Atma Ram Sanatan Dharma College

University of Delhi

Programming in Java Assignments

Submitted By –

Anshul Verma

College Roll No – 19/78065

B.Sc. (Hons) Computer Science

Submitted To –

Ms. Parul Jain

Department of Computer Science

Index

Assignment #1

Ques1: What will be output of the below program?

a)
```
class A
{
  public A() {
        system.out.println ("Class A constructor");
  }
}
class B extends A
{
  public B() {
     system.out.println ("Class B constructor"); }
}
class C extends B
{
  public C() {
     system.out.println ("Class C constructor");
  }
}
public MainClass
{
 public static vaid main (string[] args)
  {
     C c = new C();
  }
}
```

Output:

```
Class A constructor
Class B constructor
Class C constructor
```

b)
```
class A
{
  String s= "Class A";
}
class B extends A
{
  String s = "Class B";
  {
   System.out.println (super.S);
  }
}
class C extends B
{
  String s= "Class C";
  {
   System.out.println ( super.s);
} }
public class MainClass
{
  public static void main (String[] args)
  {
     C c= new C();
     System.out.println (c.s);
  }
}
```

Output:
```
Class A
Class B
Class C
```

c)
```
classCommLine {
     public static void main (String[] args){
        for (int i=0 ; i<args.length; i++)
            system.out.prinln ("args[" +i +"]:" + args[i]);
    }
}
```

<u>Output:</u> Let us assume that we ran the class as:

```
java CommLine abc efg hij klm
```

∴ output:

```
args[0] : abc
args[1] : efg
args[2]: hij
args[3]: klm
```

<u>Ques:2</u> Can abstract class have constructors in java?

<u>Ans:</u> Yes, an abstract class can have constructors in Java. This is true for all classes and it also applies to abstract class.

<u>Ques:3</u> Create an abstract class "Parent" with a method 'message', It has two subclass each having a method with same name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message" by creating an object for each subclass.

<u>Ans:</u>

```
abstract class Parent {
    abstract void message()~~{~~;
}
class SubClass1 extends Parent {
  void message() {
    System.out.println ("This is first subclass");
  }
}
```

```
class SubClass2 Extends Parent {
  void message(){
     System.out.println("This is second subclass");
  }
}
public class Main{
  public static void main(String[] args){
          SubClass1  obj1 = new SubClass1;
          SubClass2  obj2 = new SubClass2;
          ob1.message();
          ob2.message();
  }
}
```

//output:
```
This is first subclass
This is second subclass
```

Ques:4 An abstract class has a constructor which prints "This is constructor of abstract class", an abstract method named 'a_method' and a non-abstract method which prints "This is normal method of abstract class". A class 'SubClass' inherits the abstract class and has a method named and the non-abstract method. (Analyse the result).

Ans:
```
abstract class AbstractClass {
      AbstractClass(){
          System.out.println("This is constructor of
            abstract class");
      }
      abstract void a_method();
      public void print(){
          System.out.println("This is normal method of
                    abstract class");
      }
}
```

```
class SubClass extends AbstractClass {
    void a_method(){
      System.out.println("This is abstract method");
    }
}
public class Main{
  public static void main(String[] args){
     SubClass obj = new SubClass();
     obj.a_method();
     obj.print();
  }
}
```

Output:

```
This is constructor of abstract class
This is abstract method
This is nomal method of abstract class.
```

Analysis: When the SubClass is instantiated, the constructor of its Parent (Abstract Class) is called which prints "This is constructor of abstract class".

This also shows that SubClass is inherited from AbstractClass.

Definition of a_method() in AbstractClass is provided whose body is defined in the inherited "SubClass". This prints "This is abstract method".

Definition of print() method in the AbstractClass prints the line "This is the normal method of abstract class".

