Name : Pratyay Dhond

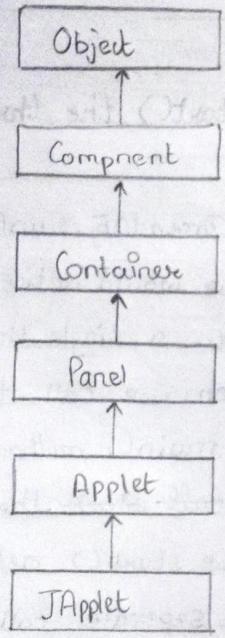
Department : IT

Roll No : 1907011

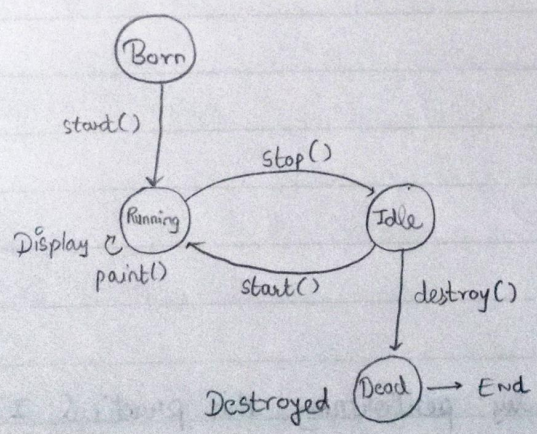
Practical No. 15

Aim: Create, debug and run Java programs based on Applets.

Diagram:



15.1 Hierarchy of Applet.



15.2 Applet Life Cycle

PAGE NO.
DATE

Practical No. 15

Aim: Create, debug and run Java programs based on Applets.

Theory:

What is a java applet?

Applet is a specific special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

What are the advantages of Applet?

- It works at client side, so less response time.
- Can be used by browsers running under various platforms, including Linux, windows, Mac OS X.

What is the drawback of applets?

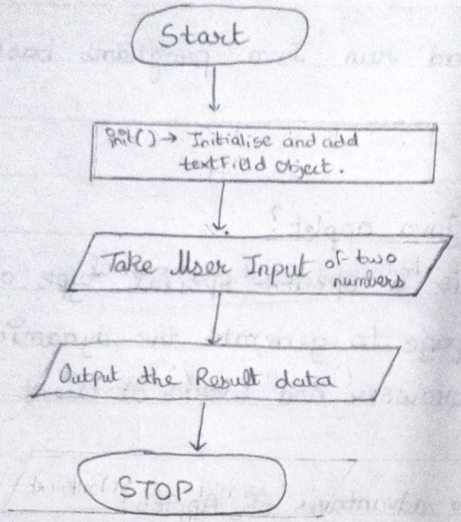
A major drawback of applets is that a plugin is required at client browser to ~~exeut~~ execute applet.

What are the stages of Applet life cycle?

The four states in an applet life cycle are:

1) Born or initialization state
2) Running State
3) Idle state
4) Dead or destroyed state

Flowchart:



Conclusion:

Hence, by performing this practical, I created, debugged and executed java programs based on Applets.

Code:

Input.java

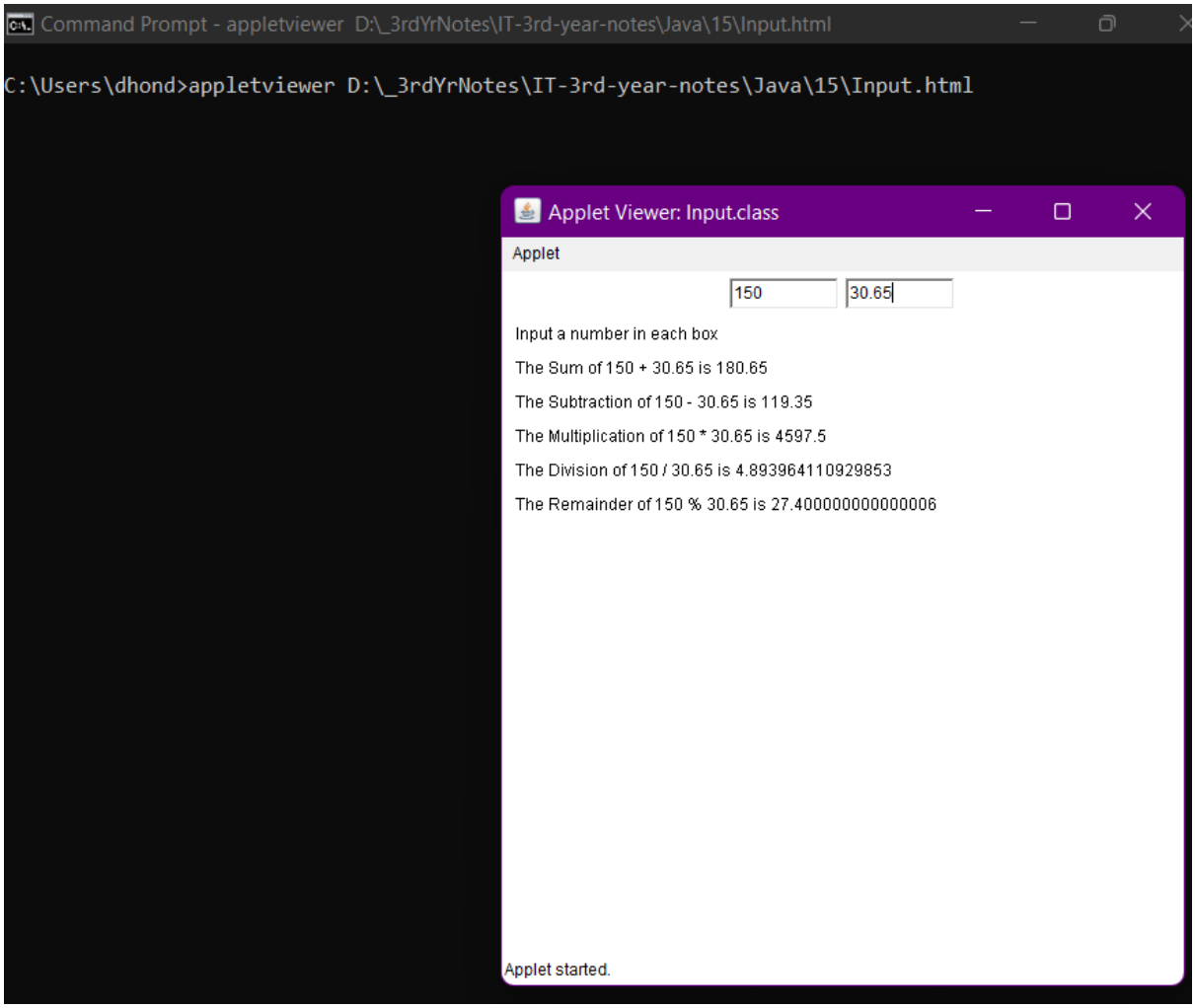
```
import java.awt.*;
import java.applet.*;

/*<applet code="Input.class" height="500" width="500">*/

public class Input extends Applet{
  TextField text1, text2;

  public void init(){
    text1 = new TextField(8);
    text2 = new TextField(8);
    add(text1);
    add(text2);
    text1.setText("0");
    text2.setText("0");
  }

  public void paint(Graphics g){
    double x = 0;
    double y = 0;
    double sum = 0, mul=0, div = 0, remainder = 0, sub = 0;
    String s1="",s2="",s3="";

    g.drawString("Input a number in each box",10,50);
    try{
      s1 = text1.getText();
      x = Double.parseDouble(s1);
      s2 = text2.getText();
      y = Double.parseDouble(s2);
    }catch(Exception e){
      System.out.println("Exception Occured : " + e);
    }finally{
    sum = x + y;
    sub = x - y;
    mul = x * y;
    div = x / y;
    remainder = x % y;

    s3 = String.valueOf(sum);
    String temp = "The Sum of " + s1 + " + " + s2 + " is " + s3;
    g.drawString(temp,10,75);
    temp = "The Subtraction of " + s1 + " - " + s2 + " is " +
String.valueOf(sub);
```

```
    g.drawString(temp,10,100);
    temp = "The Multiplication of " + s1 + " * " + s2 + " is " +
String.valueOf(mul);
    g.drawString(temp,10,125);
    temp = "The Division of " + s1 + " / " + s2 + " is " +
String.valueOf(div);
    g.drawString(temp,10,150);
    temp = "The Remainder of " + s1 + " % " + s2 + " is " +
String.valueOf(remainder);
    g.drawString(temp,10,175);
    }

  }

  public boolean action(Event event, Object object){
    repaint();
    return true;
  }

}
```

