Reg.No | | | | | | | |

SNS COLLEGE OF TECHNOLOGY

(Autonomous)

A

MCA- Internal Assessment –II (July 2023)

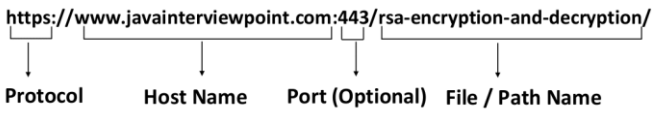
Academic Year 2022-2023(Even) / SecondSemester

19CAT608 – Java Programming – Answer Key

Time: $1^{1/2}$ Hours **Maximum Marks: 50**

Answer All Questions

PART - A (5 x 2 = 10 Marks)

		CO	BL
1	**List any four Layouts used in Java.** • BorderLayout. • BoxLayout. • CardLayout. • FlowLayout. • GridBagLayout. • GridLayout. • GroupLayout. • SpringLayout.	CO2	Und
2	**How many times may an object's finalize() method be invoked by the garbage collector?** Depends on number of times calling Sytem.gc(), finalize() will be invoked.	CO2	Rem
3	**Differentiate Multithreading and Multiprocessing** **Multithreading:** 1. Creates multiple threads of a single process to increase computing power 2. Concurrently execute the single process 3. Creation of thread is economical in both sense time and resource 4. Multi threading is not classified **Multiprocessing:** 1. Adds CPU to increase computing power 2. Multiple processes are executed concurrently 3. Creation of a process is time consuming and resource intensive 4. Mutiprocessing can be symmetric or asymmetric.	CO3	Ana
4	**Write the methods used to parse the URL.** 	CO3	Rem
5	**Check the following syntax and write the correct one:** Fileinputstream in=FileinputStream("Dsam.txt) Corrected one is FileInputStream in=FileInputStream("Dsam.txt")	CO3	Ana

PART - B (2x13=26, 1x14=14Marks)

6	(a)	**Classify Garbage Collection in Java with suitable example.** Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects. **Advantage:** 1. Memory Efficient	CO2	Eva

2. Automatically done

Difference ways:

- By nulling the reference

Employee e=new Employee();
e=null;

- By assigning a reference to another

Employee e1=new Employee();
Employee e2=new Employee();
e1=e2;

- By anonymous object etc.

new Employee();

Two methods:

1. finalize()
2. gc()

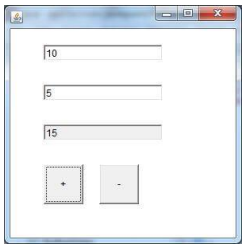
Example

```
1. public class TestGarbage1{
2.  public void finalize(){System.out.println("object is garbage collected");}
3.  public static void main(String args[]){
4.   TestGarbage1 s1=new TestGarbage1();
5.   TestGarbage1 s2=new TestGarbage1();
6.   s1=null;   s2=null;   System.gc();
7.  } }
```

(Or)

(b) Write a Java program to perform basic arithmetic operations on two numbers using CO2 App TextField and Button to handle ActionEvent in an Applet.

```
import java.awt.*;
public class TextFieldExample2 extends Frame implements Ac
TextField tf1, tf2, tf3;
Button b1, b2;
TextFieldExample2() {
tf1 = new TextField();   tf1.setBounds(50, 50, 150, 20);
tf2 = new TextField();   tf2.setBounds(50, 100, 150, 20);
tf3 = new TextField();   tf3.setBounds(50, 150, 150, 20);
tf3.setEditable(false);  b1 = new Button("+");
b1.setBounds(50, 200, 50, 50); b2 = new Button("-");
b2.setBounds(120,200,50,50);  b1.addActionListener(this);
b2.addActionListener(this);   add(tf1);  add(tf2);  add(tf3);  add(b1);  add(b2);
setSize(300,300);   setLayout(null);   setVisible(true);   }
public void actionPerformed(ActionEvent e) {
String s1 = tf1.getText();
String s2 = tf2.getText();
int a = Integer.parseInt(s1);
int b = Integer.parseInt(s2);
int c = 0;
if (e.getSource() == b1){   c = a + b;   }
else if (e.getSource() == b2){   c = a - b;   }
String result = String.valueOf(c);
tf3.setText(result);   }
public static void main(String[] args) {
new TextFieldExample2();
}   }
```



7 (a) **Demonstrate Multithread. Explain the life cycle of thread with real cases.** CO3 Ana

Multithreading is **a programming concept in which the application can create a small unit of tasks to execute in parallel**. If you are working on a computer, it runs multiple applications and allocates processing power to them.

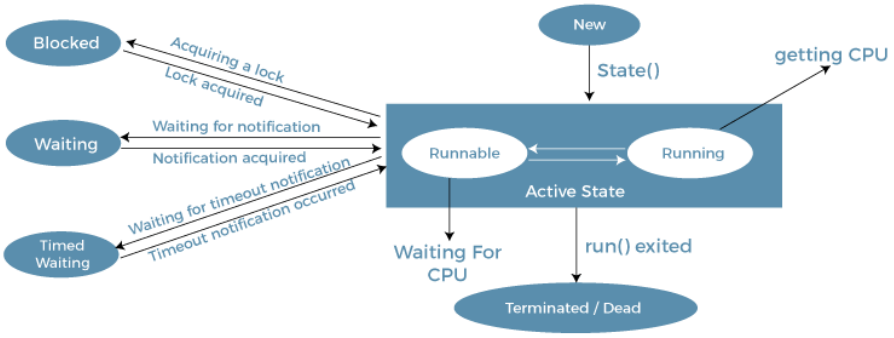
Advantages:

1. Doesn't block the user
2. Perform many operations together: time save
3. Independent

Class: Thread

Interface: Runnable

Life Cycle:



Life Cycle of a Thread

Real case example: factorial (Recursive problems)

(Or)

(b) Categorize IO packages and evaluate how to read and write txt files in java with an example. CO3 Eva

Java I/O (Input and Output) is used to process the input and produce the output. Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations. We can perform file handling in Java by Java I/O API.