Ques:5 Write a java code to find whether a number is prime or not where number is accepted from command line.

```
public class CheckPrime {
  static boolean isPrime (int n) {
     if (n<=1) return false;
     if (n<=3) return true;
     if (n%2==0 || n%3==0) return false;
     for (int i=5; i*i <=n; i++){
      if (n%i==0 || n%(i+2)==0)
        return false;
     }
     return true;
  }
  public static void main (String[] args){
      int a = Integer.parseInt (args[0]);
      System.out.println( "It is "+ isPrime(a) +
     " that " + a+" is prime.");
  }
}
```

ASSIGNMENT #2

Ques: 1 What will be ouput of the below program?

a)
```
interface A
{
  vaid myMethod();
}
class B
{
  public void myMethod()
  {
   system.out.printn("My method");
  }
}
class C extends B implements A
{    }
class MainClass
{
  public static void main (String[] args)
  {
     A a = new C();
     a.myMethod();
  }
}
```

Output :→   My method

b)
```
interface P
{
  String p = "PPPP";
  String methodP();
}
interface Q extends P
{
  String q = "QQQQ";
  String methodQ();
}
```

```
class R implements P, Q
{
  public String methodP()
  {
    return q+p;
  }
  public String methodQ()
  {
    return p+q;
  }
}
public class MainClass{
  public static void main (String[] args)
  {
     R r = new R();
     System.out.println (r.methodP());
     System.out.println (r.methodQ());
   }
 }
```

Output:

```
QQQQPPPP
PPPPQQQQ
```

Ques:2 Create a class TwoDim which contains private members as x and y coordinates in package P1. Define the default constructor, a parameterized constructor and override toString method to display dimention z ~~as its~~ private member. Define the constructor the coordinates. Now reuse this class and in Package P2 create another class ThreeDim, adding a new dimentions as z as its private member. Define the constructors for the subclass and override toString() method in the Subclass also. Write appropriate methods to show $\overset{\text{dynamic}}{,}$ method dispatch. The main() function should be in package P.

↳

P1/TwoDim.java →

```
package P1;
public class TwoDim {
        private int x,y;
        public TwoDim(){
            this.x=0;
            this.y=0;
         }
        public TwoDim(int x, int y){
            this.x = x;
            this.y= y;
        }

        @Override
        public String toString(){
            return "Coordinate : ( "+ x + " , " +
                              y + " ) ";
  }
```

P2/ThreeDim.java →

```
package P2;        import P1.TwoDim;
public class ThreeDim extends TwoDim {
        private int z;
        public ThreeDim() {
                super(0,0);
                this.z = 0;
         }
        public ThreeDim(int x, int y , int z){
                super(x,y);
                this.z = z;
        }
        @Override
        public String toString() {
            return "Coordinate :(" + x + "," + y +
                         "," + z + ")";
        }
    }
P/Main.java →   package P;
                import P1.TwoDim;
                inport P2.ThreeDim;
           public class Main {
             public static void main (String[] args) {
                 TwoDim obj= new TwoDim(4,6);
                 System.out.println ( obj);
                 obj = new ThreeDim(3,6,9);
                 System.out.println (obj);
             }
           }
```

Output:

```
Coordinate: (4,6)
Coordinate: (3,6,9)
```

Ques:3 Define an abstract Class Shape in package P1. Inherit two more classes: Rectange in package P2 and Circle in package P3. Write a program to ask the user for the type of Shape and then using the concept of dynamic method dispatch, display the area of appropriate subclass. Also write appropriate methods to read the data. The main function should not be in any package.