Input.html

```
<html>
<applet code="Input.class" height="500" width="500"> </applet>
</html>
```

Output:



Applet To Add Two Numbers

PAGE NO.
DATE

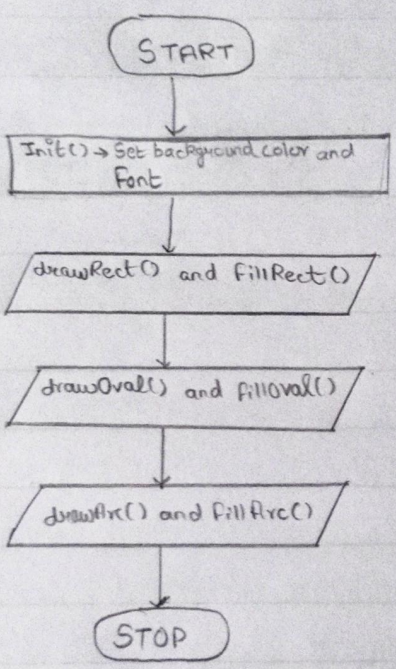
Conclusion:

Hence, by performing this practical, I created, debugged and executed Java programs based on Applets.

Practical No 16

Aim : Create, debug and run java programs based on graphics to draw, fill, different shape.

Flowchart :



PAGE NO.
DATE

Practical No. 16

Aim: Create, debug and run Java programs based on graphics to draw, fill, different shapes.

Theory:

Java.awt.Graphics class provides many methods for graphics programming.

The methods commonly used from the Graphics class are:

1. public abstract void drawString(String str, int x, int y);
2. public void drawRect(int x, int y, int height, int width);
3. public abstract void fillRect(int x, int y, int width, int height);
4. public abstract void drawOval(int x, int y, int width, int height);
5. public abstract void fillOval(int x, int y, int width, int ~~arc~~ height);
6. public abstract void drawLine(int x_1, int y_1, int x_2, int y_2);
7. ~~pt~~ public abstract ~~void~~ boolean drawImage(Image ping, int x, int y, Image Observer observer);
8. public abstract void drawArc(int x, int y, int width, int height, int ~~startAngle~~, int arcAngle);
9. public abstract void fillArc(int x, int y, int width, int height, int ~~startAngle~~, int arcAngle);
10. public abstract ~~void~~ void setColor(Color c);
11. public abstract void setFont(Font font);

Conclusion :

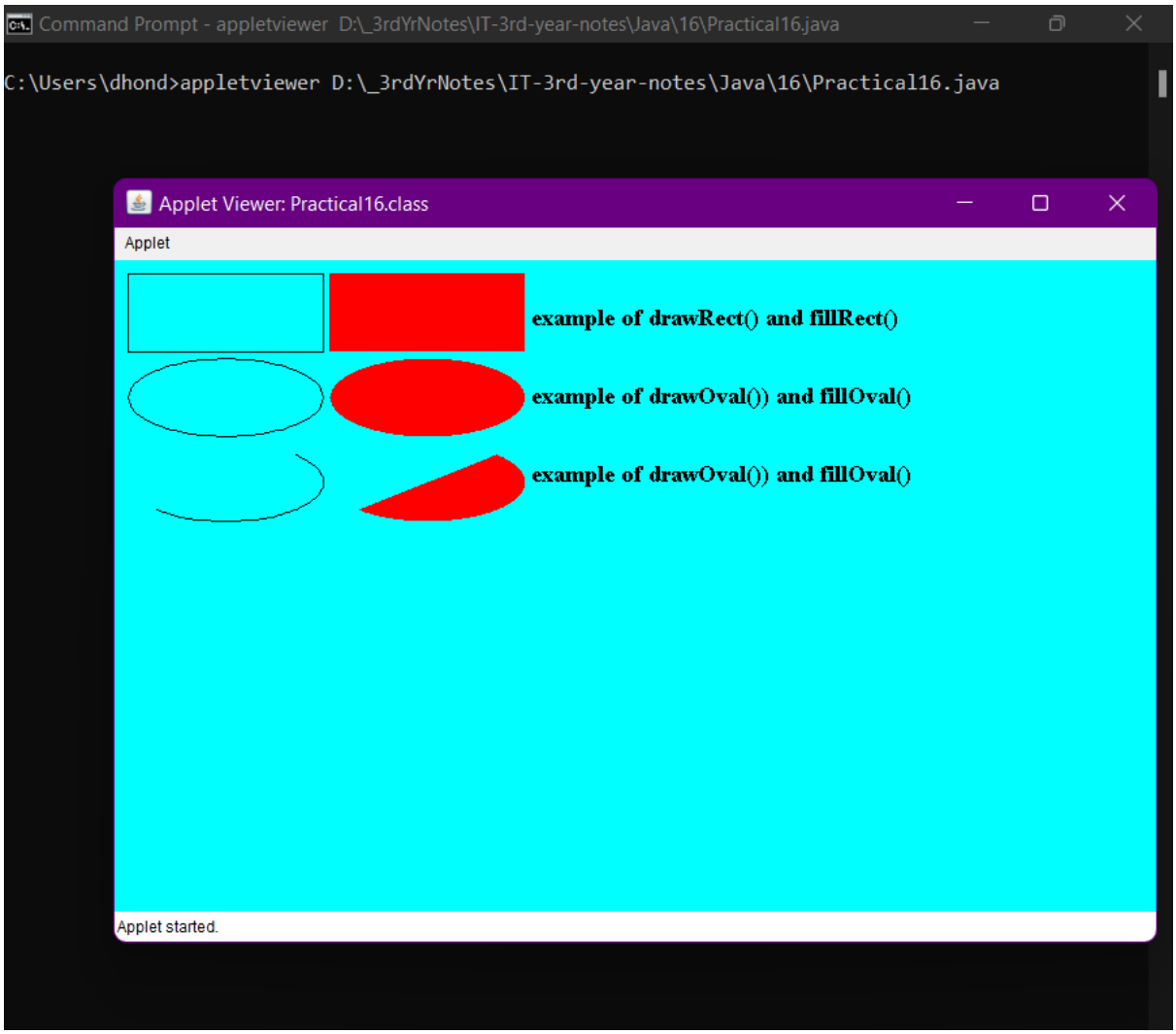
Hence, I created, debugged and executed Java programs based on graphics to draw, fill, different shapes.

