Panneerselvam. C
19TD0194
Computer Science
and Engineering
II/IV/A

01/06/2021

Programming in JAVA

Version:3

CONTINUOUS ASSESSMENT TEST-3 [CAT-3]

PART-A

1) Collections in Java:

* A collection is a framework that provides a well-defined architecture to store and manipulate the group of objects.
* It is used to achieve operations such as searching, sorting, insertion, manipulation and deletion
* It provides many interfaces (Set, List, Queue) and classes (Arraylist, LinkedList, TreeSet, etc...)

2) **Byte Stream:**

* These streams are used to read bytes from the input stream and write bytes in the output stream.
* In other words, byte streams are used to read or write the data of 8-bits.

3)

Application	Applet
i) Contains main method	Does not contain main method
ii) Does not require internet connection.	Requires Internet connection
iii) Stand alone application	Not a stand alone application
iv) Can run without browser	Can run only with browser

4) States to follow in JDBC:

* Import JDBC packages
* Load and register JDBC driver
* Open a connection to Database
* Create a statement object and perform a query
* Execute the statement object and return a query resultset.
* Process the resultset
* close the resultsit and statement objects
* close the connection

5)

AWT	Swings
i) Platform dependent	Platform independent
ii) Heavyweight	Lightweight
iii) Doesn't support pluggable look and feel.	Supports pluggable look and feel
iv) Less components than swing	More component than AWT
v) Doesn't follow MVC (Model View Component)	Follows MVC