Code: P1/Shape.java →

```
package P1;
public abstract class Shape {
        public abstract void getData();
        public abstract double area();
}
```

P2/Rectangle.java →

```
    package P2;
    import P1.Shape;
    import java.io.*;
    public class Rectangle extends Shape {
            private double len, bre;
            public vaid getData() {
              BufferedReader br = new BufferedReader (new
                        InputStreamReader (system.in));
              System.out.println ("Enter the length of
                                    Rectangle :");
              len = Double.parseDouble (br.readline());
              System.out.println ("Enter the breadth of
                                    Rectangle :");
              bre = Double.parseDouble (br.readLine());
            }
            public double area(){
                getData();
                return len * bre;
            }
    }
```

P3/Circle.java →

```
    package P3;
    import P1.Shape;
    import java.io.*;
    public class Circle extends Shape {
        private double radius;
        public vaid getData() {
          BufferedReader br = new BufferedReader( new
                    InputStreamReader (System.in));
          System.out.println (" Enter the radius ");
```

```
            radius = Double.parseDouble (br.readLine ());
        }
        public double area () {
                getData ();
                return  Math.PI *(radius * radius );
        }
    }
Main.java :→  import java.io.* ;
              import P1.Shape;
              import P2.Rectangle;
              import P3.Circle;
    public class Main {
          ~~static int~~
          ~~Buf~~ public static vaid main ( String[] args ) {
                BufferedReader br = new BufferedReader (new
                        InputStreamReader ( system.in ));
                System.out.println ( "Select a shape :
                    (1) Rectangle\n (2) Circle " ) ;
                System.out.println ("Enter Choice :");
                ~~int n~~ ;
                Shape ~~ref~~ ; shape ;
                switch ( Integer.parseInt (br.readLine()) ; {
                  case 1 :
                      shape = new Rectangle();
                      System.out.println ( " Area : " +
                                  shape.area()+ " sq units ");
                      break;
                  Case 2 :
                      shape = new Circle ( );
                      Sytstem.out.println ( "Area : " + shape.area()
                                     + " sq units");
            break;
        default:
            System.err.println("Invalid Option");
            break;
    }
}
```

Ques: 4 Define an interface shape which contains a function area(). Write the implementation of the interface for circle, rectangle, and square. Also write the main() to test the interface. Can we declare variables in an interface?

Code:
```
    import java.until.Scanner;
    interface shape {
        void area (Scanner sc);
    }
    class circle implements shape {
        public void area (Scanner sc){
            System.out.println ("Enter Radius of Circle:");
            double r= sc.nextDouble();
            System.out.println ("Area :"+ Math.PI * r * r +
                             "sq.units");
        }
    }
    class regctangle implements Shape {
        public void area (Scanner sc){
            System.out.println ("Enter the length of Rectangle:");
            double l = sc.nextDouble();
            System.out.println ("Enter the breadth of Rectangle:");
            double b = sc.nextDouble();
            System.out.println ("Area :" + Math.PI * l * b +
                             "sq. units");
        }
    }
```

```
class square implements Shape{
    public void area (Scanner sc) {
        System.out.println (" Enter edge length of
                              square :");
        double s = sc.nextDouble();
        System.out.println ("Area : "+s*s + "sq.units");
    }
}
public class Main{
 public static void main (String [] args){
      Scanner sc = new Scanner (System.in);
      Circle c = new Circle();
      c.area (sc);
      Rectangle r = new Rectangle();
      r.aria (sc);
      Square  s = new Square();
      s.area (sc);
      sc.close();
   }
}
```

Yes, variables can be declared inside interface declarations. All variables declared inside interfaces are implicitly public, static and final.

<u>Ques:5</u> Can interface have constructors?

<u>Ans:</u> No, interfaces can not have constructors.
Since constructor is called on instantization of a class, and there is no need to have object of interface, there is no need of ~~inler~~ constructor for interfaces.
furthermore, all methods in interfaces are public and abstract. and all data members are static, public, final. Constructors cannot be abstract and even being an initializer method, it cannot initialize data members which are static and final.

