


# National University of Computer and Emerging Sciences, Lahore Campus

	<b>Course Name:</b>	<b>Software Construction &amp; Development</b>	<b>Course Code:</b>	<b>CS-3001</b>
	<b>Degree Program:</b>	<b>BS(SE)</b>	<b>Semester:</b>	<b>Fall 2023</b>
	<b>Exam Duration:</b>	<b>180 minutes (3 hours)</b>	<b>Total Marks:</b>	<b>80</b>
	<b>Paper Date:</b>	<b>29 - Dec - 2023</b>	<b>Weight:</b>	<b>40.00%</b>
	<b>Section:</b>	<b>ALL</b>	<b>Pages:</b>	<b>8</b>
	<b>Exam Type:</b>	<b>Final</b>	<b>Questions:</b>	<b>5</b>

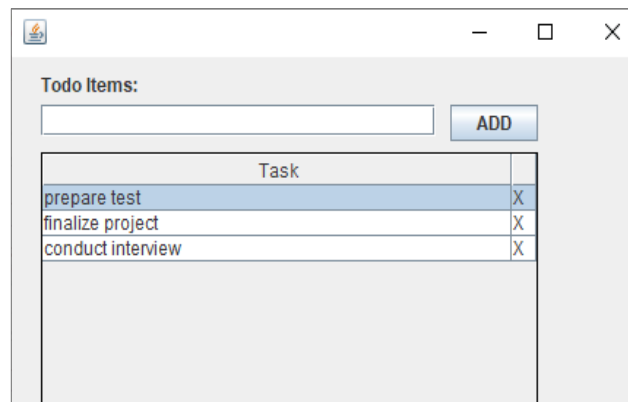
**Student Name:**\_\_\_\_\_ **Roll No.**\_\_\_\_\_ **Section:**\_\_\_\_\_

**Instruction/Notes:** Attempt all questions. Do not use pencil or red ink. In case of confusion or ambiguity make a reasonable assumption. **Do not attach any extra sheet.** Use extra sheet for rough work only. A **double-sided hand-written** cheat sheet is allowed but it shouldn't be photo-copied.

## Question 1 [CLO-1]

20 points

Consider a simple Java Swing Application that tracks a list of To-do items using a JTable component, as illustrated below:



Partial code is given below. Complete the blank portions to ensure that Todo items can be added and removed.

```
public class TodosUI extends javax.swing.JFrame{
    private javax.swing.JButton addButton;
    private javax.swing.JLabel label;
    private javax.swing.JTextField todoTextField;
    private javax.swing.JPanel todosTablePanel;

    private TodosTableModel model;
    private JTable todosTable;
    private JScrollPane scroll;

    public TodosUI(ArrayList<Todo> d) {
        initComponents(); // add components and set layout
        // relevant event handlers creation to be done here ...
    }
}
```

```

private class TodosTableModel extends AbstractTableModel{
    private ArrayList<Todo> todos;
    private final String[] columnNames = {"Task", " "};

    public TodosTableModel(ArrayList<Todo> t){
        todos = t;
        if (todos == null){
            todos = new ArrayList<>();
        }
    }

    @Override
    public int getRowCount() {
        return todos.size();
    }

    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

    @Override
    public Object getValueAt(int r, int c) {
        Todo todo = todos.get(r);
        if (c == 0){
            return todo.getTask();
        }

        return "X";
    }

    public void add(Todo t){
        // write code to add a todo item

    }

    public void delete(int index){
        // write code to delete a todo item

    }
}

private class RowSelectionListener implements ListSelectionListener{

    @Override
    public void valueChanged(ListSelectionEvent evt) {
        // write code to handle row manipulation in the table

    }
}
}

```

Consider a Java based client-server application for monitoring patient oxygen levels in a hospital ICU. Oxygen saturation is measured in percentage and a reading below 80% is problematic in which case nursing staff needs to be notified. The application is implemented using socket programming where each client measures oxygen saturation level every second for the relevant patient and transmits this information (bed number and saturation level) to the server. The task of server is to monitor all attached clients (beds) and in case the level of any patient goes below threshold (80%) then display flashing red alarm and bed number on the attached screen (assuming it is a shared resource where only a single bed information can be shown at a time).

