J. VARUN IYER CSE 3rd Sem 19115037

JAVA PRACTICALS:

Q1. Write a java program based on decision making, branching and looping.

→ Theory: In this program, we will use if-else statement for decision making, for statement for looping & break & continue statements for branching.

Code:

```
public class practical {
public static void main(String [] args){
//decision
int a=5, b=6;
if (a==b) {system.out.println("no.s are equal"); }
else {System.out.println("numbers are not equal");}
//looping
int i=0;
for(i=0; i<10; i++)
{
  System.out.println("i= " + i);
} //branching
int j;
for (j=0; j<10; j++)
{  if(j%2==0) continue;
   if(j==9) break;
   System.out.println("j= "+j);
} } }
```

Output: no.s numbers are not equal

i=0
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
j=1
j=3
j=5
j=7

Conclusion: The above program efficiently shows statements implementing decision making, branching and looping.

Q2. Write java program to perform string operations.

→ Theory: In this program, we will see various operations on strings such as length(), contains(), equals(), isEmpty(), concat(), replace(), toUpperCase(), trim().

Code:
```
public class practical{
public static void main(String[] args){
  String a= "abcdefg";
  String b= "hijklmn";
  System.out.println(a.length());
  String c= a.substring(2,6);
  System.out.println(a.contains("d"));
```

```
System.out.println(a.isEmpty());
System.out.println(c.concat(a));
System.out.println(c);
System.out.println(a.replace('a', 'd'));
System.out.println(a.toUpperCase());
String u="   asg  ";
System.out.println(u.trim());
int w=5;
String f=String.valueOf(w);
System.out.println(f);
}
}
```

Output: 7
true
false
false
cdefabcdefg
cdef
ABCDEFG
asg
5

Q3. Write java program to demonstrate interfaces.

→ Theory: An interface is reference type in java, consisting of a collection of abstract methods So, when a class implements an interface, all these methods are inherited. A class can implement multiple interfaces.

Code: (In animal.java)

```
interface animal{
public void eat();
public void travel();
}
```

(in bird.java)

```
Public class bird implements animal{
    Public void eat(){
     System.out.println("worms");
    }
    Public void travel(){
     System.out.println("wings");
    }
Public static void main(String[] args){
      bird b = new bird();
       b.eat();
       b.travel();
   }
}
```

Output: worms
wings
(program ends)

Q4. Write java program to demonstrate packages.

→ Theory: Packages are groups of similar types of classes, interfaces and subpackages. Packages can be built-in or user defined. Ex. java, lang, awt, etc.

Code: (in hello.java)

```
package Hello
public class hello {
    public int a= 5;
    public void show()
    { System.out.println("Hey guys!"); }
}
```

(In practical.java)

```
Import Hello.*;
Public class practical {
Public static void main (String [] args) {
    hello ob:- new hello();
    Ob.show();
    System.out.println(ob.a);
```

Output: Hey guys!
5
(program ends)

---

05. Write Java program to demonstrate exceptions using try and multiple catch blocks.

→ Theory: An exception is a problem that arises during the execution of the program & the program terminates abnormally. So, these exceptions need to be handled. There are checked & unchecked exceptions.

Code:

```
public class practical {
    public static void main(String[] args) {
    try {
        int a[ ] = new int [5];
        a[5] = 30/0;
        }
    catch(ArithmeticException e)
        {
        System.out.println("can't divide by 0");
        }
    catch(ArrayIndexOutOfBoundsException e)
    { System.out.println("Array index out of bounds);
```

```
}
catch (exception e)
{
  System.out.println("parent exception");
}
}
}
```

Output: Array index out of bounds

Q6. Write Java program on method overloading, constructor & constructor overloading.

→ Theory: In overloading, the same method is given different definitions depending on the no. of parameters fed into the method. Constructors are ~~builder~~ methods which can be defined by users to initialize class objects.