Code:

Practical16.java

```
import java.awt.*;
import java.applet.*;
/*<applet code="Practical16.class" height="500" width="800"></applet>*/
public class Practical16 extends Applet{
  Font f1,f2;
  public void init(){
    setBackground(Color.CYAN);
    f2 = new Font("Times New Roman",Font.BOLD,18);
  }

  public void paint(Graphics g){
    f1 = g.getFont();
    g.drawRect(10,10,150,60);
    g.setColor(Color.RED);
    g.fillRect(165,10,150,60);
    g.setColor(Color.BLACK);
    g.setFont(f2);
    g.drawString("example of drawRect() and fillRect()",320,50);

    g.setColor(Color.BLACK);
    g.drawOval(10,75,150,60);
    g.setColor(Color.RED);
    g.fillOval(165,75,150,60);
    g.setColor(Color.BLACK);
    g.setFont(f2);
    g.drawString("example of drawOval()) and fillOval()",320,110);

    g.setColor(Color.BLACK);
    g.drawArc(10,140,150,60,45,-180);
    g.setColor(Color.RED);
    g.fillArc(165,140,150,60,45,-180);
    g.setColor(Color.BLACK);
    g.setFont(f2);
    g.drawString("example of drawOval()) and fillOval()",320,170);

  }

}
```

Output:



Applet using Graphics class

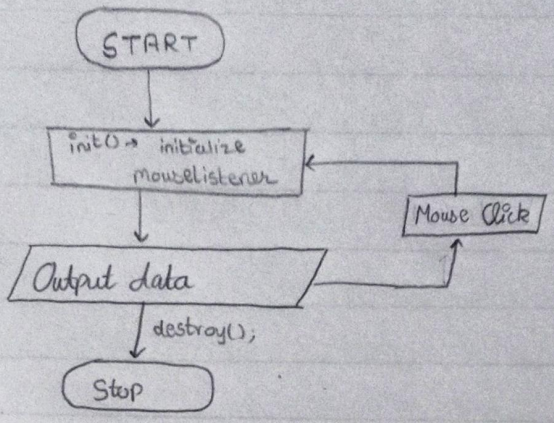
PAGE NO.
DATE

Conclusion:

Hence, I created, debugged and executed Java programs based on graphics to draw, fill, different shapes.

Practical No. 17

Aim : Create, debug and run Java programs based on mouse events and keyboard events.



PAGE NO.
DATE

Practical No. 17

Aim: Create, debug and run Java programs based on mouse events and keyboard events.

Theory:

What are Events in Java? (Event classes)?

→ • All the events in Java have corresponding event classes associated with them.

• Each of these classes is derived from the one single super class, i.e. EventObject, it is contained in the Java.util. package.

• The even object contains the following two important methods for handling events:

1) getSource();
2) toString();

Java Mouselistener Interface:

The java Mouselistener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener is found in java.awt.event package. It has five methods:

```
public abstract void mouseClicked(MouseEvents e);
public abstract void mouseEntered(MouseEvents e);
public abstract void mouseExited(MouseEvents e);
public abstract void mousePressed(MouseEvents e);
public abstract void mouseReleased(MouseEvents e);
```

Conclusion :

Hence, I created, debugged and executed java programs based on the concepts of mouseEvents and keyboardEvents.

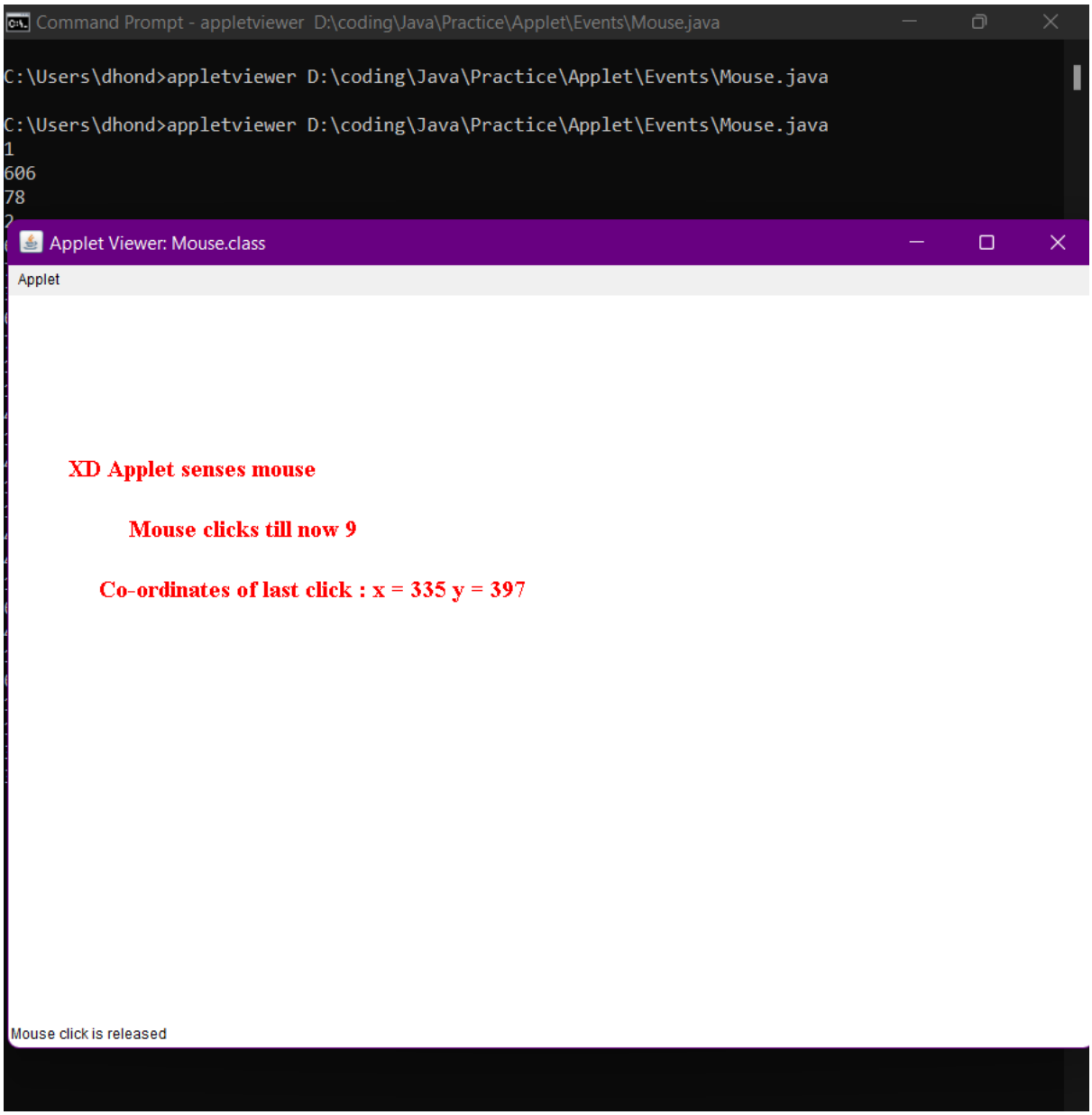
Code:

