Object Oriented Programming Using Java

SE101.3

Coursework 2023/2024

Name	J M S V JAYAWEERA
Index	29008
Degree	BSc (Hons) Software Engineering

1. Question No 1

```
import java.util.Scanner;
/*
  This is a first question answer for the java assignment
  which is to write a simple text based application using java to display your name
   */

public class Question_1 {

   public static void main(String [] args){

      String forename,lastname ;
      Scanner input = new Scanner(System.in);
      System.out.println("Please Enter Your First Name: ");
      forename = input.next();

      System.out.println("Please Enter Your Last Name: ");
      lastname = input.next();


      System.out.println("The name is: " +forename+" "+lastname);


   }

}
```

2. Question No 2

```
import java.util.Scanner;

public class Question_2 {
   /*
   This is the question number 2 which is to calculate product of three user input
inteegers.
    */

   public static void main(String[] args)
   {
      int num1,num2,num3,product;
      Scanner input = new Scanner(System.in);
      System.out.println("Please Enter Numbers: ");
      num1 = input.nextInt();
      num2 = input.nextInt();
      num3 = input.nextInt();
```

```
        product=num1*num2*num3;

        System.out.println("The Product pf the numbers is: "+product);



    }


}
```

3. Question Number 3

```
import java.util.Scanner;
/*
This is the third question which is to convert the
farenheit value into a celcius value
 */

public class Question_3 {

   public static void main(String [] args)
   {
      float farenheit,celcius;
      Scanner input = new Scanner(System.in);
      System.out.println("Please Enter Temperature in Farenheit: ");
      farenheit = input.nextFloat();

      celcius=(5.0f/9.0f)*(farenheit-32);

      System.out.println("The Temperature in Celcius is: "+celcius);
   }
}
```

4. Question Number 4

```
import java.util.Scanner;

/*
This is the question Number four which is to Write an application that inputs three
integers from the user
and displays the sum, average, product, smallest and largest of the numbers
```

```
 */
public class Question_4 {
   public static void main(String[] args)
   {
      int num1, num2,num3;
      float sum=0,average=0,product,smallest,largest;

      Scanner input = new Scanner(System.in);
      System.out.println("Please Enter the Numbers: ");
      num1 = input.nextInt();
      num2 = input.nextInt();
      num3 = input.nextInt();

      sum = (float)num1+num2+num3;
      average=sum/3;
      product=(float)num1*num2*num3;

      if(num1>num2 && num1>num3)
      {
         largest=num1;
      }
      else if(num2>num3 && num2>num1)
      {
         largest=num2;
      }

      else
      {
         largest=num3;
      }

      if(num1<num2 && num1<num3)
      {
         smallest=num1;
      }

      else if(num2<num3 && num2<num1)
      {
         smallest=num2;
      }

      else
      {
         smallest=num3;
      }
```

```
        System.out.println("The Sum is: "+sum);
        System.out.println("The Aveerage is: "+average);
        System.out.println("The Product is: "+product);
        System.out.println("The Smallest is: "+smallest);
        System.out.println("The Highest is:  "+largest);


    }
}
```

5. Question Number 5

```
import java.util.Scanner;
import java.text.DecimalFormat;
/*
This is the Fifth question which is to Write a Java application that allows the user to
enter
up to 20 integer grades into an array. Stop the loop by typing in -1.
Your main method should call an Average method that returns the average of the
grades.
Use the DecimalFormat class to format the average to 2 decimal places
 */

public class Question_5 {

    public static void main(String[] args)
    {
        int[] grades = new int[20];
        int count = 0;
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter up to 20 integer grades (-1 to stop):");
        int input = scanner.nextInt();

        while (input != -1 && count < 20)
        {
            grades[count] = input;
            count++;
            input = scanner.nextInt();
        }

        double average = calculateAverage(grades, count);
        DecimalFormat decimalFormat = new DecimalFormat("#0.00");
        String formattedAverage = decimalFormat.format(average);

        System.out.println("Average grade: " + formattedAverage);
```

```
    }

    public static double calculateAverage(int[] grades, int count) {
        if (count == 0) {
            return 0.0;
        }

        int sum = 0;
        for (int i = 0; i < count; i++) {
            sum += grades[i];
        }

        return (double) sum / count;
    }
}
```