The client code is given as follows:

```
public class OximeterClient {

    public static void main(String[] args) {
        Socket socket;

        try{

            socket = new Socket("localhost",4444);

            PrintWriter socketwriter = new PrintWriter(socket.getOutputStream());

            Thread thread = new Thread(new Runnable(){
                public void run(){
                    try{
                        while(true){
                            int value = measureO2level();
                            socketwriter.println("" + value);
                            socketwriter.flush();
                            Thread.sleep(1000);
                        }
                    } catch(Exception ex){
                        System.out.println("Exception:" + ex.getMessage());
                    }
                }
            });
            thread.start();
            thread.join();
            socket.close();

        }
        catch(Exception ex){
            System.out.println("Exception:" + ex.getMessage());
        }

    }

    private static int measureO2level(){
        return ((int)( Math.random() * 30)) + 69;
    }

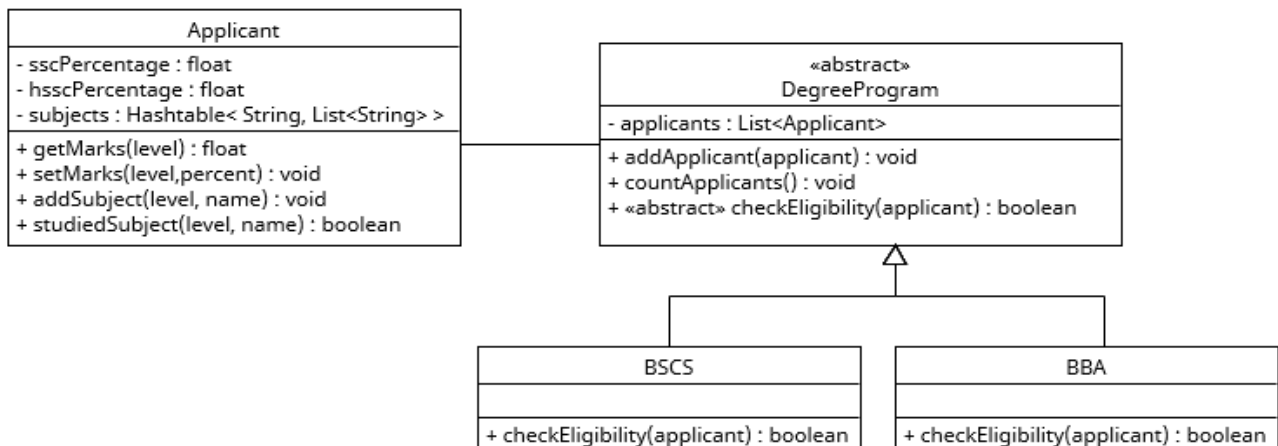
}
```

Write corresponding server code that can handle multiple clients (beds) simultaneously and raise alarm if the level goes below the threshold at any bed. Take care of necessary threading and concurrency issues.





Consider the following class diagram of an admission management system:



Details of functions are described as under:

- **setMarks** and **getMarks** set and retrieve marks in percentage respectively for the given level (SSC i.e. matric or equivalent, and HSSC i.e. intermediate or equivalent)
- **addSubject** adds a subject studied at given level e.g. Mathematics at HSSC
- **studiedSubject** checks whether the subject was studied at the given level or not
- **addApplicant** adds an applicant to the list of applicants in a degree program if the applicant is eligible (through **checkEligibility**) and **countApplicants** determine the number of eligible applicants added to the list
- **checkEligibility** for BSCS returns true if applicant has: i) 60% or above marks at SSC level, ii) 50% marks or above at HSSC level, iii) studied Mathematics at HSSC level
- **checkEligibility** for BBA returns true if applicant has: i) 60% or above marks at SSC level, ii) 50% marks or above at HSSC level

Write unit tests (using JUnit) for the following functions:

(a) Applicant.studiedSubject(level, name)

(b) DegreeProgram.addApplicant(applicant)

(c) BSCS.checkEligibility(applicant)

(d) BBA.checkEligibility(applicant)

**Question 4 [CLO-4]**

**5+5=10 points**

Answer the following questions:

(a) What is the difference between a Centralized and a Distributed Version Control System?

(b) A developer using Git committed changes to the repository but forgot to push them. What can possibly go wrong in such a situation and what is the purpose of push command?

**Question 5 [CLO-5]**

**5+5=10 points**


A development team has completed work on a new educational application (built in JAVA) for secondary school students and intends to release the corresponding JAR file on Microsoft Store:

(a) Advise them on the necessary steps for a successful release:

(b) What issue will occur if the JAR is not signed?