Assignment #3

Q1. What will be the output (write explanation also) of the below program?

```
a)
     public class JavaHungry {
          public static void main(String args[])
          {
               try
               {
                    System.out.print("A");
                    int num = 99/0;
                    System.out.print("B");
               }
               catch(ArithmeticException ex)
               {
                    System.out.print("C");
               }
               catch(Exception ex)
               {
                    System.out.print("D");
               }
               System.out.print("E");
          }
     }
```

Output -> ACE

b)

```
     public class JavaHungry
```

```
{
     public static void main(String args[]) {
          try
          {
               System.out.print("A");
               int num = 99/0;
               System.out.print("B");
          }
          catch(ArithmeticException ex)
          {
               System.out.print("C");
          }
          catch(Exception ex)
          {
               System.out.print("D");
          }
          finally
          {
               System.out.print("E");
          }
          System.out.print("F");
     }
}
```

<u>Output</u> -> ACEF

Q2. Create an exception subclass UnderAge, which prints "Under Age" along with the age value when an object of UnderAge class is printed in the catch statement. Write a class exceptionDemo in which the method test() throws UnderAge exception if the variable age passed to it as argument is less than 18. Write main() method also to show working of the program.. (Try on machine also, if possible)

Code : UnderAge.java ->

```java
public class UnderAge extends Exception {
    private int age;

    public UnderAge(int age) {
        this.age = age;
    }

    @Override
    public String getMessage() {
        return "UnderAge: " + age + " is less than 18";
    }
}
```

exceptionDemo.java ->

```java
import java.util.Scanner;
class exceptionDemo {
    static void test(int age) throws UnderAge {
        if (age < 18)
            throw new UnderAge(age);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Age: ");
        int age = sc.nextInt();
        try {
            test(age);
            System.out.println("Test Successful");
        } catch (UnderAge e) {
            System.err.println(e.getMessage());
```

```
                System.out.println("Test Unsuccessful");
            } finally {
                sc.close();
            }
        }
    }
```

Q3. Write a program to implement stack. Use exception handling to manage underflow and overflow conditions. (Try on machine also, if possible)

Code :

```
public class StackException extends Exception {
    final private String message;

    public StackException(String message) {
        this.message = message;
    }

    @Override
    public String getMessage() {
        return this.message;
    }
}
public class Stack {
     static final int size;
     int top;
     int arr[];

     boolean isEmpty() {
          return (top < 0);
     }
```

```
	public Stack(int size) {
		this.top = -1;
		this.size = size;
		this.arr = new int[this.size];
	}

	public void push(int x) throws StackException {
		if (top >= (this.size - 1)) {
			throw new StackException("Stack Overflow : 
could not push "+x);
		}
		else {
			this.arr[++this.top] = x;
			System.out.println(x + " pushed into 
stack");
		}
	}

	public int pop() throws StackException{
		if (top < 0) {
			throw new StackException("Stack Underflow: 
could not pop");
			return 0;
		}
		else {
			return this.arr[this.top--];
		}
	}

	public int peek() {
		if (top < 0) {
```

```
                throw StackException("Stack Underflow: 
could not peek");
                return 0;
            }
            else {
                return this.arr[this.top];
            }
        }

        @Override
        public String toString(){
            return “Stack size = “+this.size;
        }
    }

    // Driver code
    public class Main {
        public static void main(String args[]) {
            Stack s = new Stack(3);
            s.push(10);
            System.out.println(s);
            s.push(20);
            s.push(30);
            System.out.println(s.pop() + " Popped from 
stack");
            System.out.println(s.peek() + “ is on the top 
of stack”);
            s.push(40);
            s.push(50);
        }
    }
```

Output:

```
10 pushed into stack
```

```
Stack size = 3
20 pushed into stack
30 pushed into stack
30 popped from stack
20 is on the top of stack
40 pushed into the stack
Error: Stack Overflow: could not push 50
```

Q4. Can we write only try block without catch and finally blocks?

Ans: No, it shall result in a compilation error *error: 'try' without 'catch', 'finally' or resource declarations.*

The try block must be followed by either a *catch* or *finally* block. We can remove either catch block or finally block but not both of them. An exception can be made in the case of *try-with-resources* blocks which does not necessarily need to be followed by catch or finally blocks.