PART-C

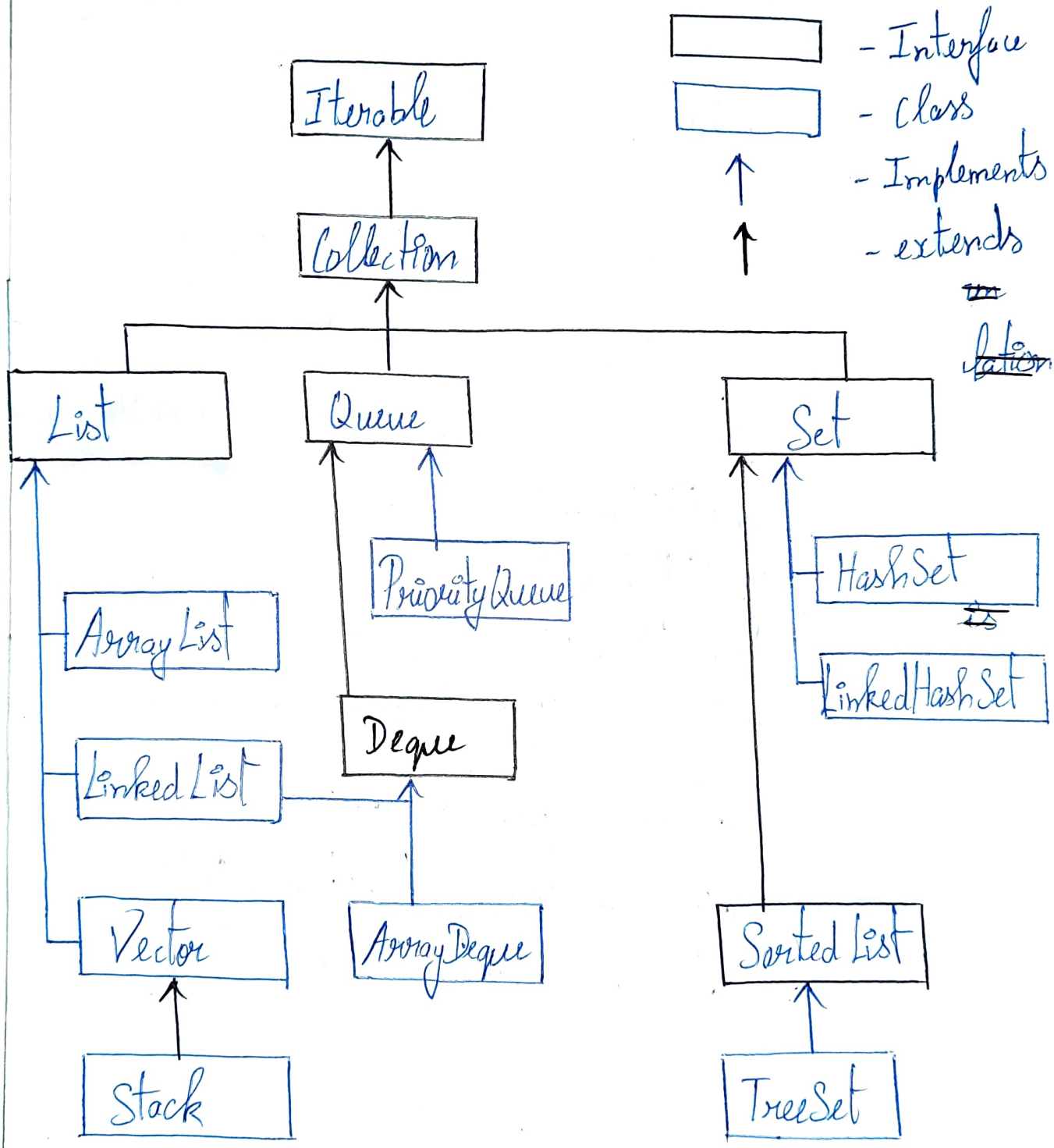
1) **COLLECTIONS IN JAVA:**

* The collections in Java is a framework that provides an architecture to store and manipulate the group of objects.
* Java collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation and deletion
* Java collection means a single unit of objects.
* Java collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet)

→ Collection represents a single unit of objects, ie a group

What is a Framework in Java

* It provides readymade architecture
* It represents a set of classes and interfaces



Example:

```
import java.io.*;
import java.util.*;
class Test
{
  public static void main (String[] args)
  {
  int arr[] = new int[] {1, 2, 3, 4};
  Vector<Integer> v = new Vector();
  Hashtable <Integer, String> h = new Hashtable();
  v.addElement(1);
  v.addElement(2);
  h.put(1, "geeks");
  h.put(2, "geeks");

  System.out.println(arr[0]);
  System.out.println(v.elementAt(0));
  System.out.println(h.get(1));
  }
}
```

Output:

```
1
1
Geek
```

2) Calculator using GUI:

```
import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class Calculator implements ActionListener {
    private static JTextField inputBox;

Calculator() { }
    public static void main(String[] args) {
    createWindow();

}
    private static void createWindow() {
JFrame frame = new JFrame("Calculator");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
createUI(frame);
frame.setSize(200,200);
frame.setLocationRelativeTo(null);
frame.setVisible(true);
}
private static void createUI(JFrame frame){
 JPanel = new JPanel();
 Calculator calculator = new Calculator()
 GridBagLayout layout = new GridBagLayout();
 GridBagConstraints gbc = new GridBagConstraints();
 panel.setLayout(layout);
 inputBox = new JTextField(10);
 inputBox.setEditable(false);
 JButton button0 = new JButton("0");
 JButton button1 = new JButton("1");
 JButton button2 = new JButton("2");
 JButton button3 = new JButton("3");
 JButton button4 = new JButton("4");
 JButton button5 = new JButton("5");
 JButton button6 = new JButton("6");
```

```
JButton button7 = new JButton("7");
JButton button8 = new JButton("8");
JButton button9 = new JButton("9");

JButtonPlus = new JButton("+"); JButtonMinus = new JButton("-");
JButtonDivide = new JButton("/"); JButtonMultiply
                                  = new JButton("x");

JButton buttonClear = new JButton("C"); JButton buttondot
                                  = new JButton(".");

JButton buttonEquals = new JButton("=");

public void actionPerformed (ActionEvent e) {
    String command = e.getActionCommand();
    if (command.charAt(0) == 'C') {
     inputBox.setText("");
    } else if (command.charAt(0) == '=') {
      inputBox.setText (inputBox.getText() = command);
    }
}
```

```
public static String evaluate (String expression)
{
 char[] arr = expression . toCharArray();
 String operand1 = " "; String operand2 = " ";
 string operator = " "; double result = 0;
 for( int i = 0 ; i < arr.length; i++){
   if(arr[i] >= '0' && arr[i] <= '9' || arr[i] == '.'){
      if(operator.isEmpty()){
        operand1 += arr[i];
      } else {
          operand2 += arr[i];
      }
   }
   if (arr[i] == '+' || arr[i] == '-' || arr[i] == '/' ||
      arr[i] == '*'){
      operator += arr[i];
   }
 }
if(operator.equals("+"))
   result = (Double.parseDouble(operand1) + Double.parseDouble
                                              (operand2));
else if (operator.equals("-"))
   result = (Double.parseDouble(operand1) - Double.parseDouble(operand
                                              2));
```

