Java Lab Manual

ITM Skills University

B.Tech CSE- Second Year

Student Name: Sakshi Kore

Roll No: 33

Batch: Steve Jobs

Date: 24 August 2024

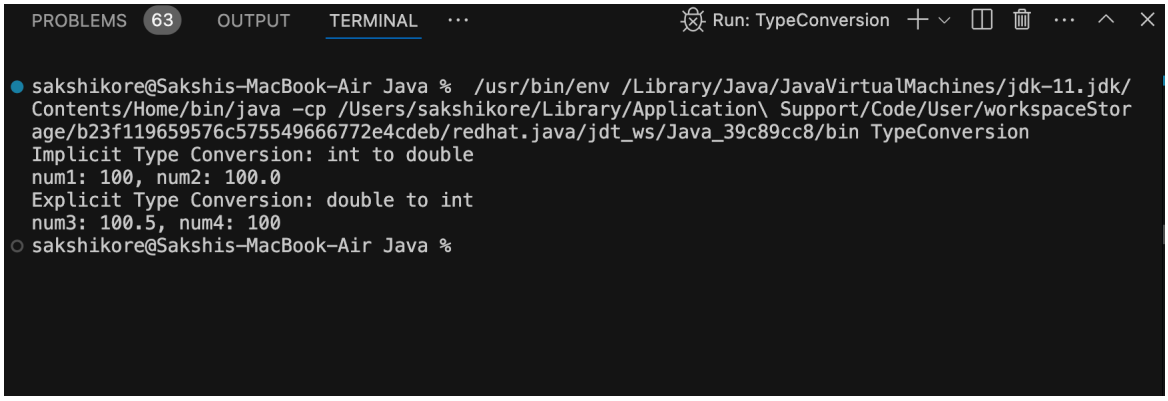
Module 1

Question 1: WAP to demonstrate implicit type conversion and explicit type conversion.

Code:

```
public class TypeConversion
{
    public static void main(String[] args)
    {
        int num1 = 100;
        double num2 = num1;
        System.out.println("Implicit Type Conversion: int to
double");
        System.out.println("num1: " + num1 + ", num2: " +
num2);

        double num3 = 100.5;
        int num4 = (int) num3;
        System.out.println("Explicit Type Conversion: double
to int");
        System.out.println("num3: " + num3 + ", num4: " +
num4);
    }
}
```

Output:



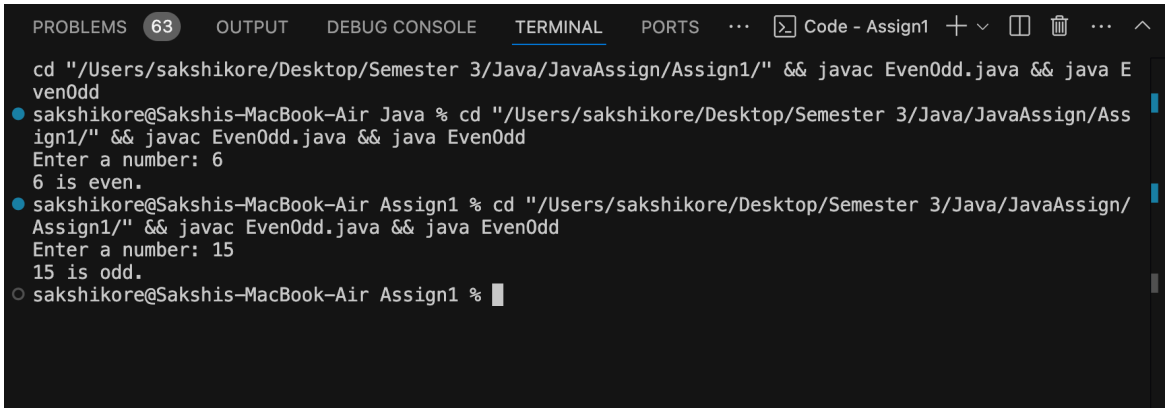
————————————————————XXX————————————————————

Question 2: WAP to find whether the inputted number is even or odd.