Q5. There are three statements in a try block – statement1, statement2 and statement3. After that there is a catch block to catch the exceptions occurred in the try block. Assume that exception has occurred in statement2. Does statement3 get executed or not?

Ans: In case that statement2 throws an exception in the try block, the execution skips to the code in the catch block.

Hence, statement3 will not be executed.

Assignment #4

Q1. What will be the output (write explanation also) of the below program?

```
 import java.io.*;
  class Chararrayinput
  {
      public static void main(String[] args)
      {
           String obj = "abcdef";
           int length = obj.length();
           char c[] = new char[length];
           obj.getChars(0, length, c, 0);
           CharArrayReader input1 = new
CharArrayReader(c);
           CharArrayReader input2 = new CharArrayReader(c,
0, 3);
           int i;
           try
           {
                while((i = input2.read()) != -1)
                {
                     System.out.print((char)i);
                }
           }
           catch (IOException e)
           {
                e.printStackTrace();
           }
      }
  }
```

Output : abc

Q2. Write a program that copies content of one file to another. Pass the names of the files through command-line arguments.

Code:

```
import java.io.*;


public class FileCopy {
    public static void main(String[] args) throws
Exception {
        if (args.length != 2) {
            System.err.println("Correct Usage: java Copy
<src> <dest>");
        } else {
            int i;
            FileInputStream fin = new
FileInputStream(args[0]);
            FileOutputStream fout = new
FileOutputStream(args[1]);
            System.out.println("Copying contents of " +
args[0] + " to " + args[1] + "………");
            while ((i = fin.read()) != -1) {
                fout.write(i);
            }
            fin.close();
            fout.close();
            System.out.println("Copying Done!");
        }
    }
}
```

Q3. Write a program to read a file and display only those lines that have the first two characters as '//' (Use try with resources).

Code:

```
import java.io.*;


public class FileAnalyzer {
    static boolean analyzeLine(String line) {
        if (line.length() >= 2)
            return line.substring(0, 2).equals("//");
        return false;
    }


    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Correct Usage: java 
FileAnalyze <file_name>");
        } else {
            try (BufferedReader br = new 
BufferedReader(new FileReader(args[0]))) {
                String str;
                while ((str = br.readLine()) != null) {
                    //Removing leading and trailing space
                    str = str.trim();
                    //Checking condition
                    if (analyzeLine(str)) {
                        System.out.println(str);
                    }
                }
                br.close();
            } catch (Exception e) {
                System.err.println(e.getMessage());
```

```
                        }
                    }
                }
            }
```

Q4. How do you handle console output using PrintWriter class?

Ans: To write to the console by using a PrintWriter, we specify System.out for the output stream and flush the stream after each newline. For example, this line of code creates a PrintWriter that is connected to console output:

```
PrintWriter pw = new PrintWriter(System.out, true);
```

Here, the first argument is the output steam object and second argument (flushOnNewline) controls whether Java flushes the output stream every time a println( ) method is called.

The following application illustrates using a PrintWriter to handle console output:

```
// Demonstrate PrintWriter
import java.io.*;
public class PrintWriterDemo {
     public static void main(String args[]) {
          PrintWriter pw = new PrintWriter(System.out,
     true);
          pw.println("This is a string");
          int i = -7;
          pw.println(i);
          double d = 4.5e-7;
          pw.println(d);
     }
}
```

The output from this program is shown here:

```
This is a string
```

```
-7
4.5E-7
```

Q5. How do you read 1) characters 2) a string , using BufferedReader class?

Ans: (i) To read a character from a BufferedReader, use read( ). The version of read() that we will be using is

```
int read( ) throws IOException
```

Each time that read( ) is called, it reads a character from the input stream and returns it as an integer value. It returns –1 when the end of the stream is encountered. As we can see, it can throw an IOException.