Mouse.java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

/*<applet code="Mouse.class" height="600" width="900"> </applet>*/

public class Mouse extends Applet implements MouseListener{
    int clicks = 0;
    int x = 0;
    int y = 0;
    public void init(){
        addMouseListener(this);
    }

    public void mouseClicked(MouseEvent mb){
        clicks++;
        System.out.println(mb.getClickCount());
        repaint();
        x = mb.getX();
        String str = String.valueOf(x);
        System.out.println(str);
        y = mb.getY();
        str = String.valueOf(y);
        System.out.println(str);
    }

    public void mousePressed(MouseEvent mb){
        showStatus("Mouse is clicked");

    }

    public void mouseReleased(MouseEvent mb){
        showStatus("Mouse click is released");
    }

    public void mouseEntered(MouseEvent mb){

    }

    public void mouseExited(MouseEvent mb){

    }
    Font f1 = new Font("Times New Roman",Font.BOLD,20);
```

```
    public void paint(Graphics g){
        g.setFont(f1);
        g.setColor(Color.RED);
        g.drawString("Applet senses mouse",50,150);
        g.drawString("Mouse clicks till now " + clicks, 100,200);
        g.drawString("Co-ordinates of last click : x = " + x + " y = " +
y,75,250);
    }


}

// comments
```

Output:



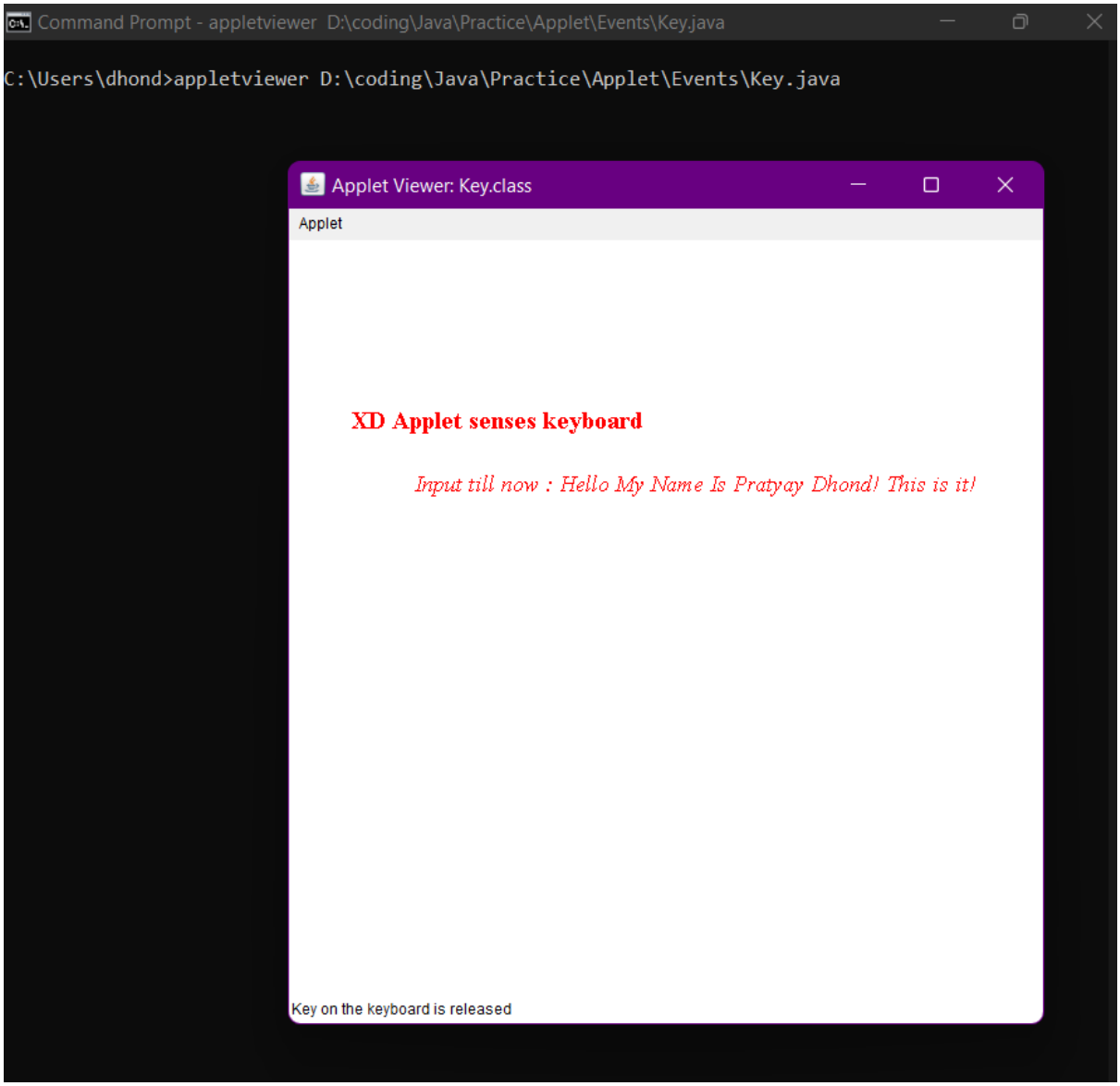
Applet using Mouse Event Listener

Code 2: KeyEventListener

Key.java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

/*<applet code="Key.class" height="600" width="600"> </applet>*/

public class Key extends Applet implements KeyListener{
    String typed = "";
    public void init(){
        addKeyListener(this);
    }

    public void keyTyped(KeyEvent kb){

        typed += kb.getKeyChar();
        repaint();
    }

    public void keyReleased(KeyEvent kb){
        showStatus("Key on the keyboard is released");
    }

    public void keyPressed(KeyEvent kb){
        showStatus("A key on the keyboard has been pressed");
    }

    Font f1 = new Font("Times New Roman",Font.BOLD,20);
    public void paint(Graphics g){
        g.setFont(f1);
        g.setColor(Color.RED);
        g.drawString("Applet Detects Keyboard",50,150);
        g.setFont(new Font("Serif",Font.ITALIC,18));
        g.drawString("Input till now : " + typed,100,200);
    }


}

// comments
```

Output:



Keyboard event in java output

PAGE NO.
DATE

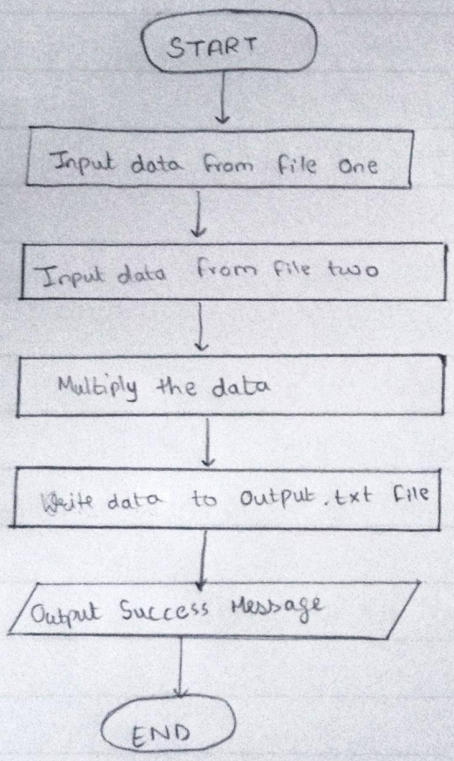
Conclusion:

Hence, I created, debugged and executed java programs based on the concepts of mouseEvents and KeyboardEvents.