6. Question Number 6

```
/*
This is the question number 6 in the java assignment
the object code can be found as the DateTest class.
refer the DateTest Class to have a good idea regarding this code and use of
constructer, set&get methods used.
 */

public class Date {
    private int month,day,year;

    public  Date(int month, int day, int year)
    {
        this.month=month;
        this.day=day;
        this.year=year;

    }

    //Getter Method for Month
    public int getMonth()
    {
        return month;
    }

    // Setter Method for Month
    public void setMonth(int month)
    {
        this.month=month;
```

```
    }

    //Gettter Method for Day
    public int getDay()
    {
       return day;
    }

    //Setter Method for Day
    public void setDay(int day)
    {
       this.day=day;
    }


    //Getter Method for Year
    public int getYear()
    {
       return year;
    }

    //Setter Method for Year
    public void setYear(int year)
    {
       this.year=year;
    }

    public void displayDate()
    {
       System.out.print(" "+month);
       System.out.print("/"+day);
       System.out.print("/"+year);
    }









}



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

```
        Date date1  = new Date(8,28,2000);
        System.out.println("The Initial Date ");
        date1.displayDate();


        date1.setMonth(2);
        date1.setDay(12);
        date1.setYear(2003);

        System.out.println("\nThe Updated Date ");
        date1.displayDate();




    }
}
```

7. Question Number 7

```
/*
This is the 7th Question of the assignment
 */
public class Item {
    protected static int Location;
    protected static String Description;

    //constructor - method
    public Item (int Location, String Description)
    {
        this.Description =Description;
        this.Location = Location;
    }

    // Getter Method for Location

    public static int getLocation()
    {
        return Location;
    }

    //Setter Method for Location

    public static void setLocation(int Location)
    {
        Item.Location = Location;
```

```
    }


    //Getter Method for Description

    public static String getDescription()
    {
       return Description;
    }

    //Setter Method for Description

    public static void setDescription(String Description)
    {
       Item.Description = Description;
    }




}



/*
This is the Sub Class of Item - Question Number 7
 */
public class Monster extends Item {
    public Monster(int Location, String Description )
    {
       super(Location,Description);
    }


}


/*
This is the main Class that we  created to perform and check all the methods within
the monster and Item
class - Question No 7
 */

public class Item_Monster_Main {
    public static void main( String [] args)
    {
       // Creating an Object for Item Class
```

```
        Item obj1 = new Item(200,"Kandy");
        System.out.println("Location: " +Item.getLocation());
        System.out.println("Description: " +Item.getDescription());

        //Creating an Object for Monster Class
        Monster obj2 = new Monster(300, "Pasikuda");
        System.out.println("Location: " +Monster.getLocation());
        System.out.println("Description: " +Monster.getDescription());


        System.out.println("");
        //Using Setter Method and Getter Method to Update Item
        Item.setLocation(900);
        Item.setDescription("Canada");
        System.out.println("The Updated Location is: " +Item.getLocation());
        System.out.println("The Updated Description is: " +Item.getDescription());


        //Using Setter and Getter Method to Update Monster
        Monster.setLocation(1500);
        Monster.setDescription("California");
        System.out.println("The Updated Location is: " +Monster.getLocation());
        System.out.println("The Updated Description is: " +Monster.getDescription());


    }
}
```

8. Question No 8

```
/*
This is the Question Number 8 in the assignment regarding static variables
the term static refers The static variable gets memory only once in the class area at
the time of class loading.
the test code is named as TestClass_SavingsAccount ( Object Code)
 */

public class SavingsAccount {
    private double SavingBalance;
    private static double annualInterestRate;



    public SavingsAccount(double SavingBalance, double annualInterestRate)
    {
        this.SavingBalance=SavingBalance;
```

```
        this.annualInterestRate=annualInterestRate;
    }
    public void calculateMonthlyInterest()
    {
        double MonthlyInterest= (SavingBalance * annualInterestRate)/12;
        SavingBalance=SavingBalance+MonthlyInterest;