# National University of Computer and Emerging Sciences, Lahore Campus

	<b>Course Name:</b>	<b>Software Construction &amp; Development</b>	<b>Course Code:</b>	<b>CS-3001</b>
	<b>Degree Program:</b>	<b>BS(SE)</b>	<b>Semester:</b>	<b>Fall 2023</b>
	<b>Exam Duration:</b>	<b>180 minutes (3 hours)</b>	<b>Total Marks:</b>	<b>80</b>
	<b>Paper Date:</b>	<b>29 - Dec - 2023</b>	<b>Weight:</b>	<b>40.00%</b>
	<b>Section:</b>	<b>ALL</b>	<b>Pages:</b>	<b>8</b>
	<b>Exam Type:</b>	<b>Final</b>	<b>Questions:</b>	<b>5</b>

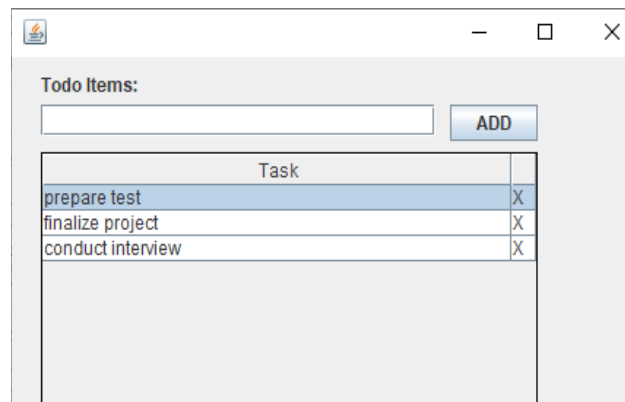
**Student Name:**\_\_\_\_\_ **Roll No.**\_\_\_\_\_ **Section:**\_\_\_\_\_

**Instruction/Notes:** Attempt all questions. Do not use pencil or red ink. In case of confusion or ambiguity make a reasonable assumption. **Do not attach any extra sheet.** Use extra sheet for rough work only. A **double-sided hand-written** cheat sheet is allowed but it shouldn't be photo-copied.

## Question 1 [CLO-1]

20 points

Consider a simple Java Swing Application that tracks a list of To-do items using a JTable component, as illustrated below:



Partial code is given below. Complete the blank portions to ensure that Todo items can be added and removed.

```
public class TodosUI extends javax.swing.JFrame{
    private javax.swing.JButton addButton;
    private javax.swing.JLabel label;
    private javax.swing.JTextField todoTextField;
    private javax.swing.JPanel todosTablePanel;

    private TodosTableModel model;
    private JTable todosTable;
    private JScrollPane scroll;

    public TodosUI(ArrayList<Todo> d) {
        initComponents(); // add components and set layout
        // relevant event handlers creation to be done here ...
        todosTable.getSelectionModel().addListSelectionListener(
            new RowSelectionListener());
        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                model.add(new Todo(todoTextField.getText()));
                todoTextField.setText("");
            }
        });
    }
}
```

```

private class TodosTableModel extends AbstractTableModel{
    private ArrayList<Todo> todos;
    private final String[] columnNames = {"Task", " "};

    public TodosTableModel(ArrayList<Todo> t){
        todos = t;
        if (todos == null){
            todos = new ArrayList<>();
        }
    }

    @Override
    public int getRowCount() {
        return todos.size();
    }

    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

    @Override
    public Object getValueAt(int r, int c) {
        Todo todo = todos.get(r);
        if (c == 0){
            return todo.getTask();
        }

        return "X";
    }

    public void add(Todo t){
        // write code to add a todo item
        todos.add(t);
        fireTableDataChanged();
    }

    public void delete(int index){
        // write code to delete a todo item
        if (index >= 0 && index < todos.size()){
            todos.remove(index);
        }
        fireTableDataChanged();
    }
}

private class RowSelectionListener implements ListSelectionListener{

    @Override
    public void valueChanged(ListSelectionEvent evt) {
        // write code to handle row manipulation in the table
        if (evt.getSource() == todosTable.getSelectionModel()) {
            if (todosTable.getSelectedColumn() > 0){
                model.delete(todosTable.getSelectedRow());
            }
        }
    }
}
}

```

Consider a Java based client-server application for monitoring patient oxygen levels in a hospital ICU. Oxygen saturation is measured in percentage and a reading below 80% is problematic in which case nursing staff needs to be notified. The application is implemented using socket programming where each client measures oxygen saturation level every second for the relevant patient and transmits this information (bed number and saturation level) to the server. The task of server is to monitor all attached clients (beds) and in case the level of any patient goes below threshold (80%) then display flashing red alarm and bed number on the attached screen (assuming it is a shared resource where only a single bed information can be shown at a time).

