|  |  |  |
| --- | --- | --- |
| Sl. No. | **Particulars** | **Page No**. |
| **1** | **Introduction** | 5 |
| **2** | **Algorithm / Flowchart / Procedure** | 6 |
| **3** | **Requirements (Hardware and Software)** | 7 |
| **4** | **Implementation** | 7 |
| **5** | **Results** | 13 |
| **6** | **Conclusion & References** | 14 |

**INTRODUCTION**

* This project aims at building a JAVA program that helps in attendance management.
* This program takes the attendance of a student subject-wise as input.
* It mainly consists of GUI that provides spaces for entering data, buttons and interactive elements that can be used to get desired results.
* An attendance management system is pretty useful as it makes it easy to filter students based on their attendance and also helps in finding out te students who aren’t eligible to attend a particular exam or don’t meet requirements of a event.
* The Java programming language class library provides a user interface toolkit called the Abstract Windowing Toolkit, or the AWT.
* The AWT is both powerful and flexible. Newcomers, however, often find that its power is veiled. The class and method descriptions found in the distributed documentation provide little guidance for the new programmer.
* Furthermore, the available examples often leave many important questions unanswered. Of course, newcomers should expect some difficulty. Effective graphical user interfaces are inherently challenging to design and implement, and the sometimes complicated interactions between classes in the AWT only make this task more complex.
* However, with proper guidance, the creation of a graphical user interface using the AWT is not only possible, but relatively straightforward.
* The user interface is that part of a program that interacts with the user of the program. User interfaces take many forms. These forms range in complexity from simple command-line interfaces to the point-and-click graphical user interfaces provided by many modern applications.
* A graphical user interface is built of graphical elements called components. Typical components include such items as buttons, scrollbars, and text fields. Components allow the user to interact with the program and provide the user with visual feedback about the state of the program. In the AWT, all user interface components are instances of class Component or one of its subtypes.

**ALGORITHM**

* **STEP 1 :** We have to import the necessary packages and classes that are used in the creation of the GUI .
* **STEP 2 :** Create a class that extends “Applet” class and implements “ActionListener” interface.
* **STEP 3 :** Create reference variables of the various components that are to be used in the container(here it is the Applet).
* **STEP 4 :** Invoke the constructors for the all the reference variables created to create objects for the same components.
* **STEP 5 :** Use the add() method to add those components into the container(here it is the Applet).
* **STEP 6 :** Define the method “actionPerformed()” by coding the logic for the buttons present in the container.
* **STEP 7 :** Make sure that try catch blocks are used in order to ensure the exception handling.
* **STEP 8 :** Execute the program by giving the inputs into the text fields and pressing the button that has to be pressed.

**REQUIREMENTS**

1. A computer/Laptop compatible with NetBeans
2. AppletViewer
3. NetBeans IDE
4. JAVA JDK
5. JAVA JRE

**IMPLEMENTATION / SOURCE CODE**

import java.applet.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Micro extends Applet implements ActionListener{

Button x,y;

Label p,q,r,s;

TextField a,b,c,d,e;

String m;

@Override

public void init(){

x = new Button("AVG");

x.addActionListener(this);

y = new Button("CONDITION");

y.addActionListener(this);

a = new TextField(20);

b = new TextField(20);

c = new TextField(20);

d = new TextField(20);

e = new TextField(50);

p = new Label("Attendance in DM");

q = new Label("Attendance in DBMS");

r = new Label("Attendance in JAVA");

s = new Label("Attendance in DLD");

add(p);

add(a);

add(q);

add(b);

add(r);

add(c);

add(s);

add(d);

add(e);

add(x);

add(y);

}

@Override

public void actionPerformed(ActionEvent ae){

if(ae.getSource()==x){

try{

float l = Float.parseFloat(a.getText());

float m = Float.parseFloat(b.getText());

float n = Float.parseFloat(c.getText());

float o = Float.parseFloat(d.getText());

float f = (l+m+n+o)/4;

e.setText(""+f);

}

catch(NumberFormatException e){

System.out.println("EXCEPTION ARISED");