    }

    public static void modifyInterestRate(double newAnnualInterestRate)
    {
        annualInterestRate=newAnnualInterestRate;

    }

    public double getSavingBalance()
    {
        return SavingBalance;
    }

}


public class TestClass_SavingsAccount {

    public static void main(String [] args)
    {
        SavingsAccount saver1 = new SavingsAccount(2000, 4.00);
        SavingsAccount saver2 = new SavingsAccount(3000, 4.00);
        saver1.calculateMonthlyInterest();
        saver2.calculateMonthlyInterest();

        System.out.println("The Monthly Interest of Saver 1 is: "
+saver1.getSavingBalance());
        System.out.println("The Monthly Interest of Saver 2 is: "
+saver2.getSavingBalance());

        System.out.println("");

        //The next Month interest New Balance under the updated interest rate
        SavingsAccount.modifyInterestRate(0.05);

        saver1.calculateMonthlyInterest();
        saver2.calculateMonthlyInterest();

        System.out.println("The New Monthly Interest Rate for Saver 1 is: "
+saver1.getSavingBalance());
```

```
        System.out.println("The New Monthly Interest Rate for Saver 2 is: "
+saver2.getSavingBalance());


    }
}
```

9. Question No 9

```
/*
This is the 9th Question of the Assignment which has many child classes
child classes - Truck,Ford,Sedan
 */

public class Car {
   private int speed;
   double regularprice;
   private String color;

   public Car(int speed, double regularprice, String color)
   {
      this.speed=speed;
      this.regularprice=regularprice;
      this.color=color;
   }

   public double getSalepPrice()
   {
      return regularprice;
   }

}

public class Truck extends Car{
   private int weight;
   public Truck(int speed, double regularprice, String color, int weight)
   {
      super(speed,regularprice,color);
      this.weight=weight;
   }

   public double getSalePrice()
   {
      if(weight>20000)
      {
         return regularprice*0.9;    //10% discount for weight >2000
```

```
        }
        else
            return regularprice*0.8;  //20% discount for weight<=2000
    }

}


public class Ford extends Car {
    private int year,manufacturerDiscount;
    public Ford(int speed, double regularprice,String color, int year, int
manufacturerDiscount )
    {
        super(speed,regularprice,color);
        this.year=year;
        this.manufacturerDiscount=manufacturerDiscount;
    }

    public double getSalePrice()
    {
        return super.getSalepPrice()-manufacturerDiscount;
    }


}


public class Sedan extends Car{
    int length;

    public Sedan(int speed, double regularprice, String color, int length)
    {
        super(speed, regularprice, color);
        this.length=length;
    }

    double getSalePrice()
    {
        if(length>20)
        {
            return regularprice*0.95; // 5% discount for length > 20
        }
        else{
            return regularprice*0.9; // 10% discount for length <=20
        }
    }
```

```
}

public class MyOwnAutoShop {
   public static void main(String [] args)
   {
      //Creating an instance of Sedan Class
      Sedan sedan =  new Sedan(280,25000,"Black", 20);

      //Creating two instances of Ford Class
      Ford ford1 = new Ford(180,15000,"Blue",2022, 2500);
      Ford ford2 = new Ford(190,18000,"White",2023,1500);

      //Creating an instance of Car Class
      Car car1 = new Car(290,25000, "Gray");


      //Displaying the Sale prices of all the instances
      System.out.println("Sedan Sale Price: $" +sedan.getSalePrice());
      System.out.println("Ford1 Sale Price: $" +ford1.getSalePrice());
      System.out.println("Ford2 Sale Price: $" +ford2.getSalepPrice());
      System.out.println("Car Sale Price: $" +car1.getSalepPrice());
   }
}
```

10. Question No 10

```
/*
This is the Question 10 of the Assignemnet which is anout Abstract Classes and
Methods
the Object code is Main.java
 */
   // Base class Shape
  public abstract class Shape {
      abstract void draw();
      abstract void erase();
   }

   // Subclass Circle
   class Circle extends Shape {
      @Override
      void draw() {
         System.out.println("Drawing a circle");
      }

      @Override
```

```
    void erase() {
        System.out.println("Erasing a circle");
    }
}

// Subclass Triangle
class Triangle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a triangle");
    }