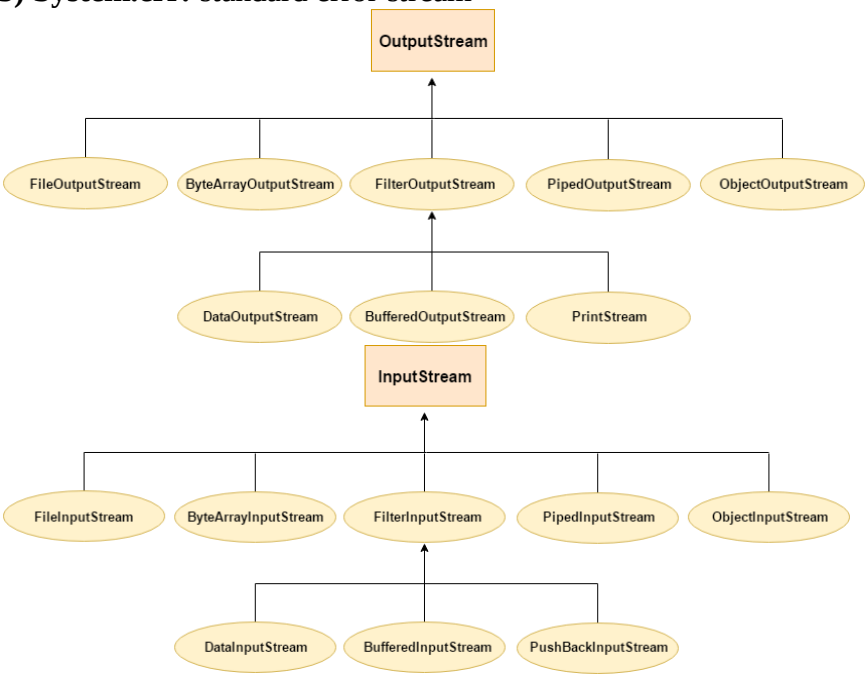
Stream:

It is a sequence of data

1) System.out: standard output stream

2) System.in: standard input stream

3) System.err: standard error stream



```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
public static void main(String args[]){
try{
FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
fout.write(65);    fout.close();    System.out.println("success...");
}catch(Exception e){System.out.println(e);}
}    }
```

```
import java.io.FileInputStream;
public class DataStreamExample {
public static void main(String args[]){
try{
FileInputStream fin=new FileInputStream("D:\\testout.txt");
int i=fin.read();  System.out.print((char)i);    fin.close();
}catch(Exception e){System.out.println(e);}
}    }
```

8 (a) Write a java program to design user interface and implement event handler.

```
1. import java.awt.*;
2. class AEvent extends Frame implements ActionListener{
3. TextField tf;
4. AEvent(){
5. tf=new TextField();
6. tf.setBounds(60,50,170,20);
7. Button b=new Button("click me");
8. b.setBounds(100,120,80,30);
9. b.addActionListener(this);//passing current instance
10. add(b);add(tf);  setSize(300,300);
11. setLayout(null);  setVisible(true);  }
12. public void actionPerformed(ActionEvent e){
13. tf.setText("Welcome");  }
14. public static void main(String args[]){
15. new AEvent();  }  }
```

CO2 App

(Or)

(b) Case Study: Keep client to read input from the user and send it to server until "Over" is typed. CO3 Cre

Client Program

```
import java.io.*;import java.net.*;
public class Client {
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
     public Client(String address, int port)
    {
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");
             input = new DataInputStream(System.in);
             out = new DataOutputStream(socket.getOutputStream());
        }
```

```
 catch (UnknownHostException u) {System.out.println(u);return;
        }
        catch (IOException i) {System.out.println(i);return;
        }
         String line = "";
         while (!line.equals("Over")) {
             try {line = input.readLine();
                 out.writeUTF(line);
             }
             catch (IOException i) {
                 System.out.println(i);
             }       }
         // close the connection
        try {input.close();
             out.close();
             socket.close();
        }
        catch (IOException i) {
             System.out.println(i);
        }     }
     public static void main(String args[])
    {         Client client = new Client("127.0.0.1", 5000);     }}
```

Server Program

```
import java.net.*;import java.io.*;
 public class Server
{
    private Socket          socket   = null;
    private ServerSocket    server   = null;
    private DataInputStream in       =  null;
    public Server(int port)
    {  try
        {   server = new ServerSocket(port);
            System.out.println("Server started");
             System.out.println("Waiting for a client ...");
             socket = server.accept();
            System.out.println("Client accepted");
             in = new DataInputStream(
            new BufferedInputStream(socket.getInputStream()));
             String line = "";
             while (!line.equals("Over"))
            {try
                {   line = in.readUTF();
                    System.out.println(line);
                 }catch(IOException i)
                {System.out.println(i); }
            } System.out.println("Closing connection");
             socket.close();
            in.close();
        }catch(IOException i)
        {System.out.println(i);
        }   }
     public static void main(String args[])
    {     Server server = new Server(5000);    }
}
```

Client Input

```
Hello    I made my first socket connection  Over
```

Server Output

```
Hello I made my first socket connection   Over
Closing connection
```