Practical No. 18

Aim: Create, debug and run java program based on read and write characters from a file using input/output stream.

Flowchart:



PAGE NO.
DATE

Practical No. 18

Aim: Create, debug and run java programs based on read and write characters from a file using input/output stream.

Theory:

The File class from the java.io package, allows us to work with files.

To use Files, we need to create object of the File class, and specify the filename or directory name

Example,

```
import java.io.File;   OR   import java.io.*;
File File = new File("Filename.txt");
```

The File class contains the following methods:

Method	Return Value	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not.
createNewFile()	Boolean	Creates an empty file.
delete()	Boolean	Deletes a File
exists()	Boolean	Tests whether file exists
getName()	String	Returns the name of the File.
getAbsolutePath()	String	Returns the absolute pathname of the file.
length()	Long	Returns the size of files in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Create a directory.

Conclusion :

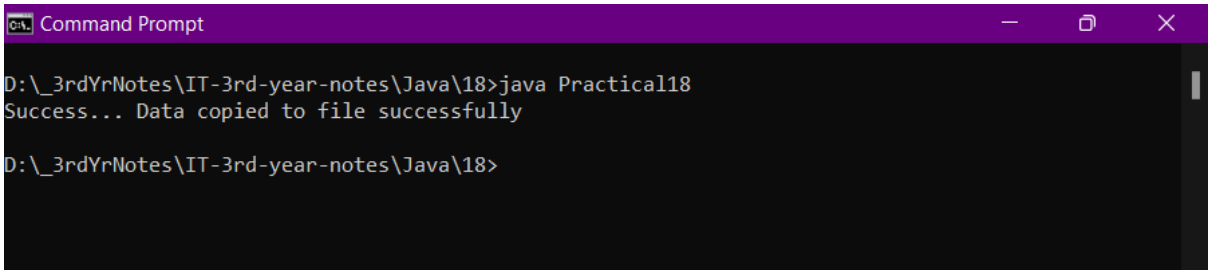
Hence, by performing this practical I get to know about the concepts of Files and performing I/O operations of them. I also created, debugged and executed Java programs based on reading and writing characters from a file using input/output stream.

Code:

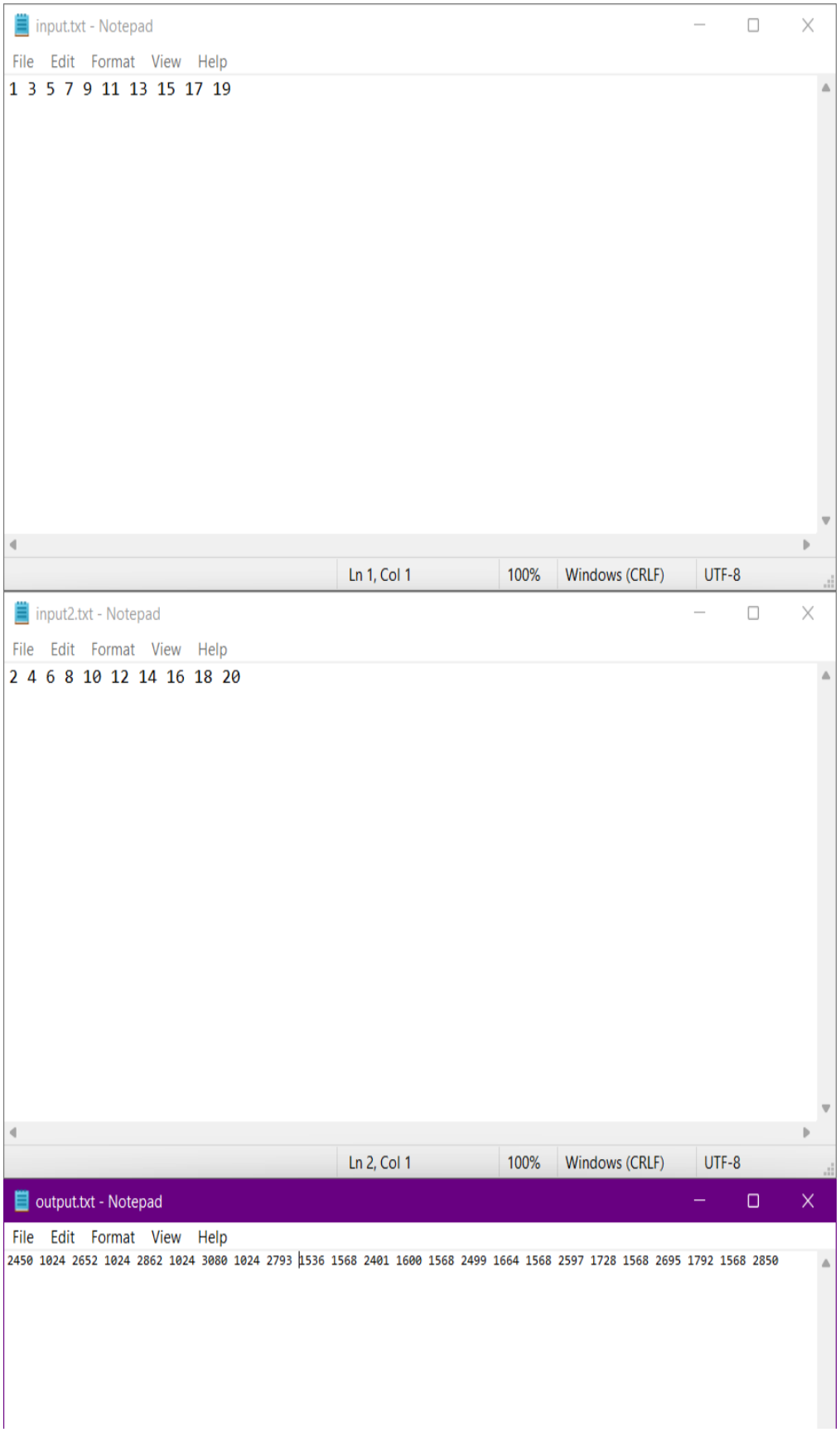
```
import java.util.*;
import java.io.*;

class Practical18{
    public static void main(String[] args){
        File input = new File("input.txt");
        File input2 = new File("input2.txt");
        File output = new File("output.txt");

            Vector<Integer> product = new Vector<Integer>();

        try{

            FileReader reader = new FileReader(input);
            Vector<Integer> v1 = new Vector<Integer>();
            Vector<Integer> v2 = new Vector<Integer>();
            int ch;
            int temp;
            while((temp = reader.read()) != -1){
                v1.add(temp);
            }
            reader.close();
            reader = new FileReader(input2);
            while((temp = reader.read()) != -1){
                v2.add(temp);
            }
            reader.close();

            int n = v1.size() > v2.size() ? v2.size() : v1.size();

            for(int i = 0; i < n; i++){
                product.add(v1.elementAt(i) * v2.elementAt(i));
            }

            if(v1.size() != v2.size()){
                if(v1.size() > v2.size()){
                    int l = v1.size() - n;
                    for(int i = n; i < l; i++){
                        product.add(v1.elementAt(i));
                    }
                }else{
                    int l = v2.size()-n;
                    for(int i = n; i < l; i++){
                        product.add(v2.elementAt(i));
                    }
```

```
                }
            }
        }catch(EOFException e){
            System.out.println(e);
        }catch(IOException e){
            System.out.println(e);
        }catch(Exception e){
            System.out.println(e);
        }

        try(FileWriter writer = new FileWriter("output.txt")) {
        for( int i=0; i<product.size(); ++i) {
            writer.write(product.elementAt(i)+" ");
        }
        writer.close();
        System.out.println("Success... Data copied to file successfully");
    }
    catch (Exception e ) {
        System.out.println(e);
    }

    }
}
```