Code:

```java
import java.util.Scanner;

public class EvenOdd
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = sc.nextInt();

        if (number % 2 == 0)
        {
            System.out.println(number + " is even.");
        }
        else
        {
            System.out.println(number + " is odd.");
        }
        sc.close();
    }
}
```

Output:



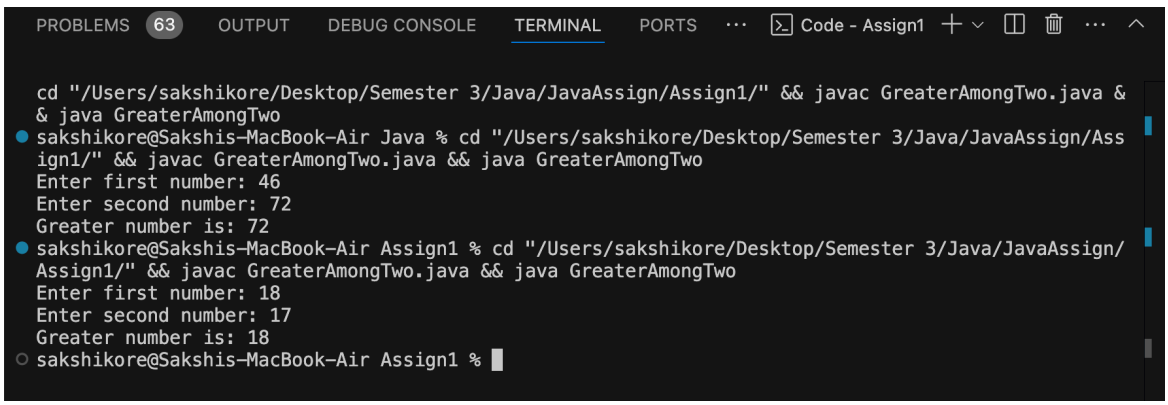
————————————————————XXX————————————————————

Question 3: WAP to find greater among two numbers using conditional operator.

Code:

```java
import java.util.Scanner;

public class GreaterAmongTwo
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter first number: ");
        int num1 = sc.nextInt();

        System.out.print("Enter second number: ");
        int num2 = sc.nextInt();

        int greater = (num1 > num2) ? num1 : num2;
        System.out.println("Greater number is: " + greater);
        sc.close();
    }
}
```

Output:



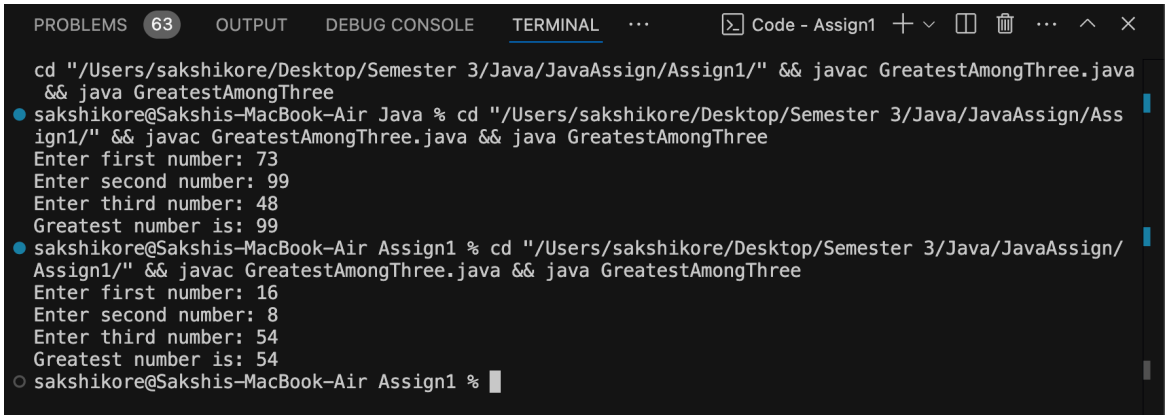
——————————————————XXX——————————————————

Question 4: WAP to find greatest among three numbers using if else.