The client code is given as follows:

```
public class OximeterClient {

    public static void main(String[] args) {
        Socket socket;

        try{

            socket = new Socket("localhost",4444);

            PrintWriter socketwriter = new PrintWriter(socket.getOutputStream());

            Thread thread = new Thread(new Runnable(){
                public void run(){
                    try{
                        while(true){
                            int value = measureO2level();
                            socketwriter.println("" + value);
                            socketwriter.flush();
                            Thread.sleep(1000);
                        }
                    } catch(Exception ex){
                        System.out.println("Exception:" + ex.getMessage());
                    }
                }
            });
            thread.start();
            thread.join();
            socket.close();

        }
        catch(Exception ex){
            System.out.println("Exception:" + ex.getMessage());
        }

    }

    private static int measureO2level(){
        return ((int)( Math.random() * 30)) + 69;
    }

}
```

Write corresponding server code that can handle multiple clients (beds) simultaneously and raise alarm if the level goes below the threshold at any bed. Take care of necessary threading and concurrency issues.

```

public class IcuServer {

    public static void main(String[] args) {
        ServerSocket server;
        Display display;

        try {
            server = new ServerSocket(4444);
            display = new Display();

            Socket client = null;
            do {
                client = server.accept();

                if (client != null){
                    OximeterTask task = new OximeterTask(client,display);
                    Thread thread = new Thread(task);
                    thread.start();
                }

            } while (client != null);

        } catch (Exception ex) {
            log(ex.toString());
        }
    }

    private static void log(String message){
        System.out.println(message);
    }
}

public class Display{

    // other attributes and functions

    public synchronized void raiseAlarm(String bedNumber){
        System.out.println("Alarm at " + bedNumber);
        // flash alarm signal
    }
}

```

```

public class OximeterTask implements Runnable {

    Socket client;
    Display display

    public OximeterTask(Socket client, Display display) {
        this.client = client;
        this.display = display;
    }

    public void run() {

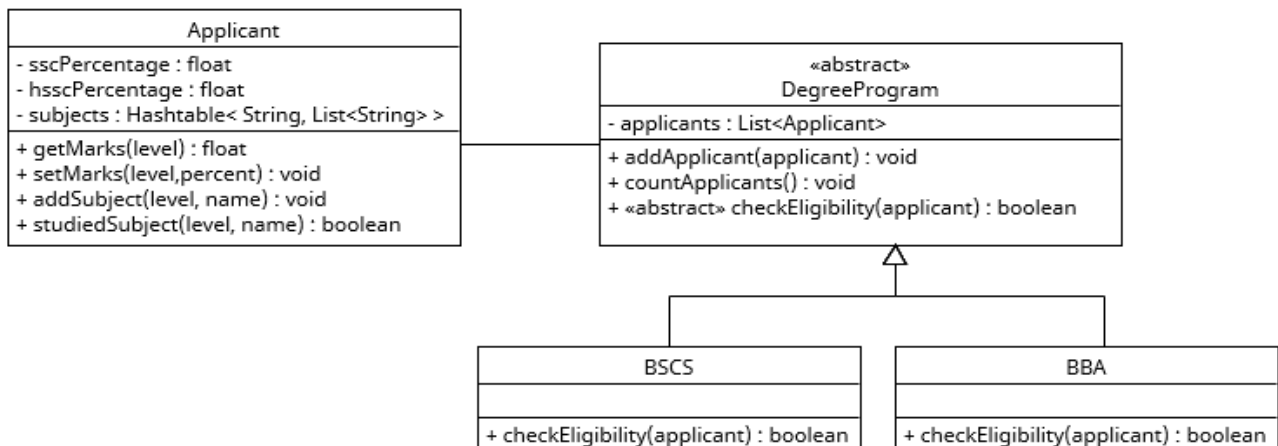
        try {
            PrintWriter writer = new PrintWriter(client.getOutputStream());
            BufferedReader reader = new BufferedReader(new
                InputStreamReader(client.getInputStream()));

            String message = "";
            while((message = reader.readLine()) != null){
                System.out.println("bed " + client.getPort() + ":" + message);
                Integer value = Integer.parseInt(message);
                if (value < 80){
                    display.raiseAlarm("" + client.getPort());
                }
            }

        } catch(Exception ex){
            ex.printStackTrace();
        }
    }
}

```

Consider the following class diagram of an admission management system:



Details of functions are described as under:

- **setMarks** and **getMarks** set and retrieve marks in percentage respectively for the given level (SSC i.e. matric or equivalent, and HSSC i.e. intermediate or equivalent)
- **addSubject** adds a subject studied at given level e.g. Mathematics at HSSC
- **studiedSubject** checks whether the subject was studied at the given level or not
- **addApplicant** adds an applicant to the list of applicants in a degree program if the applicant is eligible (through **checkEligibility**) and **countApplicants** determine the number of eligible applicants added to the list
- **checkEligibility** for BSCS returns true if applicant has: i) 60% or above marks at SSC level, ii) 50% marks or above at HSSC level, iii) studied Mathematics at HSSC level
- **checkEligibility** for BBA returns true if applicant has: i) 60% or above marks at SSC level, ii) 50% marks or above at HSSC level