Output:



Output on terminal



Input files and the output file

PAGE NO.
DATE

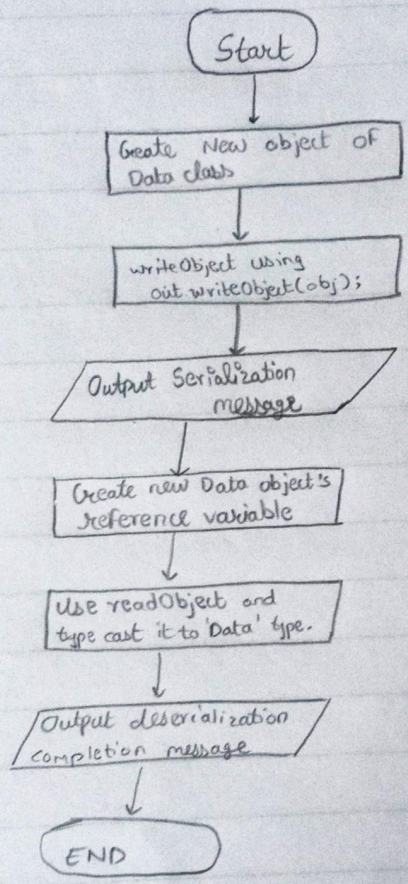
Conclusion :

Hence, by performing this practical I got to know about the concepts of Files and Performing I/O operations on them. I also created, debug and executed Java programs based on reading and writing characters from a file using input / output stream.

Practical No. 19

Aim : Create, debug and run java programs based on object Serialization.

flowchard :



PAGE NO.
DATE

Practical No. 19

Aim: Create, debug and run java programs based on Object Serialization.

Theory:

What is Object Serialization?

Object serialization is a mechanism provided in Java, using which an object can be represented as a sequence of bytes that includes the object's data as well as information about the object's type and the type's of data stored in the object.

After a serialized object has been written to a file, it can be read from a file and be deserialized, i.e. the type of information and bytes used to store the object can be used to recreate the object in memory.

Most impressive is that the entire process of serialization is JVM independent, that is, an object can be serialised on one platform and can be deserialized from a completely different platform.

The serialization and deserialization of objects is done through classes 'ObjectInputStream' and 'ObjectOutputStream' which are high level streams which contains the methods.

The method used to Serialize an object:

public final void writeObject(Object x) throws IOException

The method used to deserialise an object

public final Object readObject(Object x) throws IOException, ClassNotFoundException.

Conclusion :

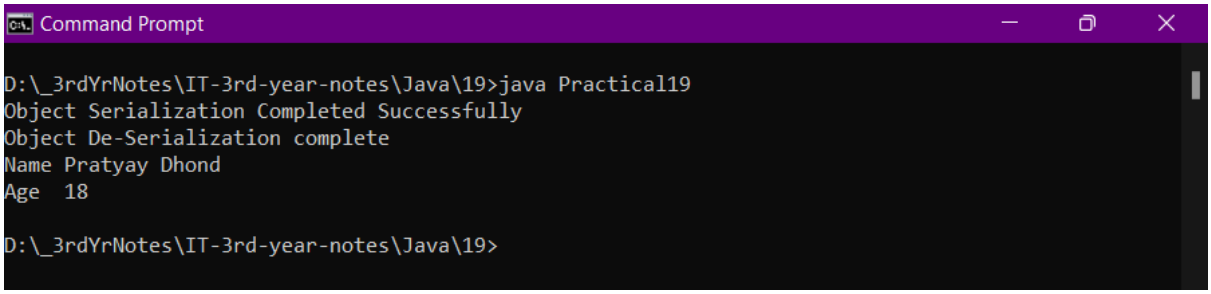
Hence, I create, debugged and executed java programs based on object serialization.

Code:

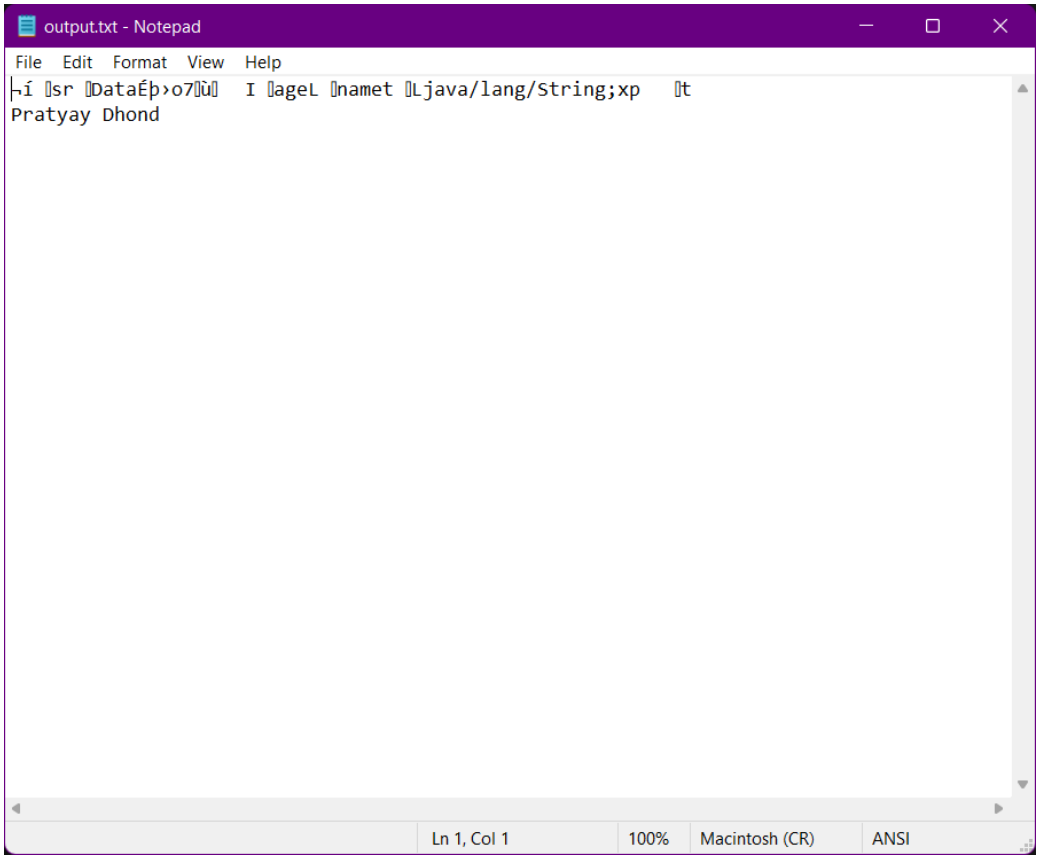
```
import java.io.*;
import java.io.Serializable;

class Data implements java.io.Serializable{
  public String name;
  public int age;

  public Data(String name, int age){
    this.name = name;
    this.age = age;
  }
}

class Practical19{
  public static void main(String[] args){

    Data obj = new Data("Pratyay Dhond",18);
    String filename = "output.txt";

    try{
      FileOutputStream file = new FileOutputStream(filename);
      ObjectOutputStream out = new ObjectOutputStream(file);

      out.writeObject(obj);
      out.close();
      file.close();

    System.out.println("Object Serialization Completed Successfully");

    }catch(Exception e){
          System.out.println("Error : Object Serialization Failed");
          System.out.println("Error : " + e);
    }

    try{
      FileInputStream file = new FileInputStream(filename);
      ObjectInputStream in = new ObjectInputStream(file);

      Data from_file = null;

      from_file = (Data) in.readObject();
      in.close();
      file.close();


      System.out.println("Object De-Serialization complete");
      System.out.println("Name " + from_file.name);
```

```
      System.out.println("Age  " + from_file.age);
    }catch(Exception e){
            System.out.println("Error : Object De-Serialization failed...");
                  System.out.println("Error : " + e);

    }


  }
}
```