The following program demonstrates read( ) by reading characters from the console until the user types a "q.”

```
import java.io.*;
class BRRead {
    public static void main(String args[])
    throws IOException{
        char c;
        BufferedReader br = new
        BufferedReader(new
    InputStreamReader(System.in));
        System.out.println("Enter characters, 'q' to
        quit.");

        // read characters
        do {
            c = (char) br.read();
            System.out.println(c);
        } while(c != 'q');
    }
}
```

(ii) To read a string from the keyboard, use the version of readLine( ) that is a member of the BufferedReader class. Its general form is shown here:

```
String readLine( ) throws IOException
```

As we can see, it returns a String object.

The following program demonstrates BufferedReader and the readLine( ) method; the program reads and displays lines of text until you enter the word “stop”:

```
// Read a string from console using a BufferedReader.
import java.io.*;
class BRReadLines {
     public static void main(String args[]) throws
IOException {
          // create a BufferedReader using System.in
          BufferedReader br = new BufferedReader(new
     InputStreamReader(System.in));
          String str;
          System.out.println("Enter lines of text.");
          System.out.println("Enter 'stop' to quit.");
          do {
               str = br.readLine();
               System.out.println(str);
          } while(!str.equals("stop"));
     }
}
```

Q6. Write name and meaning of all stream classes discussed in reference book.

Ans: Input Stream Classes ->

Class	Description
BufferedInputStream	contains methods to read bytes from the buffer (memory area)
ByteArrayInputStream	contains methods to read bytes from a byte array
DataInputStream	contains methods to read Java primitive data types

FileInputStream	contains methods to read bytes from a file
FilterInputStream	contains methods to read bytes from other input streams which it uses as its basic source of data
ObjectInputStream	contains methods to read objects
PipedInputStream	contains methods to read from a piped output stream. A piped input stream must be connected to a piped output stream
SequenceInputStream	contains methods to concatenate multiple input streams and then read from the combined stream

Output Stream Classes ->

Class	Description
BufferedOutputStream	Contains methods to write bytes into the buffer
ByteArrayOutputStream	Contains methods to write bytes into a byte array
DataOutputStream	Contains methods to write Java primitive data types
FileOutputStream	Contains methods to write bytes to a file
FilterOutputStream	Contains methods to write to other output streams
ObjectOutputStream	Contains methods to write objects
PipedOutputStream	Contains methods to write to a piped output stream
PrintStream	Contains methods to print Java primitive data types

Assignment #5

Q1. Write a program that copies content of one file to another. Pass the names of the files through command-line arguments.

Code:

```
import java.io.*;


public class FileCopy {

    public static void main(String[] args) throws
Exception {

        if (args.length != 2) {

            System.err.println("Correct Usage: java Copy
<src> <dest>");

        } else {

            int i;

            FileInputStream fin = new
FileInputStream(args[0]);

            FileOutputStream fout = new
FileOutputStream(args[1]);

            System.out.println("Copying contents of " +
args[0] + " to " + args[1] + "………");

            while ((i = fin.read()) != -1) {

                fout.write(i);

            }

            fin.close();

            fout.close();

            System.out.println("Copying Done!");

        }

    }

}
```

Q2. Write a program in Java (using try-with-resources functionality) to do the following:

i) open two files "exam1.txt" and "exam2.txt". Accept file names through command-line arguments.

ii) exit the program if any of the two files is unable to open.

iii) append contents of "exam1.txt" to "exam2.txt".

iv) re-write contents of "exam2.txt" after removing all white spaces from the updated content (without using built-in methods).