    @Override
    void erase() {
        System.out.println("Erasing a triangle");
    }
}

// Subclass Square
class Square extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a square");
    }

    @Override
    void erase() {
        System.out.println("Erasing a square");
    }
}


public class Main {
    public static void main(String[] args) {
        // Polymorphism in action
        Shape[] shapes = {new Circle(), new Triangle(), new Square()};

        for (Shape shape : shapes) {
            shape.draw();
            shape.erase();
            System.out.println();
        }
    }
}
```

Part 2

```
/*
This is the Question No 10 Part two - Example code for Abstract classes and methods
the object code is Animal_Main.java
 */
// Abstract class Animal
public abstract class Animal {
    // Abstract method without implementation
    abstract void makeSound();

    // Concrete method with implementation
    void sleep() {
        System.out.println("Zzzz...");
    }
}

// Subclass Dog inheriting from Animal
class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Woof Woof!");
    }
}

// Subclass Cat inheriting from Animal
class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("Meow Meow!");
    }
}


public class Animmal_Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        Cat cat = new Cat();

        dog.makeSound();
        dog.sleep();

        cat.makeSound();
        cat.sleep();
    }
}
```

Part 3

```
/*
This is the 3rd Part of the Question 10 in the Assignment Regarding Abstract classes and
methods
the object code is Project_Main.java
 */

// Abstract base class providing a common interface
public abstract class ProjectClass {
    // Common method that all subclasses should implement
    abstract void debug();

    // Other common methods can be included here if needed
}

// Subclass 1 inheriting from the abstract base class
class Subclass1 extends ProjectClass {
    @Override
    void debug() {
        System.out.println("Debugging Subclass1...");
        // Implement debug-specific functionality here
    }
}

// Subclass 2 inheriting from the abstract base class
class Subclass2 extends ProjectClass {
    @Override
    void debug() {
        System.out.println("Debugging Subclass2...");
        // Implement debug-specific functionality here
    }
}

// Subclass 3 inheriting from the abstract base class
class Subclass3 extends ProjectClass {
    @Override
    void debug() {
        System.out.println("Debugging Subclass3...");
        // Implement debug-specific functionality here
    }
}


public class Project_Main {
    public static void main(String[] args) {
        // Create instances of the subclasses and call their debug() methods
```

```
        ProjectClass obj1 = new Subclass1();
        ProjectClass obj2 = new Subclass2();
        ProjectClass obj3 = new Subclass3();

        obj1.debug();
        obj2.debug();
        obj3.debug();
    }
}
```

11. Question 11

```
// Interface A
interface A {
    void meth1();
    void meth2();
}

// Implementing interface A in MyClass
class MyClass implements A {
    public void meth1() {
        System.out.println("Method 1 implementation in MyClass");
    }

    public void meth2() {
        System.out.println("Method 2 implementation in MyClass");
    }
}

public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        obj.meth1();
        obj.meth2();
    }
}



// Interface One
interface One {
    void methodOne();
}
```

```
// Interface Two
interface Two {
   void methodTwo();
}

// Class implementing both interfaces
class MultipleInheritance implements One, Two {
   public void methodOne() {
      System.out.println("Method One implementation");
   }

   public void methodTwo() {
      System.out.println("Method Two implementation");
   }
}

public class Main {
   public static void main(String[] args) {
      MultipleInheritance obj = new MultipleInheritance();
      obj.methodOne();
      obj.methodTwo();
   }
}
```

```
// Interface Test
interface Test {
   int square(int num);
}

// Implementing interface Test in Arithmetic class
class Arithmetic implements Test {
   public int square(int num) {
      return num * num;
   }
}

public class Main {
   public static void main(String[] args) {
      Arithmetic obj = new Arithmetic();
      int result = obj.square(5);
      System.out.println("Square of 5: " + result);
   }
}
```

```
class ToTestInt {
   public static void main(String[] args) {
      Arithmetic obj = new Arithmetic();
      int result = obj.square(10);
      System.out.println("Square of 10: " + result);
   }
}
```

12. Question 12

- Program for the example of try and catch block to check whether the given array size is negative or not:

```
public class ArraySizeCheck {
   public static void main(String[] args) {
      try {
         int size = -5;
         int[] array = new int[size];
         System.out.println("Array created successfully.");
      } catch (NegativeArraySizeException e) {
         System.out.println("Error: Array size cannot be negative.");
      }
   }
}
```

- Program for the example of multiple catch statements occurring in a program:

```
public class MultipleCatchExample {
   public static void main(String[] args) {
      try {
         int[] arr = new int[5];
         arr[5] = 10 / 0;
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Array index out of bounds.");
      } catch (ArithmeticException e) {
         System.out.println("Cannot divide by zero.");
      } catch (Exception e) {
         System.out.println("Some other exception occurred.");
      }
   }
}
```

- Program to illustrate subclass exception precedence over base class:

```
class BaseException extends Exception {
   public BaseException(String message) {
      super(message);
   }
}

class SubException extends BaseException {
   public SubException(String message) {
      super(message);
   }
}

public class ExceptionPrecedence {
   public static void main(String[] args) {
      try {
         throw new SubException("Subclass exception occurred.");
      } catch (BaseException e) {
         System.out.println("Caught BaseException: " + e.getMessage());
      } catch (Exception e) {
         System.out.println("Caught Exception: " + e.getMessage());
      }
   }
}
```

- Program to illustrate usage of try/catch with finally clause:

```
public class FinallyExample {
   public static void main(String[] args) {
      try {
         int num = 10 / 0;
         System.out.println("This won't be executed.");
      } catch (ArithmeticException e) {
         System.out.println("Caught ArithmeticException: " + e.getMessage());
      } finally {
         System.out.println("Finally block executed.");
      }
   }
}
```

- Program for the creation of a user-defined exception:

```
class MyCustomException extends Exception {
   public MyCustomException(String message) {
      super(message);
```

```
    }
}

public class UserDefinedException {
    public static void main(String[] args) {
        try {
            throw new MyCustomException("This is a user-defined exception.");
        } catch (MyCustomException e) {
            System.out.println("Caught MyCustomException: " + e.getMessage());
        }
    }
}
```

13. Question No 13

```
class MyRunnable implements Runnable {
    @Override
    public void run() {
        try {
            System.out.println(Thread.currentThread().getName() + " is running.");
            Thread.sleep(500);
            System.out.println(Thread.currentThread().getName() + " is awake.");
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread().getName() + " is interrupted.");
        }
    }
}

public class RunnableThreadExample {
    public static void main(String[] args) {
        MyRunnable myRunnable = new MyRunnable();
        Thread thread1 = new Thread(myRunnable, "Thread-1");
        Thread thread2 = new Thread(myRunnable, "Thread-2");

        thread1.start();
        thread2.start();
    }
}
```

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

    @Override
    public void run() {
        try {
            System.out.println(getName() + " is running.");
            Thread.sleep(1000);
            System.out.println(getName() + " is awake.");
        } catch (InterruptedException e) {
            System.out.println(getName() + " is interrupted.");
        }
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        System.out.println("Main thread is running.");

        MyThread thread1 = new MyThread("Thread-1");
        MyThread thread2 = new MyThread("Thread-2");

        thread1.start();
        thread2.start();

        try {
            // Main thread waits for thread1 and thread2 to complete
            thread1.join();
            thread2.join();
        } catch (InterruptedException e) {
            System.out.println("Main thread is interrupted.");
        }

        System.out.println("Main thread is finished.");
    }
}
```

14. Question No 14

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TemperatureConverterGUI extends JFrame {
  private JTextField celsiusTextField, kelvinTextField;

  public TemperatureConverterGUI() {
    setTitle("Temperature Converter");
    setSize(300, 150);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new GridLayout(3, 2));

    JLabel celsiusLabel = new JLabel("Celsius:");
    celsiusTextField = new JTextField();
    JLabel kelvinLabel = new JLabel("Kelvin:");
    kelvinTextField = new JTextField();

    JButton toKelvinButton = new JButton("Convert to Kelvin");
    toKelvinButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        convertToKelvin();
      }
    });