Write unit tests (using JUnit) for the following functions:

(a) Applicant.studiedSubject(level, name)

```

Applicant app = new Applicant();
assertFalse(app.studiedSubject("HSSC", "Mathematics"));
app.addSubject("HSSC", "Mathematics");
assertTrue(app.studiedSubject("HSSC", "Mathematics"));
assertFalse(app.studiedSubject("SSC", "English"));
app.addSubject("SSC", "English");
assertTrue(app.studiedSubject("SSC", "English"));
  
```

(b) DegreeProgram.addApplicant(applicant)

```
DegreeProgram prog = new BBA(); // or any other concrete subclass e.g. BSCS
assertEquals(prog.countApplicants(), 0);

Applicant app = new Applicant();
app.setMarks("SSC", 65);
app.setMarks("HSSC", 65);
prog.addApplicant(app);

assertEquals(prog.countApplicants(), 1);
```

(c) BSCS.checkEligibility(applicant)

```
DegreeProgram prog = new BSCS();
Applicant app = new Applicant();
assertFalse(prog.checkEligibility(app));

app.setMarks("SSC", 65);
assertFalse(prog.checkEligibility(app));

app.setMarks("HSSC", 65);
assertFalse(prog.checkEligibility(app));
```

(d) BBA.checkEligibility(applicant)

```
DegreeProgram prog = new BBA();
Applicant app = new Applicant();
assertFalse(prog.checkEligibility(app));

app.setMarks("SSC", 65);
assertFalse(prog.checkEligibility(app));

app.addSubject("HSSC", "Mathematics");
assertFalse(prog.checkEligibility(app));

app.setMarks("HSSC", 65);
assertTrue(prog.checkEligibility(app));
```

#### Question 4 [CLO-4]

5+5=10 points

Answer the following questions:

(a) What is the difference between a Centralized and a Distributed Version Control System?

In a centralized system, repository is maintained centrally – clients only have their working copies.

In a distributed system, repository is distributed / replicated among all clients i.e. each client has a working copy as well as the snapshot of repository.

(b) A developer using Git committed changes to the repository but forgot to push them. What can possibly go wrong in such a situation and what is the purpose of push command?

If the changes are committed but not pushed, then they are tracked locally on the client machine only. If the client machine is corrupted, information loss may occur. The purpose of push command is to synchronize the changes with a remote repository as well.

**Question 5 [CLO-5]**

**5+5=10 points**

A development team has completed work on a new educational application (built in JAVA) for secondary school students and intends to release the corresponding JAR file on Microsoft Store:

(a) Advise them on the necessary steps for a successful release:


Develop and test the release, run QA procedures, prepare all the artifacts (executables, source-code, relevant docs such as user manuals and API docs, etc.) , apply version control labels

(b) What issue will occur if the JAR is not signed?

If JAR is not signed then its integrity (correct JAR prepared by trusted vendor) cannot be verified and Microsoft Store will not accept it for publishing.



## National University of Computer and Emerging Sciences, Lahore Campus

	<b>Course Name:</b>	Software Construction & Development	<b>Section:</b>	ALL
	<b>Program:</b>	BS (Software Engineering)	<b>Semester:</b>	Fall 2022
	<b>Duration/Date:</b>	3 Hours, 22-Dec-2022	<b>Total Marks:</b>	80
	<b>Evaluation Type:</b>	Final Exam	<b>Weight:</b>	40%
	<b>Course Code:</b>	SE3001	<b>Page(s)</b>	16
	<b>Name:</b>		<b>Roll Number:</b>	

### Important Note:

- The quality of the code will affect the marks.
- Students will receive **ZERO** marks if the answers are plagiarized.
- Use of **mobile phones, internet, and, ANY type of smart devices** during the exam is strictly prohibited.
- Discussion with other students is not allowed.
- Exchange of notes and stationery with other students are not allowed.
- If any of the above rules is violated by the student. The invigilator has the right to file DC case against that student and the invigilator also has the right to take your exam away and ask you to leave the exam hall.