Code:

```
import java.util.Scanner;

public class GreatestAmongThree
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter first number: ");
        int num1 = sc.nextInt();
        System.out.print("Enter second number: ");
        int num2 = sc.nextInt();
        System.out.print("Enter third number: ");
        int num3 = sc.nextInt();

        if (num1 >= num2 && num1 >= num3)
        {
            System.out.println("Greatest number is: " +
num1);
        } else if (num2 >= num1 && num2 >= num3) {
            System.out.println("Greatest number is: " +
num2);
        } else {
            System.out.println("Greatest number is: " +
num3);
        }
        sc.close();
    }
}
```

Output:



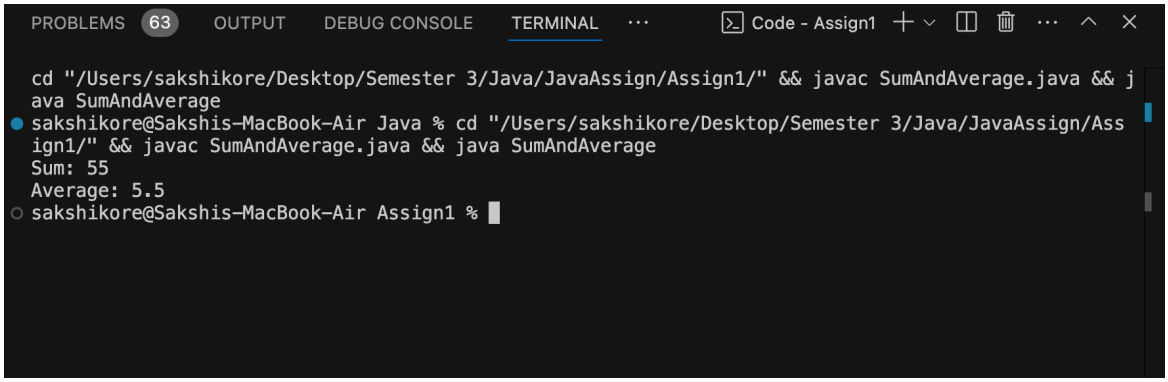
————————————————————XXX————————————————————

Question 5: WAP to find sum and average of numbers from 1 to 10.

Code:

```java
public class SumAndAverage
{
    public static void main(String[] args)
    {
        int sum = 0;

        for (int i = 1; i <= 10; i++)
        {
            sum += i;
        }
        double average = sum / 10.0;
        System.out.println("Sum: " + sum);
        System.out.println("Average: " + average);
    }
}
```

Output:



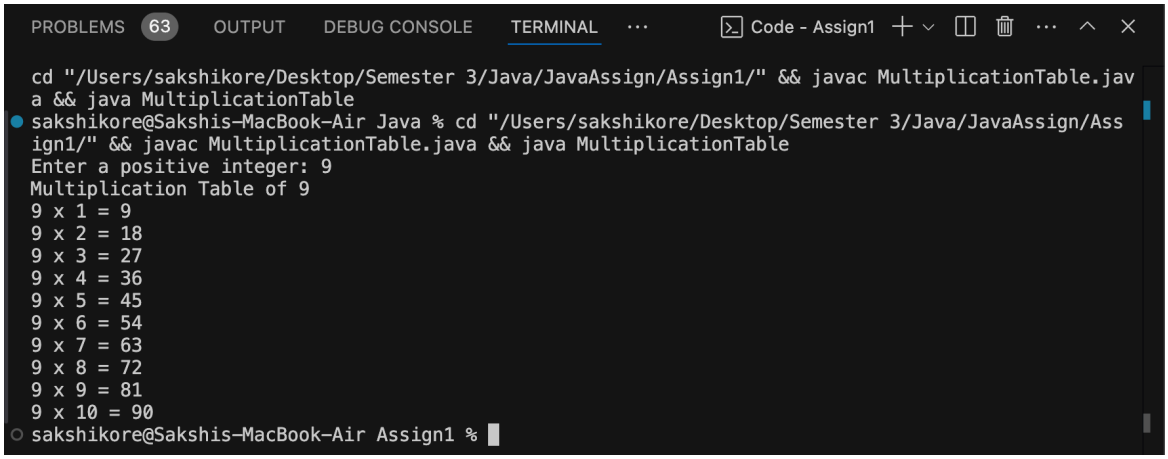
——————————————————————XXX——————————————————————

Question 6: Write a program that prompts the user to input a positive integer. It should then print the multiplication table of that number.