Code:

```
import java.io.*;


public class Main {

    public static void main(String[] args) {

        if (args.length != 2) {

            System.out.println("Correct Usage: java Main
<file1> <file2>");

            System.exit(1);

        }


        try (BufferedReader finA = new BufferedReader(new
FileReader(args[0]));

                BufferedWriter foutB = new
BufferedWriter(new FileWriter(args[1], true))) {

            String s;

            while ((s = finA.readLine()) != null) {

                foutB.newLine();

                foutB.write(s);

                foutB.flush();

            }

            System.out.println("Task 1 Done...");

        } catch (Exception e) {
            System.out.println("Could not open one of the
files, exiting...");
            System.exit(-1);
        }

        try (BufferedReader finB = new BufferedReader(new
FileReader(args[1]))) {
            String s, o = "";
            while ((s = finB.readLine()) != null) {
                char[] array = s.toCharArray();
                for (int i = 0; i < s.length(); i++)
                    switch (array[i]) {
                        case ' ':
                        case '\t':
                        case '\n':
                        case '\r':
                            break;
                        default:
                            o += array[i];
                            break;
                    }
            }
            try (BufferedWriter foutB = new
BufferedWriter(new FileWriter(args[1]))) {
                foutB.write(o);
            }
            System.out.println("Task 2 Done!");
        } catch (Exception e) {
            System.out.println("Could not open one of the
files, exiting...");
            System.exit(-1);
        }
        }
    }
```

Q3. What is Delegation Event Model? Explain its components in short.

Ans: **Delegation Event Model** : The modern approach to handling events is based on the delegation event model, which defines standard and consistent mechanisms to generate and process events. Its concept is quite simple: a source generates an event and sends it to one or more listeners. In this scheme, the listener simply waits until it receives an event. Once an event is received, the listener processes the event and then returns. The advantage of this design is that the application logic that processes events is cleanly separated from the user interface logic that generates those events. A user interface element is able to “delegate” the processing of an event to a separate piece of code.

The components of the Delegation Event Model are as follows :

Events : In the delegation model, an event is an object that describes a state change in a source. It can be generated as a consequence of a person interacting with the elements in a graphical user interface. Events may also occur that are not directly caused by interactions with a user interface.

Event Sources : A source is an object that generates an event. This occurs when the internal state of that object changes in some way. Sources may generate more than one type of event. A source must register listeners in order for the listeners to receive notifications about a specific type of event. Each type of event has its own registration method.

Event Listeners : A listener is an object that is notified when an event occurs. It has two major requirements. First, it must have been registered with one or more sources to receive notifications about specific types of events. Second, it must implement methods to receive and process these notifications.

Q4. Write commonly used Event Classes in java.awt.event and their description.

Ans.

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.
InputEvent	Abstract superclass for all component input event classes.
ItemEvent	Generated when a check box or list item is clicked; also occurs when a choice selection is made or a checkable menu item is selected or deselected.
KeyEvent	Generated when input is received from the keyboard.
MouseEvent	Generated when the mouse is dragged, moved, clicked, pressed, or released; also generated when the mouse enters or exits a component.
MouseWheelEvent	Generated when the mouse wheel is moved.
TextEvent	Generated when the value of a text area or text field is changed.
WindowEvent	Generated when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

Assignment #6

Q1. Write different constants and their description available in AdjustmentEvent class.

Ans: The different constants and their meanings of AdjustmentEvent class are shown below :

Constant	Description
BLOCK_DECREMENT	The user clicked inside the scroll bar to decrease its value.
BLOCK_INCREMENT	The user clicked inside the scroll bar to increase its value.
TRACK	The slider was dragged.
UNIT_DECREMENT	The button at the end of the scroll bar was clicked to decrease its value.
UNIT_INCREMENT	The button at the end of the scroll bar was clicked to increase its value.
ADJUSTMENT_VALUE_CHANGED	An integer constant, that indicates that a change has occurred.

Q2. Write different constants and their description available in ComponentEvent class.

Ans: The different constants and their meanings of ComponentEvent class are shown below :

Constant	Description
COMPONENT_HIDDEN	The component was hidden.
COMPONENT_MOVED	The component was moved.
COMPONENT_RESIZED	The component was resized.
COMPONENT_SHOWN	The component became visible.