    JButton toCelsiusButton = new JButton("Convert to Celsius");
    toCelsiusButton.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        convertToCelsius();
      }
    });

    add(celsiusLabel);
    add(celsiusTextField);
    add(toKelvinButton);
    add(kelvinLabel);
    add(kelvinTextField);
    add(toCelsiusButton);

    setVisible(true);
  }
```

```
    private void convertToKelvin() {
        try {
            double celsius = Double.parseDouble(celsiusTextField.getText());
            double kelvin = celsius + 273.15;
            kelvinTextField.setText(String.format("%.2f", kelvin));
        } catch (NumberFormatException ex) {
            kelvinTextField.setText("Invalid input");
        }
    }

    private void convertToCelsius() {
        try {
            double kelvin = Double.parseDouble(kelvinTextField.getText());
            double celsius = kelvin - 273.15;
            celsiusTextField.setText(String.format("%.2f", celsius));
        } catch (NumberFormatException ex) {
            celsiusTextField.setText("Invalid input");
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new TemperatureConverterGUI();
            }
        });
    }
}
```

15. Question No 15

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class BasicCalculator extends JFrame implements ActionListener {
    private JTextField inputField;
    private double num1, num2, result;
    private char operator;

    public BasicCalculator() {
        setTitle("Basic Calculator");
        setSize(300, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
// Initialize input field
inputField = new JTextField();
inputField.setEditable(false);

// Create buttons for digits and operators
JButton[] numberButtons = new JButton[10];
for (int i = 0; i < 10; i++) {
   numberButtons[i] = new JButton(String.valueOf(i));
   numberButtons[i].addActionListener(this);
}

JButton addButton = new JButton("+");
JButton subtractButton = new JButton("-");
JButton multiplyButton = new JButton("*");
JButton divideButton = new JButton("/");
JButton equalButton = new JButton("=");
JButton clearButton = new JButton("C");

// Add action listeners to operator buttons
addButton.addActionListener(this);
subtractButton.addActionListener(this);
multiplyButton.addActionListener(this);
divideButton.addActionListener(this);
equalButton.addActionListener(this);
clearButton.addActionListener(this);

// Create the layout using GridBagLayout
setLayout(new GridBagLayout());
GridBagConstraints constraints = new GridBagConstraints();
constraints.gridx = 0;
constraints.gridy = 0;
constraints.gridwidth = 4;
constraints.fill = GridBagConstraints.HORIZONTAL;
add(inputField, constraints);

constraints.gridwidth = 1;
constraints.gridy = 1;
add(numberButtons[7], constraints);
constraints.gridx = 1;
add(numberButtons[8], constraints);
constraints.gridx = 2;
add(numberButtons[9], constraints);
constraints.gridx = 3;
add(addButton, constraints);

constraints.gridy = 2;
```

```
        constraints.gridx = 0;
        add(numberButtons[4], constraints);
        constraints.gridx = 1;
        add(numberButtons[5], constraints);
        constraints.gridx = 2;
        add(numberButtons[6], constraints);
        constraints.gridx = 3;
        add(subtractButton, constraints);

        constraints.gridy = 3;
        constraints.gridx = 0;
        add(numberButtons[1], constraints);
        constraints.gridx = 1;
        add(numberButtons[2], constraints);
        constraints.gridx = 2;
        add(numberButtons[3], constraints);
        constraints.gridx = 3;
        add(multiplyButton, constraints);

        constraints.gridy = 4;
        constraints.gridx = 0;
        add(numberButtons[0], constraints);
        constraints.gridx = 1;
        constraints.gridwidth = 2;
        add(equalButton, constraints);
        constraints.gridx = 3;
        constraints.gridwidth = 1;
        add(divideButton, constraints);

        constraints.gridy = 5;
        constraints.gridx = 0;
        constraints.gridwidth = 4;
        add(clearButton, constraints);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        if (command.charAt(0) >= '0' && command.charAt(0) <= '9') {
            inputField.setText(inputField.getText() + command);
        } else if (command.charAt(0) == 'C') {
            inputField.setText("");
            num1 = num2 = result = 0;
            operator = '\0';
        } else if (command.charAt(0) == '=') {
```

```
                num2 = Double.parseDouble(inputField.getText());
                switch (operator) {
                    case '+':
                        result = num1 + num2;
                        break;
                    case '-':
                        result = num1 - num2;
                        break;
                    case '*':
                        result = num1 * num2;
                        break;
                    case '/':
                        result = num1 / num2;
                        break;
                }
                inputField.setText(String.valueOf(result));
                num1 = result;
                operator = '\0';
            } else {
                num1 = Double.parseDouble(inputField.getText());
                operator = command.charAt(0);
                inputField.setText("");
            }
        }

        public static void main(String[] args) {
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    new BasicCalculator();
                }
            });
        }
    }
```