Question No	1	2	3	4	5	Total
Maximum Marks	10	10	20	30	10	80
Marks Obtained						

### Question 1: Short answer (CLO1, 10 points)

#### Part-A (3 points)

Consider the following code.

```
class A {
private int x;
public A(int x) {
    this.x= x;
}

public void m() {
    System.out.println(x-1);
}
}
```

```

class B extends A{
    private int y;

    public B(int y) {
        super(y-1);
        this.y= y;
    }

    public void m() {
        System.out.println(y+1);
    }
}

```

(i) What is printed to the console by **(new A(3)).m()** ? (1 point)

(ii) What is printed to the console by **(new B(3)).m()** ? (1 point)

(iii) 1 point What is printed to the console by **((A)(new B(3))).m()** ? (1 point)

### **Part-B** (3 points)

Suppose that the integer array `list` has been declared and initialized as follows:

```
private int[] list = { 10, 20, 30, 40, 50 };
```

This statement sets up an array of five elements with the initial values shown below:

list				
10	20	30	40	50

Given this array, what is the effect of calling the method

```
mystery(list);
```

if `mystery` is defined as:

```
public void mystery(int[] array) {  
    int tmp = array[array.length - 1];  
    for (int i = 1; i < array.length; i++) {  
        array[i] = array[i - 1];  
    }  
    array[0] = tmp;  
}
```

Work through the method carefully and indicate your answer by filling in the boxes below to show the final contents of `list`:

Answer to Part-B:

list				

**Part-C** (4 points)

You are given the following code:

```
public static void m(int x){  
    try{  
        m2(x);  
        System.out.println(1);  
    } catch(ArithmeticException e) {  
        System.out.println(2);  
    } catch(Exception e) {  
        System.out.println(3);  
    }  
}
```

```

public static void m2(int x) throws IOException {
    System.out.println(4);
    if (x==1)
        throw new IOException();
    if (x==0)
        throw new ArithmeticException();
    System.out.println(5);
}

```

(i) Write what is printed to the console by m(1). (2 points)

(ii) Write what is printed to the console by m(0). (2 points)

## Question 2: (CLO2, 10 points)

### Part-A (5 points)

Class **SearchEngine**, below, provides a search bar in a GUI. The GUI is displayed properly, but clicking the search button does nothing. Your task: make changes to SearchEngine so that it will listen for a click of the search button and call method **search** with the appropriate text if that event occurs.

Information to recall:

- **TextField** has a method **String getText()**
- **Button** has a method **void addActionListener(ActionListener)**
- **Button** notifies its action listeners whenever it is clicked and only when it is clicked
- Interface **ActionListener** has a single method **void actionPerformed(ActionEvent)**

**Note:** You need to fill the boxes only

```

public class SearchEngine extends JFrame
{
    private JTextField searchBar= new JTextField("Enter your search here");
    private JButton submit= new JButton("Search");

    public SearchEngine() {
        Container cp= getContentPane();

        setSize(300, 100);
        setResizable(false);

        cp.add(searchBar, BorderLayout.CENTER);
        cp.add(submit, BorderLayout.WEST);
        setVisible(true);
        pack();
    }

    private void search(String input) { ... }
}

```

### Part-B (5 points)

In continuation of above scenario regarding SearchEngine class, your task is now to enhance the functionality of search method given in the Part-A in order to save the searches entered in the searchBar textfield into the DB on each time search button is clicked. Assume that DB connection is already established with MySQL database that is “mydb” having only one table namely, ‘**search\_keywords**’ that has following structure:

Column Name	Type	Constraints
SearchID	Number (auto increment)	Primary Key
Search_Keyword	Text	Not Null

DB Connection object is already provided into the **search** method. Write the missing code of the given method so that searches can be saved into the search\_keywords database table.

```
private void search(String input){
```

```
.....
```

```
try{
```

```
    Class.forName("com.mysql.jdbc.Driver");
```

```
    Connection con=null;
```

```
    con = DriverManager.getConnection("jdbc:mysql://localhost: 3306/mydb", "root", "123");
```

```
//write the missing code here
```

```
        con.close();
```

```
    }catch (Exception sqlEx) {
```

```
        System.out.println(sqlEx);
```

```
    }
```

```
}
```

**Question 3:** (CLO1, 20 points)

Write a class, call it **GradesCount**, to read a list of grades from the file called input.txt stored on a Hard Disk (integer numbers in the range 0 to 100). User enter the grades manually between 0 to 100 into a file namely, input.txt line by line and save it each time. Store each grade in a A, B, C, D or F ArrayList as follows: 90 to 100 = A, 80 to 89 = B, 70 to 79 = C, 60 to 69 = D, and 0 to 59 = F.

Finally, your program should write an output.txt file on the disk that contains the total number of grades entered, the number of A, B, C, D and F, and a list of the A's .