Code:

```
class box {
    int width, height, depth;
    Box(int w, int h, int d){
        width = w;
        height = h;
        depth = d;
    }
    Box(){    //same constructor overloaded
      width = height = depth = 0;
    }
    Box(int l){
      width = height = depth = l;
    }
     int volume()
    {
      return width * height * depth;
    }
```

```
    public static void main (String [] args)
{
    box b1 = new box();
    box b2 = new box(1,2,3);
    box b3 = new box(7);
    System.out.println(b1.volume());
    System.out.println(b2.volume());
    System.out.println(b3.volume());
  }
}
```

Output: 0
6
343
(program ends)

---

07. Write java program to demonstrate inheritance.

→ Theory: Inheritance can be defined as the process where one class acquires the properties (methods & fields) of another class. Information is managed in hierarchial order using inheritance. We use the 'extends' keyword.

Code:
```
class calc{
    int z;
    public void add(int x, int y)
    {  z = x+y;
       System.out.println("Sum is"+z);
    }
    public void sub(int x, int y)
    { z = x-y;
      System.out.println("Diff is"+z);
    }
```

```
}
public class mycal extends calc {
    public void mul (int x, int y)
    { z = x*y;
      System.out.println ("product is"+z);
    }
public static void main (String [] args) {
    int a=5, b=6;
    mycal c = new mycal();
     c.add (a, b);
     c.sub (a,b);
     c.mul (a,b);
   }
}
```

Output : 11
-1
30

---

08. Write java program to demonstrate multi-threading.

→ Theory: Multithreading is a java feature that allows concurrent execution of two or more parts of a program for maximum utilization of cpu. Each part of such program is called a thread.

Code: (in Newthr.java)

```
public class Newthr extends Thread
{
  public Newthr (String name) {
      super (name);
    }
```

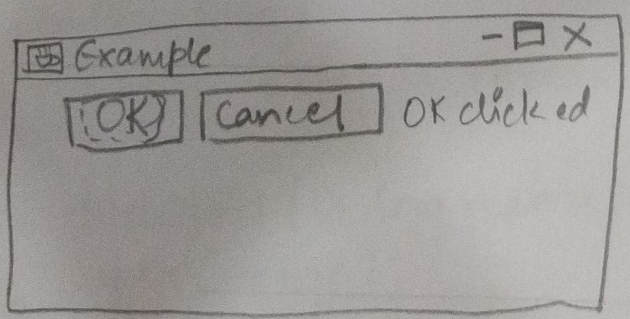
@Override
```
Public void run( ){
    try{
        for(int i=5;i>0;i--){
        System.out.println(this.getName()+":"+i);
        Thread.sleep(2000);
        }
    }
    catch(InterupptedException e){
        System.out.println(this.getName()+"interupted");
    }
}
```
(In demo.java)
```
public class demo{
public static void main(String[] args){
    Newthr t1= new Newthr("one");
    Newthr t2= new Newthr("two");
     t1.start();
     t2.start();
    }
}
```
Output:
```
one: 5
two: 5
one: 4
two: 4
two: 3
one: 3
one: 2
two: 2
one: 1
two: 1
```

Q9. Write java program to demonstrate swing components.

→ Theory: Swing is used to make interactive GUI on java. It is built on foundation of AWT. Swing has light-weight components & are powerful & flexible. All classes with with letter J. ex. JCheckBox

Code:
```
import javax.swing.*; import java.awt.*;
    Public class practical{
Public static void main(String[] args){
 JFrame frame=new JFrame("Example");
    frame.setSize(200,300);
    frame.setVisible(true);
    frame.setLayout(new FlowLayout());
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel label = new JLabel("Which button clicked");
    JButton ok= new JButton("ok");
    JButton can = new JButton("can");
    ok.addActionListener((ae)-> label.setText("OK clicked"));
can.addActionListener((ae)-> label.setText("Cancel clicked"));
    frame.add(ok);
    frame.add(can);
    frame.add(label);
    frame.pack();
  }
 }
```

Output:



Q10. Write Java program to demonstrate AWT packages.

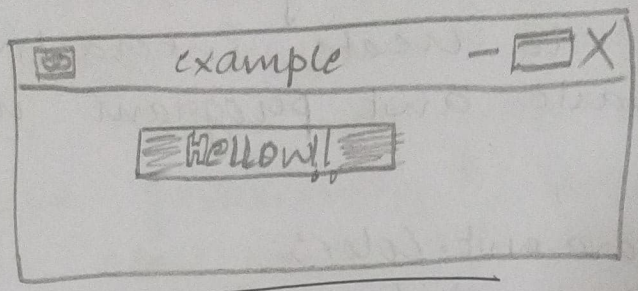
→ Theory: The AWT (Abstract Window Toolkit) classes are contained in the java.awt.* package. AWT is used to create interactive GUI. Here we see a basic awt perogram using various packages.