16. Question No. 16

```
    import javax.swing.*;
    import java.awt.*;
    import java.awt.event.ActionEvent;
    import java.awt.event.ActionListener;
    import java.sql.*;

    public class LoginForm extends JFrame implements ActionListener {
        private JTextField usernameField;
        private JPasswordField passwordField;
        private JButton loginButton;
```

```
    public LoginForm() {
        setTitle("Login Form");
        setSize(300, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Initialize components
        JLabel usernameLabel = new JLabel("Username:");
        JLabel passwordLabel = new JLabel("Password:");
        usernameField = new JTextField(20);
        passwordField = new JPasswordField(20);
        loginButton = new JButton("Login");
        loginButton.addActionListener(this);

        // Create the layout using GridBagLayout
        setLayout(new GridBagLayout());
        GridBagConstraints constraints = new GridBagConstraints();
        constraints.gridx = 0;
        constraints.gridy = 0;
        add(usernameLabel, constraints);
        constraints.gridx = 1;
        add(usernameField, constraints);
        constraints.gridy = 1;
        add(passwordLabel, constraints);
        constraints.gridx = 1;
        add(passwordField, constraints);
        constraints.gridy = 2;
        constraints.gridx = 0;
        constraints.gridwidth = 2;
        add(loginButton, constraints);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == loginButton) {
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());

            // TODO: Perform database check for valid login credentials here
            // For demonstration purposes, let's assume username and password are both
"admin"
            if ("admin".equals(username) && "admin".equals(password)) {
                // If login is successful, open the next form (insert, delete, update)
                new StudentForm();
                dispose(); // Close the login form
            } else {
```

```
            JOptionPane.showMessageDialog(this, "Invalid username or password.",
"Login Failed", JOptionPane.ERROR_MESSAGE);
        }
      }
    }

    public static void main(String[] args) {
      SwingUtilities.invokeLater(new Runnable() {
        public void run() {
          new LoginForm();
        }
      });
    }
}



import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class StudentForm extends JFrame implements ActionListener {
    private JButton insertButton;
    private JButton deleteButton;
    private JButton updateButton;
    private JButton loadButton;

    public StudentForm() {
      setTitle("Student Form");
      setSize(300, 150);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      // Initialize components
      insertButton = new JButton("Insert");
      deleteButton = new JButton("Delete");
      updateButton = new JButton("Update");
      loadButton = new JButton("Load All");
      insertButton.addActionListener(this);
      deleteButton.addActionListener(this);
      updateButton.addActionListener(this);
      loadButton.addActionListener(this);

      // Create the layout using GridBagLayout
      setLayout(new GridBagLayout());
      GridBagConstraints constraints = new GridBagConstraints();
      constraints.gridx = 0;
```

```
        constraints.gridy = 0;
        add(insertButton, constraints);
        constraints.gridx = 1;
        add(deleteButton, constraints);
        constraints.gridy = 1;
        constraints.gridx = 0;
        constraints.gridwidth = 2;
        add(updateButton, constraints);
        constraints.gridy = 2;
        constraints.gridwidth = 2;
        add(loadButton, constraints);

        setVisible(true);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO: Implement insert, delete, update, and load operations here
        if (e.getSource() == insertButton) {
            // Handle insert operation
        } else if (e.getSource() == deleteButton) {
            // Handle delete operation
        } else if (e.getSource() == updateButton) {
            // Handle update operation
        } else if (e.getSource() == loadButton) {
            // Handle load operation
        }
    }
}
```