Output:



Output on terminal



Serialised data stored in file

PAGE NO.
DATE

Conclusion:

Hence, I created, debugged, and executed java programs based on object serialization.

Practical No. 20

Aim : Mini project - Painting Applet

PAGE NO.
DATE

Practical No. 20

Aim : Mini project → Painting Applet.

Theory:

Painting Applet :

The aim of this project is to enable the user to draw, erase and create, drawings, paintings, art, etc. This applet can also be used as a whiteboard.

The applet consists of 7 color options ~~colors~~ and 1 clear button and a x time indicator in the status bar which is x updated after every 1000 ms, using threads.

The color buttons are rectangles of approximately 12.35 % height of the applet height so that all color rectangles and clear button can fit on the right ~~hand~~ hand side of the applet.

The timer on the bottom-left i.e. the status bar uses the Calendar class to get the current hour of the day, minute of the day, and second of the day. This data is converted into 'HH:MM:SS' format and saved as string in a ~~variable~~ variable called timestring, which is passed to status ~~bar~~ bar, then the thread goes to sleep for 1000 ms, by the use of wait(1000); call.

The drawing on the applet is happening by the use of mousedragged() motion listener, to get the x axis and y axis, this x and y axis is used to draw an oval in the current color and width.

Conclusion:

Hence, we created our mini project and used various java concepts in it to make the paint applet work.

Code:

SimplePaint.java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

/*<applet code="SimplePaint.class" height="1080" width="1920">  */
public class SimplePaint extends Applet implements MouseListener,
MouseMotionListener, KeyListener,Runnable {
    private int currentColor = BLACK;

    private final static int
                BLACK = 0,
                RED = 1,
                GREEN = 2,
                BLUE = 3,
                CYAN = 4,
                MAGENTA = 5,
                YELLOW = 6,
                 WHITE = 7;

    int STROKE = 2;

    Thread t = null;
    int hours = 0, minutes = 0, seconds = 0;
    String timeString = "";

    private int prevX, prevY; // previous values clicked/ dragged by users

    private boolean dragging;    // checking if the user is currently dragging
the mouse or not

    private Graphics g;

   public void init() {
      addMouseListener(this);
      addMouseMotionListener(this);
      addKeyListener(this);
    }
```

```
public void update(Graphics g) {
   paint(g);
}

public void paint(Graphics g) {

  int width = getSize().width;
  int height = getSize().height;

  int colorSpacing = (height - 56) / 8;


  //drawing screen
  g.setColor(Color.white);
  g.fillRect(3, 3, width - 59, height - 6);


  //backgroundscreen
  g.setColor(Color.gray);
  g.drawRect(0, 0, width-1, height-1);
  g.drawRect(1, 1, width-3, height-3);
  g.drawRect(2, 2, width-5, height-5);


  g.fillRect(width - 56, 0, 56, height);


  g.setColor(Color.white);
  g.fillRect(width-53,  height-53, 50, 50);
  g.setColor(Color.black);
  g.drawRect(width-53, height-53, 49, 49);
  g.drawString("CLEAR", width-48, height-23);

    // For color button colors
  g.setColor(Color.black);
  g.fillRect(width-53, 3 + 0*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.red);
  g.fillRect(width-53, 3 + 1*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.green);
  g.fillRect(width-53, 3 + 2*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.blue);
  g.fillRect(width-53, 3 + 3*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.cyan);
  g.fillRect(width-53, 3 + 4*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.magenta);
  g.fillRect(width-53, 3 + 5*colorSpacing, 50, colorSpacing-3);
  g.setColor(Color.yellow);
  g.fillRect(width-53, 3 + 6*colorSpacing, 50, colorSpacing-3);
```

```
  // BOOKMARK
    g.setColor(Color.white);
    g.fillRect(width-53, 3 + 7*colorSpacing, 50, colorSpacing-3);

    //for background of clear button
  g.setColor(Color.white);
  g.drawRect(width-55, 1 + currentColor*colorSpacing, 53, colorSpacing);
  g.drawRect(width-54, 2 + currentColor*colorSpacing, 51, colorSpacing-2);

  g.setColor(Color.BLACK);
  g.drawString("ERASE", width-48, height-105);

}

private void changeColor(int y) {

// border of the selected color
  int width = getSize().width;
  int height = getSize().height;
  int colorSpacing = (height - 56) / 8;
  int newColor = y / colorSpacing;

  if (newColor < 0 || newColor > 7)
     return;

  Graphics g = getGraphics();
  g.setColor(Color.gray);
  g.drawRect(width-55, 1 + currentColor*colorSpacing, 53, colorSpacing);
  g.drawRect(width-54, 2 + currentColor*colorSpacing, 51, colorSpacing-2);
  currentColor = newColor;
  //current border set to white
  g.setColor(Color.white);
  g.drawRect(width-105, 1 + currentColor*colorSpacing, 53, colorSpacing);
  g.drawRect(width-54, 2 + currentColor*colorSpacing, 51, colorSpacing-2);
  g.dispose();

}

public void start ()
{
    t = new Thread (this);
    t.start ();
}

public void run ()
{
```

```
        try
        {


            while (true)
            {
                String temp;
                Calendar cal = Calendar.getInstance ();
                hours = cal.get (Calendar.HOUR_OF_DAY);
                minutes = cal.get (Calendar.MINUTE);
                seconds = cal.get (Calendar.SECOND);
                 if (hours > 12){
                    temp = " PM";
                    hours -= 12;
                 }else{
                    temp = " AM";
                 }
                SimpleDateFormat formatter = new SimpleDateFormat("hh:mm:ss");
                Date date = cal.getTime ();
                timeString = formatter.format (date);
                timeString += temp;
                t.sleep (1000); // interval given in milliseconds
                showStatus(timeString);
            }
        }
        catch (Exception e)
        {
        }
    }

    private void setUpDrawingGraphics() {

      g = getGraphics();
      switch (currentColor) {
         case BLACK:
            g.setColor(Color.black);
            break;
         case RED:
            g.setColor(Color.red);
            break;
         case GREEN:
            g.setColor(Color.green);
            break;
         case BLUE:
            g.setColor(Color.blue);
            break;
         case CYAN:
            g.setColor(Color.cyan);
```