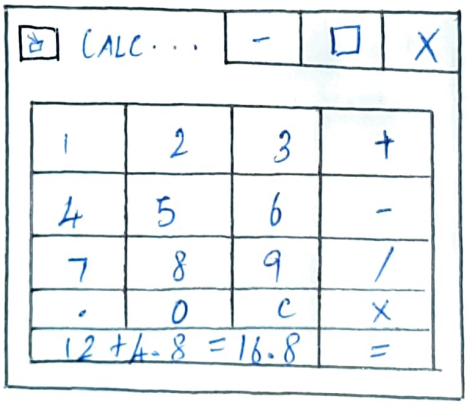
```
else if (operator.equals("/"))
    result = (Double.parseDouble(operand1) / Double.parseDouble
                                                (operand2));

else
    result = (Double.parseDouble(operand1) * Double.parseDouble
                                                ((operand2));

    result operand1 + operator + operand2 + "=" + result;

    }
}
```

Output:



PART-B:

1) List Interface - Collection Framework [JAVA]:

List Interface:

* List Interface is the child interface of Collection interface.
* It inhibits a list type data structure in which we can store the ordered collection of objects
* It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

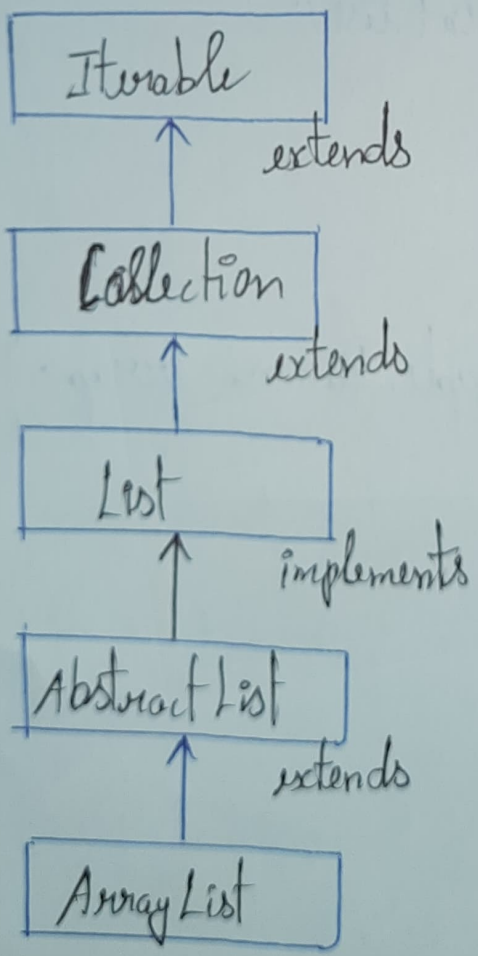
To instantiate the List interface, we must use:

1. List <data-type> list1 = new ArrayList();
2. List <data-type> list2 = new LinkedList();
3. List <data-type> list3 = new Vector();
4. List <data-type> list4 = new Stack();

There are various methods in List interface that can be used to insert, delete, and access the elements from the list

Array List:

* The Array List class implements the List interface
* It uses a dynamic array to store the duplicate element of different data types
* The Array List class maintains the insertion order and is non-synchronized.
* The elements stored in the Array List class can be randomly accessed



Java ArrayList Example:

```
import java.util.*;
public class ArrayListExample {
public class void main(String args[]) {
  ArrayList<String> list = new ArrayList<String>();
  list.add("Mango");
  list.add("Apple");
  list.add("Banana");
  list.add("Grapes");
  System.out.println(list);
  }
}
```

Output:

[Mango, Apple, Banana, Grapes]

2) Java I/O Stream:

Java I/O (Input and Output) is used to process the input and produce the output.

Stream: A stream is a sequence of data. In Java, a stream is composed of bytes

Inputstream:

Java application uses an input stream to read data from a source, it may be a file, an array, peripheral device or socket.

Java Inputstream class

Java FileInputStream class obtain input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image, data, audio, video, etc..

You can also read character-stream data

Java FileInputStream : read single character

```
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.println((char)i);
    fin.close();
    } catch (Exception e) { System.out.println(e);}
    }
}
```

Java FileOutputStream:

Java application uses an output stream to write data to a destination; it may be a file, an array, peripheral device or socket.

Java FileOutputstream Class:

* Java FileOutputStream is an output stream used write data to a file
* If you have to write primitive values into a file, uses FileOutputStream class
* You can write byte-oriented as well as character-oriented data through FileOutputStream class
* But for character data, it is preferred to use FileWriter than FileOutputstream

Java FileOutputStream : write byte

```
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);}
}
}
```

Output:

Success...