For example, if the input is...

input.txt

38  
86  
92  
55  
83  
42  
90

then the output should be:

output.txt

Total number of grades = 7  
Number of A = 2  
Number of B = 2  
Number of C = 0  
Number of D = 0  
Number of F = 3  
The A grades are: 92, 90



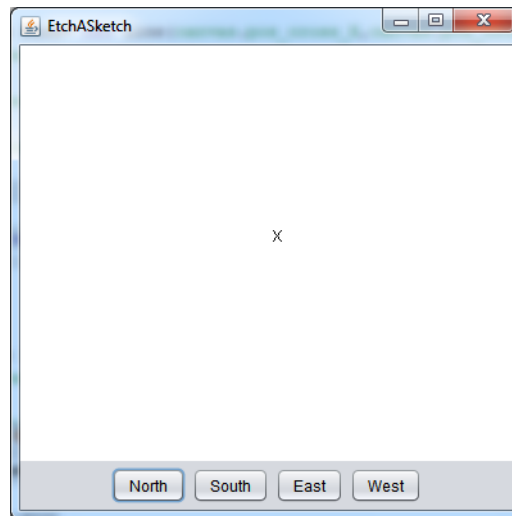




#### Question 4: Graphics and Interactivity (CLO2, 30 points)

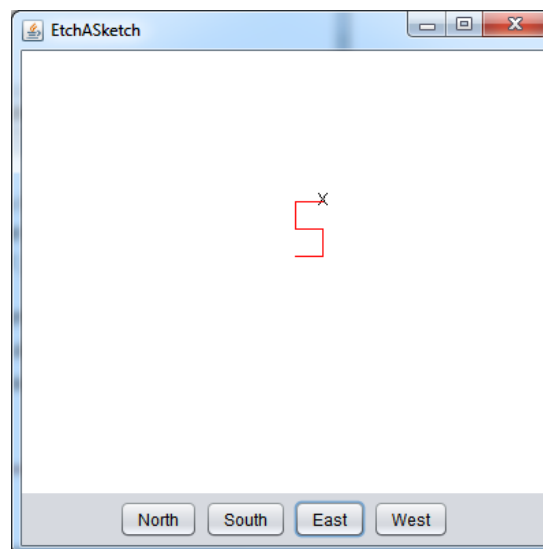
Write a `GraphicsProgram` that does the following:

1. Add buttons to the South region labeled "North", "South", "East", and "West".
2. Create an x-shaped cross 10 pixels wide and 10 pixels high (*you may draw 'X' alphabet instead by using `drawString` method*).
3. Adds the cross so that its center is at the center of the graphics canvas. Once you have completed these steps, the display should look like this:



4. Implement the actions for the button so that clicking on any of these buttons moves the cross 20 pixels in the specified direction. At the same time, your code should add a red `Line` that connects the old and new locations of the pen (*hint: you may use `drawLine(x1,y1,x2,y2)` method to draw a Line*).

Keep in mind that each button click adds a new `Line` that starts where the previous one left off. The result is therefore a line that charts the path of the cross as it moves in response to the buttons. For example, if you clicked `East`, `North`, `West`, `North`, and `East` in that order, the screen would show a "S" like this (note the "S" would be red):









**Question 5: (CLO3, 10 points)**

You need to design the two distinct test cases only to cover the boundary of **addMinute** method for the **Time** class that is given below and also write the implementation code of aforementioned designed test cases by using **JUnit** for the Time class to test addMinute method.

**Time.java**

```
public class Time {
    private int hours, minutes;

    public Time (int h, int m) {
        hours = h; minutes = m;
    }

    public String toString() {
        return Integer.toString(hours)+":"+Integer.toString(minutes);
    }

    void addMinute() {
        if (minutes==59) {
            minutes = 0; hours++;
        } else
            minutes++;
        }
    }
```







# Software Construction & Development (SE3001)

## Final Exam

Date: December 31, 2024

Course Instructor(s)

Dr. Farooq Ahmed, Mr. Waqas Ali

Total Time:

3 hour

Total Marks:

80

Total Questions:

5

22L-7971

Roll No

BSE-518

Section

Student Signature

Do not write below this line

Attempt all questions on the answer sheet

**CLO 1: Apply software engineering concepts to construct (i.e. design, develop, and test) software in team setting**

### Question 1

[20 marks]

A retail company wants to create a desktop application to manage sales on items which are present in database of named "company\_db" under the table "items\_table".

