can use a method charAt(int) of java.lang.String where int argument is the index.

```
try {
String str = "easysteps2buildwebsites"
System.out.println(str.length());
char c = str.charAt(0);
c = str.charAt(40);
System.out.println(c);
}
catch(StringIndexOutOfBoundsException e) {
System.out.println("StringIndexOutOfBoundsException !!");
}
}
```

v) NullPointer Exception

An object of this class gets created whenever a member is invoked with a "null" object. Example:

```
package beginnersbook.com
class Exception2
{
  try{
    String str = null;
    System.out.println(str.
    }
  catch (NullPointerException e) {
  System.out.println("NullPointerExcephion..");
  }
```

11>

→ An event listener is used to process events. For example a graphical component like a JButton or JTextfield are known as event sources. This means they can generate events - when a user clicks on the JButton or types text into the JTextfield. The event listeners job is to catch those events and do something with them.

1) ActionListener : It declares one method to receive action events. Action events are generated when button is pressed
void actionPerformed (ActionEvent ae)

2) AdjustmentListener : It declares one method to receive adjustment events. Adjustment events are generated when a scroll bar is manipulated.

3) ItemListener : It defineds one method to recognize when a check box or list item is clicked, when choice selection is made.
void itemStateChanged (ItemEvent ie)

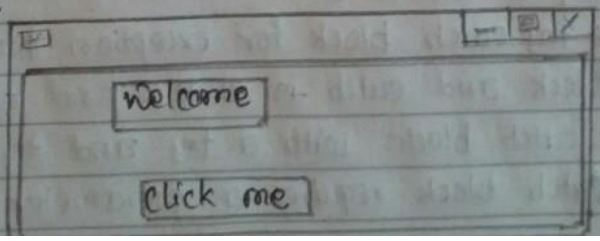
4) ContainerListener : It declares two methods to recognize when a component is added to or removed from container.
void componentAdded (ContainerEvent ce)
void componentRemoved (ContainerEvent ce)

5) FocusListener : It declares two methods to recognize when a component gains or loses keyboard focus.
void focusGained (FocusEvent fe)
void focusLost (FocusEvent fe)

Example :

```
import javax.swing.*;
import java.awt.event:
class EventDemo extends JFrame implements ActionListener
{
  JTextField tf;
  EventDemo()
  {
    tf = new JTextField(15);
    tf.setBounds(60,50,170,20);
    setLayout(null);
    JButton b = new JButton("Click me");
     b.setBounds(100,120,80,30);
      b.addActionListener(this);
      add(b);
      add(tf);
      setSize(300,300);
      setVisible(true);
    }
public void actionPerformed(ActionEvent e)
  {
    tf.setText("Welcome");
  }
public static void main(String args[])
  {
   new EventDemo();
  }
}
```

Output :



handle exception.

2069

1) What are exceptions ? Why is it important to handle exceptions ? Discuss different keywords that are used to handle exception. (2+2+6)

→ An exception is an abnormal condition that arises in a code sequence at run time. In other words, an exception is a run-time error.

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. For example: Suppose there are 10 statements in our program and there occurs an exception at statement 5, rest of the code will not be executed i.e. statement 6 to 10 will not run. If we perform exception handling, rest of the exception will be executed. Just to avoid program terminating automatically and

There are 5 keywords used in java exception handling.

i) throw - If any exception occurs, an exception object is getting created and then java runtime starts processing to handle them. Sometime we might want to generate exception explicitly in our code, for example in a user authentication program we should throw exception to client if the password is null. throw keyword is used to throw exception to the runtime to handle it. Syntax : throw ThrowableInstance;

ii) throws - When we are throwing any exception in a method and not handling it, then we need to use throws keyword to let caller program know the exceptions that might be thrown by the method. Using throws keyword, we can provide multiple exceptions in the throws clause and can be used with main() method also. Syntax : type method-name(parameter-list) throws exception-list { // body of method }

iii) try-catch - We use try-catch block for exception handling. try is the start of the block and catch is at the end of try block. We can have multiple catch blocks with a try and try-catch block