}

}

else if(ae.getSource()==y){

float l = Float.parseFloat(a.getText());

float m = Float.parseFloat(b.getText());

float n = Float.parseFloat(c.getText());

float o = Float.parseFloat(d.getText());

if(l<75.0){

e.setText("YOUR ATTENDANCE IN DM IS LOW!!!");

}

else if(m<75.0){

e.setText("YOUR ATTENDANCE IN DBMS IS LOW!!!");

}

else if(n<75.0){

e.setText("YOUR ATTENDANCE IN JAVA IS LOW!!!");

}

else if(o<75.0){

e.setText("YOUR ATTENDANCE IN DLD IS LOW!!!");

}

else {

e.setText("YOUR ATTENDANCE IS GOOD");

}

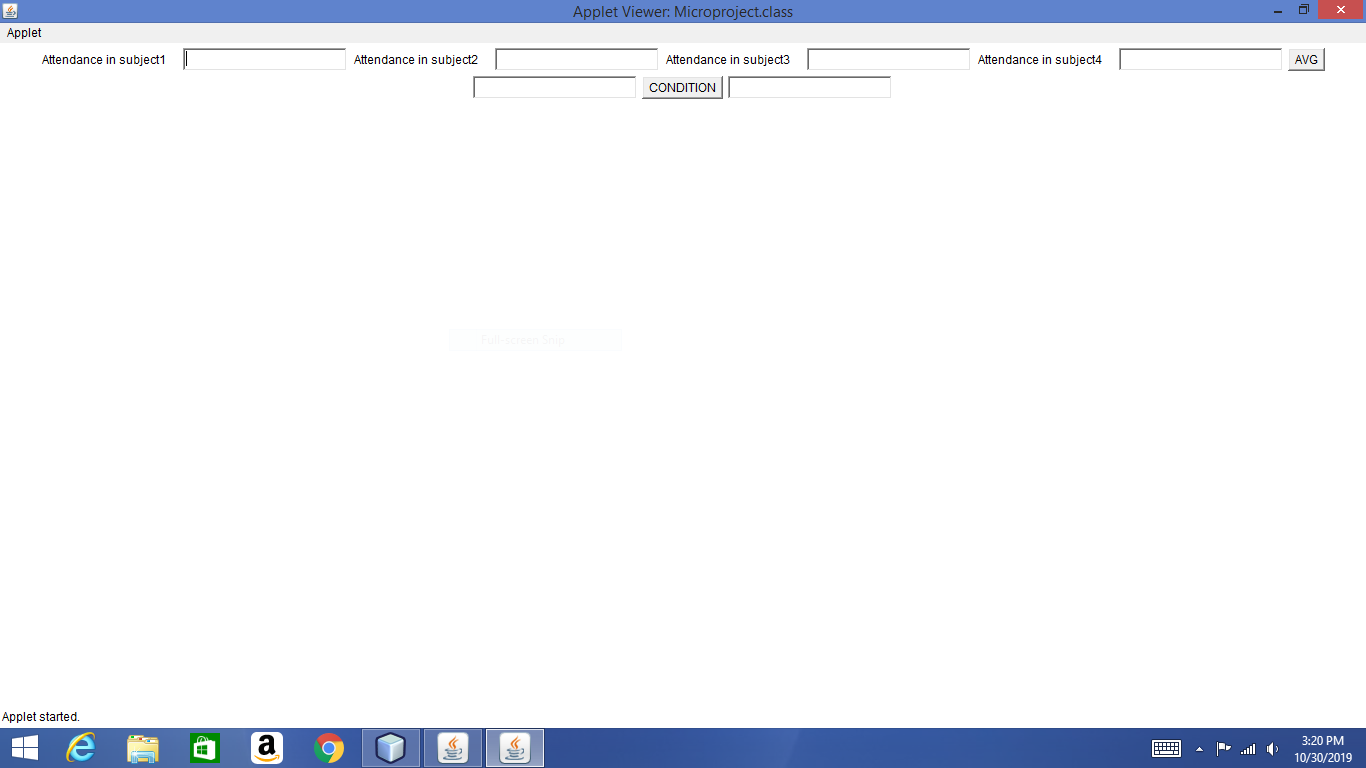
}

}

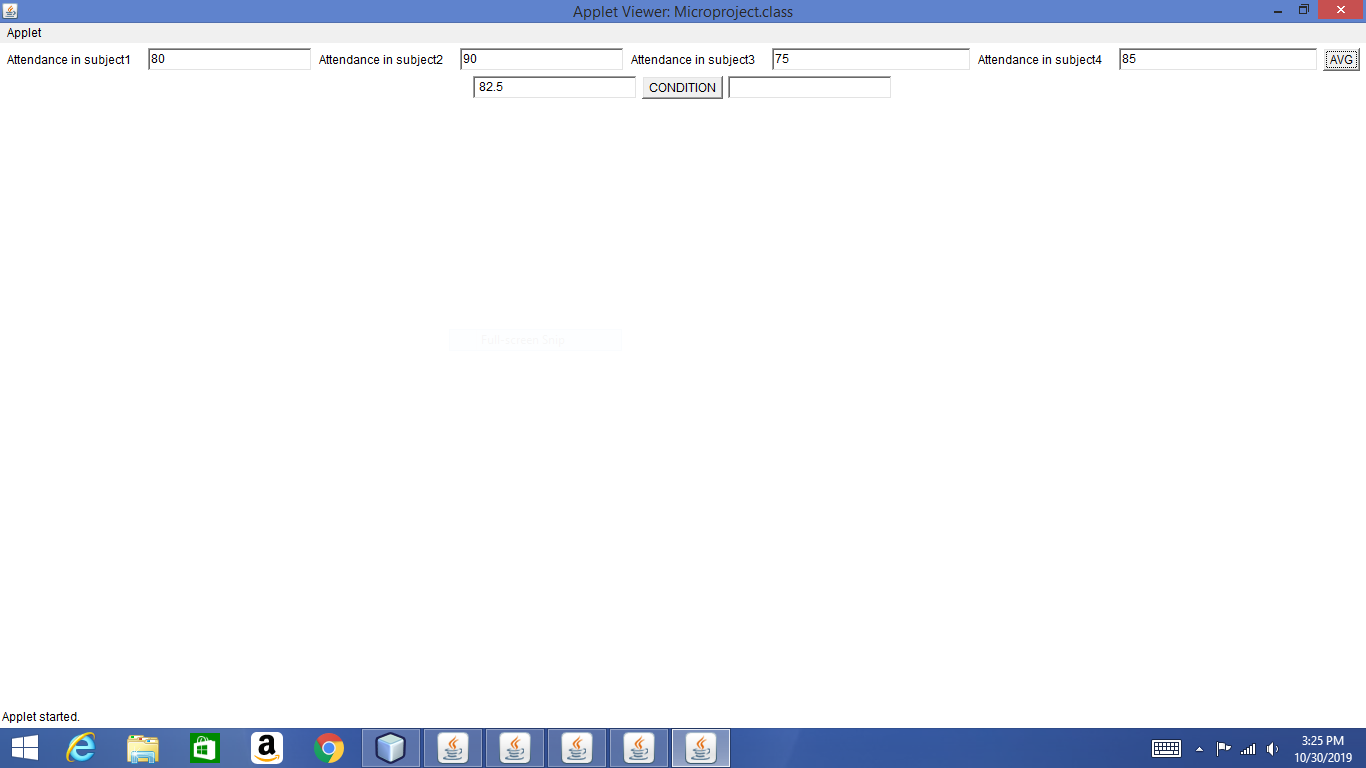
}

**RESULTS**

* The applet output GUI.



* After giving input.



* Shows in which subject the attendance is less than 75%.



**CONCLUSION**

**REFERENCES**

1. Java the complete reference, 8th Editon, Herbert Schildt, TMH.
2. Java How to Program, H. M. Dietel and P. J. Dietel, 6th Edition, Pearson Education/PHI.
3. Introduction to Java programming, Y. Daniel Liang, Pearson Education.