Code:

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class ex extends Frame {
    public ex() {
    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent e){
            System.exit(0);
        }
    });
    }
public static void main(String[] args) {
    ex e1 = new ex();
    e1.setSize(new Dimension(400,200));
    e1.setTitle("example");
    e1.setVisible(true);
}
@Override
public void paint(Graphics g){
    g.drawRect(80,50,100,30);
    g.setColor(Color.BLACK);
    g.fillRect(80,50,100,30);
```

```
        g.setColor(Color.WHITE);
        g.drawString("Hellow!!", 100, 70);
    }
}
```

Output:

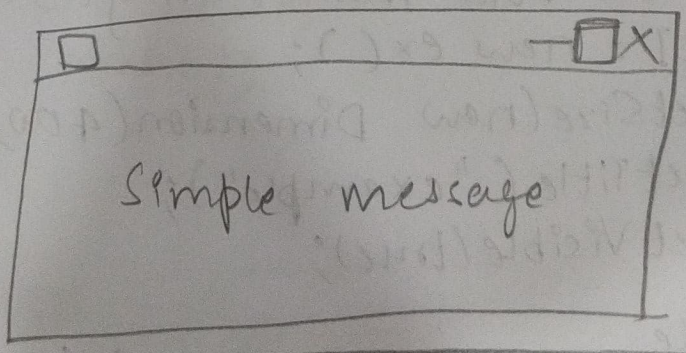


Q11. Develop an applet that displays a simple message.

→ Theory: An applet is a Java program that runs in a web browser. It has interactive GUI. Applet was discontinued after Java 8.

Code:
```
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet {
    public void paint(Graphics g) {
        g.drawString("simple message", 150, 150);
    }
}
```

Output:



Q12. Write Java program to demonstrate Appletevent handling.

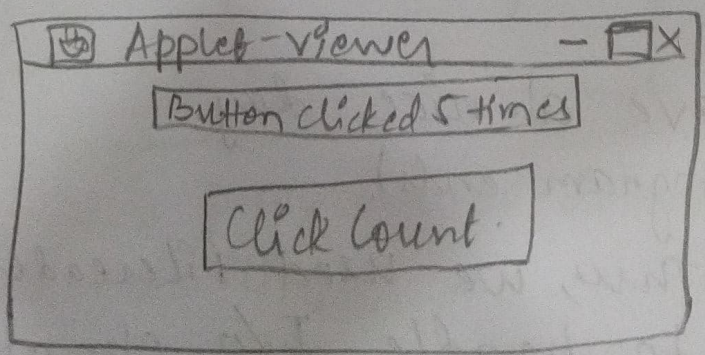
→ Theory: GUI components are responsible to

generate events based on user interactions like clicking a key or a mouse-click. So, applet lets us define what actions will occur when we click certain buttons/key.

Code:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class ev extends Applet implements ActionListener{
    Button b;
    TextField tf;
    int c;
    public void init(){
    tf = new TextField();
    tf.setBounds(30, 40, 150, 20);
    b = new Button("Click count");
    b.setBounds(80, 150, 60, 50);
    add(b);
    add(tf);
    b.addActionListener(this);
    setLayout(null);
    }
    public void actionPerformed(ActionEvent a){
        c++;
    tf.setText("Button clicked "+c+" times");
    }
}
```

Output:



013. Write java program based on I/O handling.

→ Theory: In I/O (Input/output) we use java programs to process input & produce output using Streams. The package is java.io & can be used for performing file handling.

Code:
```
public class ex {
public static void main(String[] args) {
try{
    FileWriter fw=new FileWriter("New.txt");
   fw.write("have a good day");
    fw.close();
   FileReader fr=new FileReader("New.txt");
   int i;
   while((i=fr.read())!=-1)
     { System.out.println((char)i);
      fr.close();
   }
}
catch(Exception e){
   System.out.println(e);
  }
}}
```

Output: have a nice day.
(program ends).

Conclusion: Thus, we used Filereader & FileWriter classes to handle I/O operations.