Q3. Explain syntax of all constructors available in ContainerEvent and FocusEvent class.

Ans:

Constructors in ContainerEvent Class

```
ContainerEvent(Component src, int type, Component
comp)
```

It instantiates a ContainerEvent object.

The first argument src is a reference to the Component object (container) that originated the event. This method throws an IllegalArgumentException if src is null. The second argument type is an integer indicating the type of event. The third argument comp is the reference to the Component that was added or removed.

Constructors in FocusEvent Class

```
FocusEvent(Component src, int type)
```

It instantiates a FocusEvent object and identifies it as a permanent change in focus.

Here, the first argument src is the Component that originated the event. This method throws an IllegalArgumentException if src is null. The second argument type is an integer indicating the type of event.

```
FocusEvent(Component src, int type, boolean
temporaryFlag)
```

It instantiates a FocusEvent object and identifies whether or not the change is temporary.

Here, the first argument src is the Component that originated the event. This method throws an IllegalArgumentException if src is null. The second argument type is an integer indicating the type of event. The third argument temporaryFlag is a boolean value which is set to true if the focus change is temporary and is false otherwise.

```
FocusEvent(Component src, int type, boolean
temporaryFlag, Component other)
```

It instantiates a FocusEvent object with the specified temporary state and opposite Component. The opposite Component is the other Component involved in this focus change. For a FOCUS_GAINED event, this is the Component that lost focus. For a FOCUS_LOST event, this is the Component that gained focus. If this focus change occurs with a native application, with a Java application in a different VM, or with no other Component, then the opposite Component is null.

Here, the first argument src is the Component that originated the event. This method throws an IllegalArgumentException if src is null. The second argument type is an integer indicating the type of event. The third argument temporaryFlag is a boolean value which is set to true if the focus change is temporary and is false otherwise. The fourth argument other is the opposite Component involved in the focus change, or null.

Q4. Write the name of constants present in InputEvent class.

Ans:

1) ALT_MASK
2) ALT_GRAPH_MASK
3) BUTTON1_MASK
4) BUTTON2_MASK
5) BUTTON3_MASK
6) CTRL_MASK
7) META_MASK
8) SHIFT_MASK
9) ALT_DOWN_MASK
10) ALT_GRAPH_DOWN_MASK
11) BUTTON1_DOWN_MASK
12) BUTTON2_DOWN_MASK
13) BUTTON3_DOWN_MASK
14) CTRL_DOWN_MASK
15) META_DOWN_MASK
16) SHIFT_DOWN_MASK

Q5. List all the constants present in KeyEvent class.

Ans:

1) VK_ALT
2) VK_DOWN
3) VK_LEFT
4) VK_RIGHT
5) VK_CANCEL
6) VK_ENTER
7) VK_PAGE_DOWN
8) VK_SHIFT

9) VK_CONTROL
10) VK_ESCAPE
11) VK_PAGE_UP
12) VK_UP

Q6. List all the constants present in MouseEvent class.

Ans:

1) MOUSE_CLICKED
2) MOUSE_DRAGGED
3) MOUSE_ENTERED
4) MOUSE_EXITED
5) MOUSE_MOVED
6) MOUSE_PRESSED
7) MOUSE_RELEASED
8) MOUSE_WHEEL

Q7. List all the constants and their meaning present in WindowEvent class.

Ans:

1) WINDOW_ACTIVATED
2) WINDOW_CLOSED
3) WINDOW_CLOSING
4) WINDOW_DEACTIVATED
5) WINDOW_DEICONIFIED
6) WINDOW_GAINED_FOCUS
7) WINDOW_ICONIFIED
8) WINDOW_LOST_FOCUS
9) WINDOW_OPENED
10) WINDOW_STATE_CHANGED