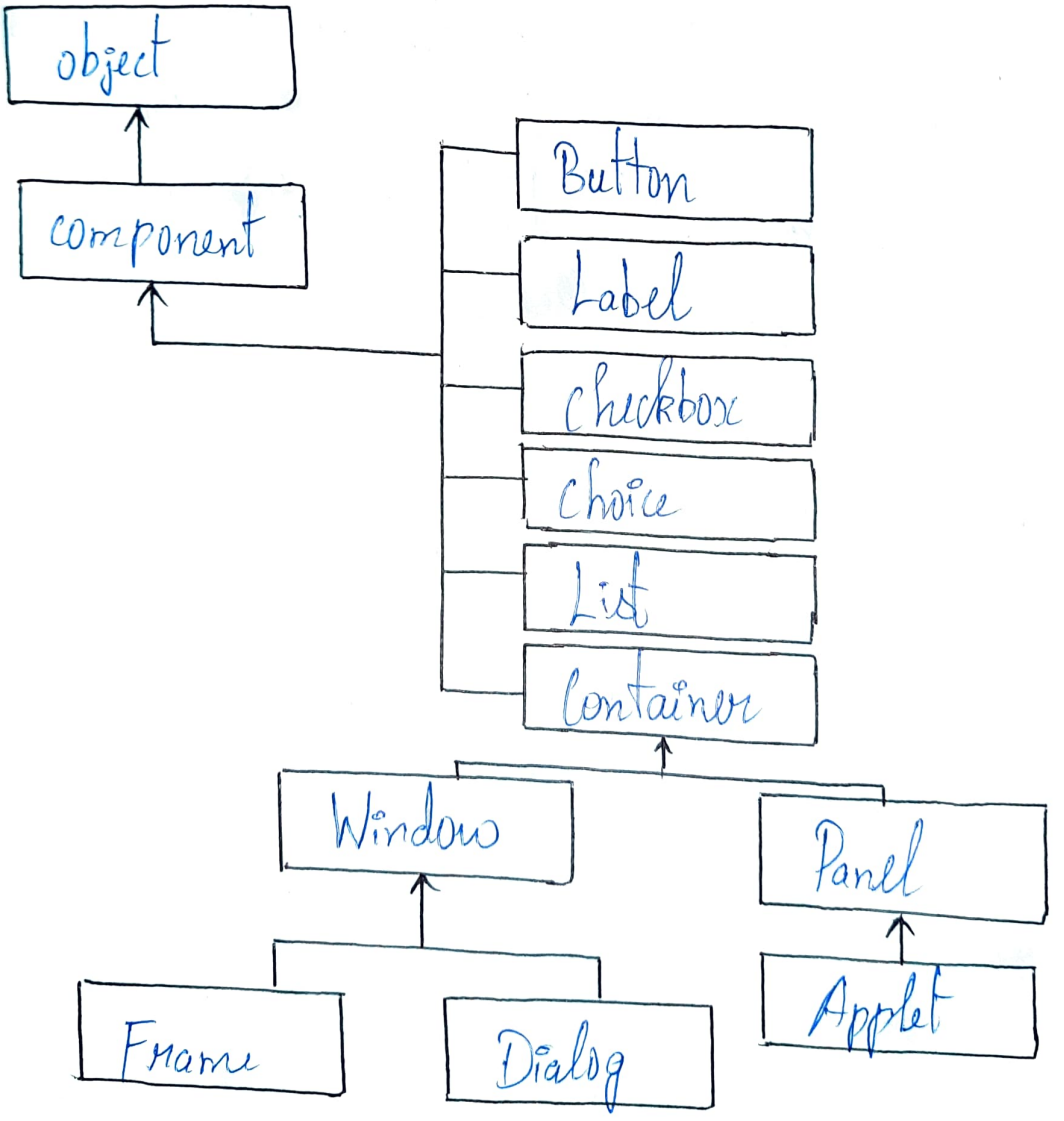
The content of a text file testout.txt is set with the data A.

testout.txt

A

3) Component and container class:

* Component and containers come under Java AWT controls
* Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in JAVA
* Java AWT components are platform-dependents (ie) components are displayed according to the view of operating system.