```
            break;
         case MAGENTA:
            g.setColor(Color.magenta);
            break;


         case YELLOW:
            g.setColor(Color.yellow);
            break;
          case WHITE:
              g.setColor(Color.white);
              break;
      }
   }

   public void mousePressed(MouseEvent evt) {


      int x = evt.getX();
      int y = evt.getY();

      int width = getSize().width;
      int height = getSize().height;

      if (dragging == true)
          return;

      if (x > width - 53) {

         if (y > height - 53)
            repaint();
         else
            changeColor(y);
      }
      else if (x > 3 && x < width - 56 && y > 3 && y < height - 3) {

         prevX = x;
         prevY = y;
         dragging = true;
         setUpDrawingGraphics();
      }

   }     // end mousePressed()

   public void mouseReleased(MouseEvent evt) {

       if (dragging == false)
          return;  // Nothing to do because the user isn't drawing.
```

```
        dragging = false;
        g.dispose();
        g = null;
    }

    public void mouseDragged(MouseEvent evt) {


        if (dragging == false)
            return;  // Nothing to do because the user isn't drawing.

        int x = evt.getX();   // x-coordinate of mouse.
        int y = evt.getY();   // y=coordinate of mouse.

        if (x < 3)                          // Adjust the value of x,
            x = 3;                          //   to make sure it's in
        if (x > getSize().width - 57)       //   the drawing area.
            x = getSize().width - 57;

        if (y < 3)                          // Adjust the value of y,
            y = 3;                          //   to make sure it's in
        if (y > getSize().height - 4)       //   the drawing area.
            y = getSize().height - 4;

//        g.drawLine(prevX, prevY, x, y);  // Draw the line.
        g.fillOval (prevX, prevY, STROKE, STROKE);
        g.fillOval (x, y, STROKE, STROKE);

        prevX = x;  // Get ready for the next line segment in the curve.
        prevY = y;

    } // end mouseDragged.

     public void keyTyped(KeyEvent evt) {
        char c = evt.getKeyChar();
        switch (c){
            case '1':
                STROKE = 1;
                break;
            case '2':
                STROKE = 2;
                break;
            case '3':
                STROKE = 3;
                break;
            case '4':
                STROKE = 4;
                break;
```

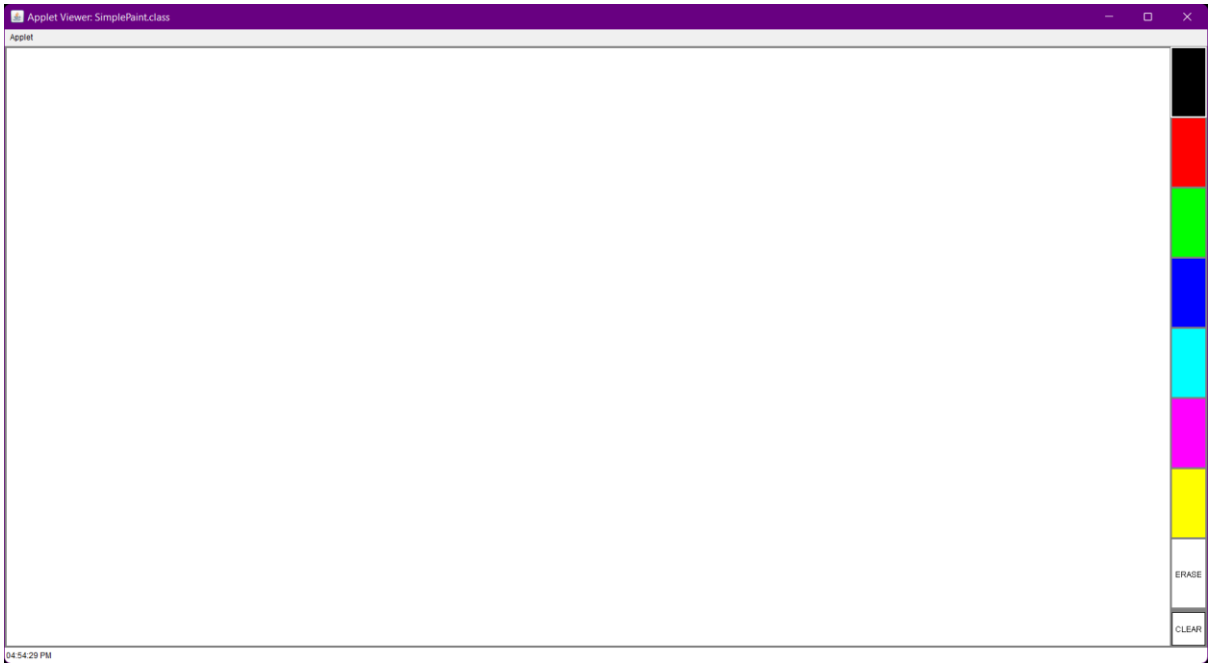
```
            case '5':
                STROKE = 5;
                break;
            case '6':
                STROKE = 6;
                break;


            case '7':
                STROKE = 7;
                break;
            case '8':
                STROKE = 8;
                break;
            case '9':
                STROKE = 9;
                break;
            case 'a':
                if(STROKE==1){}
                else {
                    STROKE -= 1;
                }
                break;
            case 'd':
                STROKE += 1;
                break;
            case 'w':
                if(currentColor <= 0 ){
                    currentColor=6;
                }else{
                    currentColor-=1;
                }
                break;
            case 's':
                currentColor = (currentColor+1) % 7;
                break;
            default:
                return;
        }
    }

    public void keyPressed(KeyEvent evt) {
        char c = evt.getKeyChar();
        switch (c){
            case 'a':
                if(STROKE==1){}
                else {
                    STROKE -= 1;
```

```
                }
                break;
            case 'd':
                STROKE += 1;
                break;
            default:
                return;
        }


        System.out.println(STROKE);
    }

    public void keyReleased(KeyEvent evt) { }
    public void mouseEntered(MouseEvent evt) { }   // Some empty routines.
   public void mouseExited(MouseEvent evt) { }      //  (Required by the
MouseListener
   public void mouseClicked(MouseEvent evt) { }   //    and
MouseMotionListener
   public void mouseMoved(MouseEvent evt) { }     //    interfaces).
} // end class SimplePaint
```

SimplePaint.html

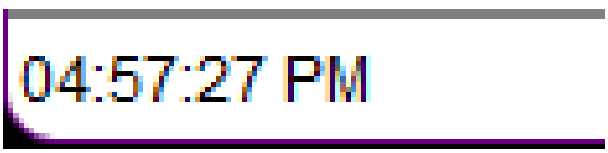
```
<html>

<applet code="SimplePaint.class" height="900" width="1800" >  </applet>

</html>
```

Output:



Painting Applet First View

Drawing in the applet



Live time in the applet implemented by using multi-threading concept

PAGE NO.
DATE

Conclusion:

Hence, we created our miniproject and used various java projects concepts in it to make the paint applet work.