Retail Sales Management

Search Item:

ID	Name	Price	Expiry Date	Stock
----	------	-------	-------------	-------

ID	Name	Price	Quantity
----	------	-------	----------

Total Price: \$0.0

In above figure, left table is itemsTable and right one is cartTable. The application should allow the user to search items and add them to a cartTable, calculate the total price, and remove selected items. Design of GUI for this application is done in initComponents() method using Java Swing and there is NO need to implement that.

Your task is to implement the necessary functionality for:

- searching items
- adding items
- calculating the total cart price dynamically
- removing selected items.

```
public class Main extends JFrame {
    private JTextField searchField;
    private JButton searchButton, addButton, removeButton;
    private JTable itemsTable, cartTable;
    private JLabel totalLabel;
    private DefaultTableModel itemsTableModel, cartTableModel;
    private double totalPrice = 0.0;

    public Main() {
        initComponents(); // no need to implement this method
        // Event Handlers
        searchButton.addActionListener(new SearchAction());
        addButton.addActionListener(new AddToCartAction());
        removeButton.addActionListener(new RemoveFromCartAction());
    }

    // Implement the missing code
} // end of Main class
```

## CLO 2: Implement software design patterns as part of software construction activity

### Question 2

[20 marks]

A smart home system uses a client-server architecture (implemented using socket programming) to control lights in different rooms. Each client represents a light bulb that sends its status (on/off) to the server every second. The server monitors all bulbs and sends a notification to a central dashboard if any light stays on for more than a specified duration (e.g., 5 minutes).

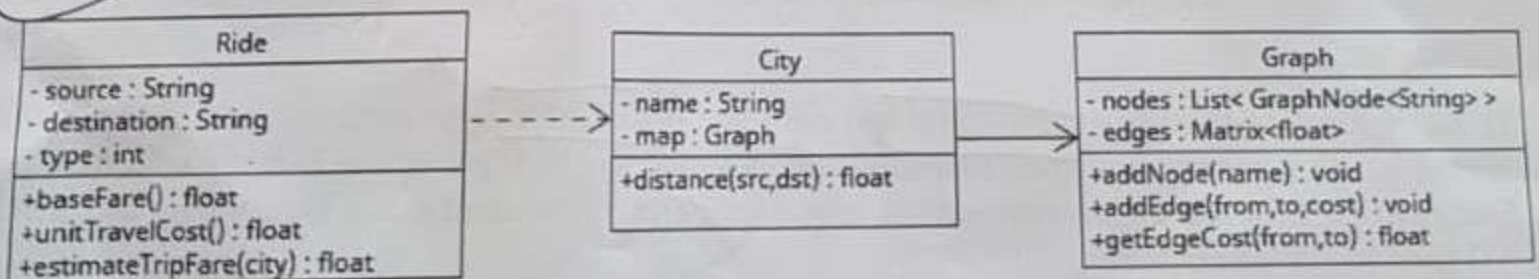
Implement a server that can handle multiple clients representing light bulbs. The server should:

1. Monitor each client's status in real-time using threading.
2. Maintain a record of how long each light bulb remains on.
3. Notify the dashboard when any bulb exceeds the specified on-time limit.

## CLO 3: Design Test Cases for a Software System

### Question 3

[20 marks]





Consider a simplified class diagram for a ride hailing service. Rides can be available for traveling from source to destination within the city. Each ride has a type (e.g. 1 for car, 2 for bike, etc.) that helps determine the base fare (e.g. Rs. 100 for car and Rs. 50 for bike, etc.) as well as travel cost per unit distance (e.g. Rs. 50 per km for car and Rs. 20 per km for bike, etc.). Total fare can be estimated as a sum of base fare and travel distance (from source to destination) multiplied by unit cost.

City class maintains a map (using a Graph) to determine the distance from a given source to destination. Graph is implemented as an adjacency matrix to determine cost of a direct edge from one node to other. For simplicity, it can be assumed that ride hailing service is available within the city only and inter-city travel is not covered.

Write unit test code (using JUnit) for Ride and City classes while using the Graph class appropriately. There is **no need** to write tests for Graph class but it will be helpful in testing other (Ride and City) classes.

**CLO 4: Use a version control system as part of software construction activity**

**Question 4**

[5+5 marks]

Answer the following questions:

- (a) What is the purpose of using tags in Git version control system? Explain in comparison with commit messages and revision numbers.
- (b) What problems can occur while merging one branch to the other and how to handle them?

**CLO 5: Implement the deployment related steps to bring the constructed software in use**

**Question 5**

[5+5 marks]

Answer the following questions:

- (a) Why are digital signatures important during the release distribution process?
- (b) What is the significance of generating API documentation? How can it be facilitated?