Useful Methods of Component class

Method	Description
public void add(component c)	Inserts a component on this component
public void setSize(int width, int height)	sets the size(width and height) of the component
public void setLayout(LayoutManager m)	defines the layout manager for the component
public void setVisible(boolean status)	changes the visibility of the component, by default false

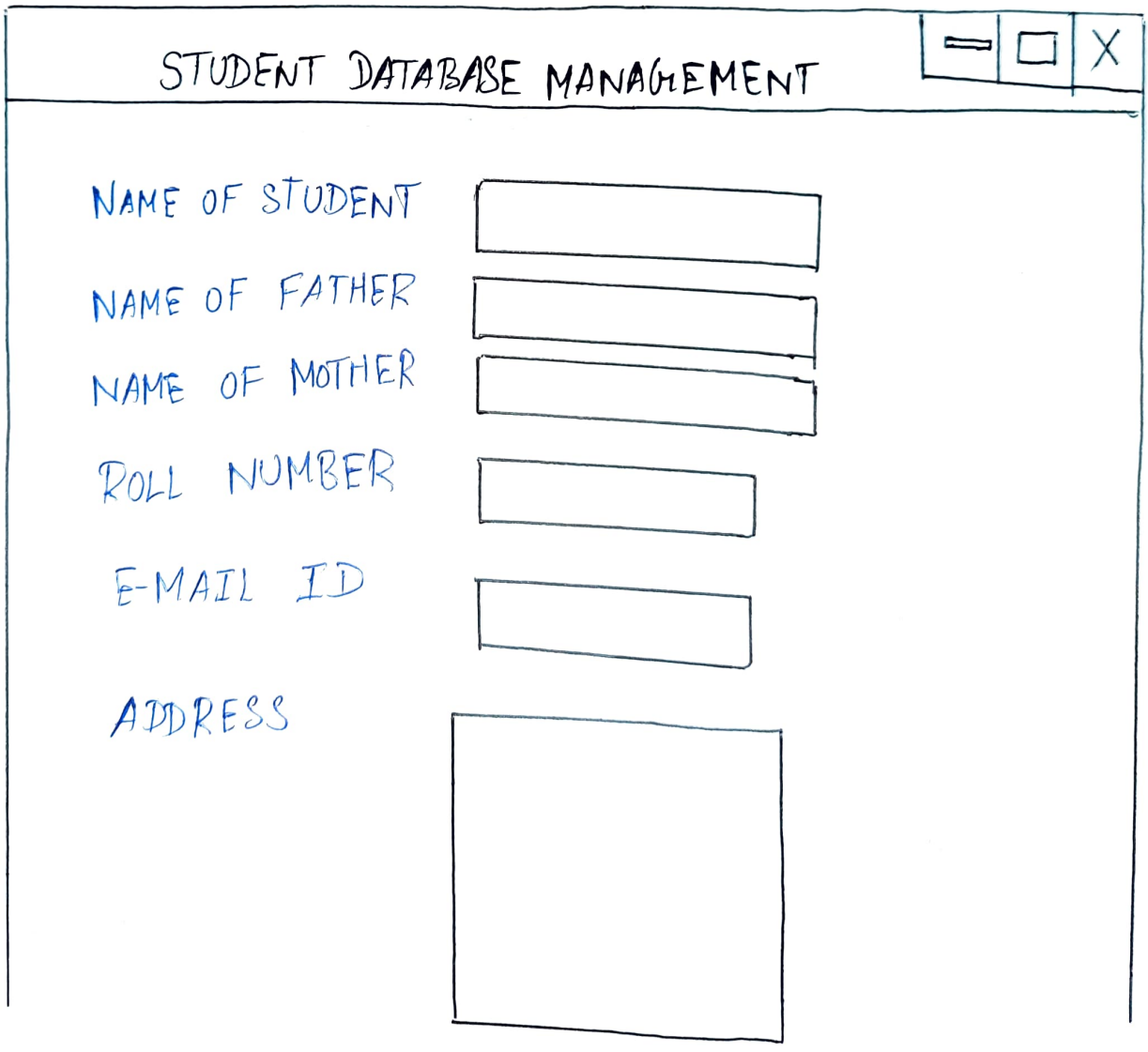
Component class:

* The component class is found under Java.awt package
* The container class is the subclass of component class
* All non-menu related elements that compose a graphical user interface are derived from abstract class component
* The component class defines a number of methods for handling events, changing window bounds, controlling fonts and colors, and drawing components and their fonts

Containers:

* A container is a component that can accomodate other components and also other containers
Containers provide the support for building complex hierarchial graphical user interface
* Container provides the overloaded method add() to include components in the container

4) GUI - student data base:



GENDER O MALE O FEMALE

% OF MARKS IN 10TH

% OF MARKS IN 12TH

DATE OF BIRTH DD | MM | YYYY

DEPARTMENT CHOSEN CSE

SUBMIT