Code:

```
import java.util.Scanner;

public class MultiplicationTable
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        int num = sc.nextInt();

        System.out.println("Multiplication Table of " +
num);
        for (int i = 1; i <= 10; i++)
        {
            System.out.println(num + " x " + i + " = " + num
* i);
        }
        sc.close();
    }
}
```

Output:



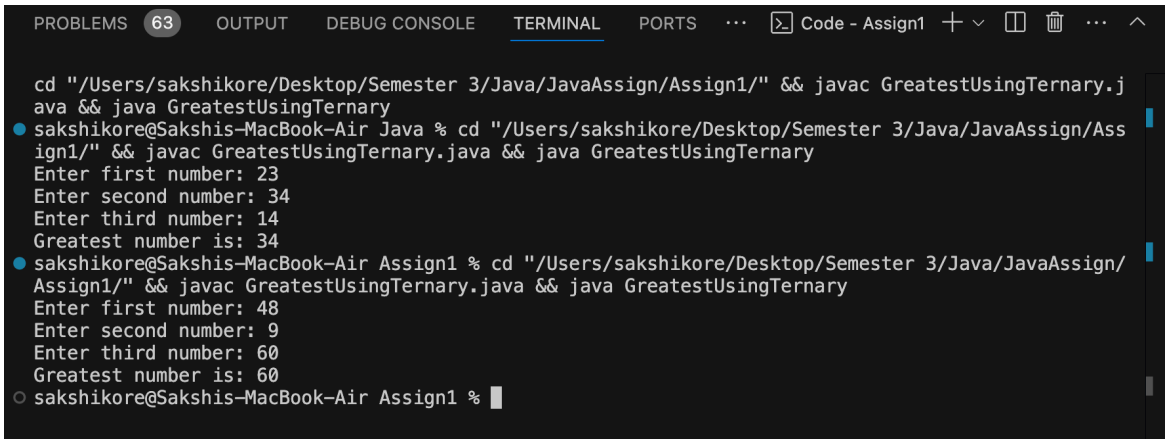
————————————————————XXX————————————————————

Question 7: WAP to find greatest among three numbers using conditional operator.

Code:

```
import java.util.Scanner;

public class GreatestUsingTernary
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter first number: ");
        int num1 = sc.nextInt();
        System.out.print("Enter second number: ");
        int num2 = sc.nextInt();
        System.out.print("Enter third number: ");
        int num3 = sc.nextInt();

        int greatest = (num1 > num2 && num1 > num3) ? num1 :
(num2 > num3) ? num2 : num3;
        System.out.println("Greatest number is: " +
greatest);
        sc.close();
    }
}
```

Output:



——————————————————XXX——————————————————

Question 8: WAP to print odd numbers between 1 to 20.

Code:

```java
public class OddNumbers
{
    public static void main(String[] args)
    {
        System.out.println("Odd numbers between 1 and 20:");

        for (int i = 1; i <= 20; i++)
        {
            if (i % 2 != 0)
            {
                System.out.println(i);
            }
        }
    }
}
```

Output:

```
PROBLEMS 63  OUTPUT  DEBUG CONSOLE  TERMINAL  ···  Code - Assign1

cd "/Users/sakshikore/Desktop/Semester 3/Java/JavaAssign/Assign1/" && javac OddNumbers.java && java
 OddNumbers
sakshikore@Sakshis-MacBook-Air Java % cd "/Users/sakshikore/Desktop/Semester 3/Java/JavaAssign/Ass
ign1/" && javac OddNumbers.java && java OddNumbers
Odd numbers between 1 and 20:
1
3
5
7
9
11
13
15
17
19
sakshikore@Sakshis-MacBook-Air Assign1 %
```

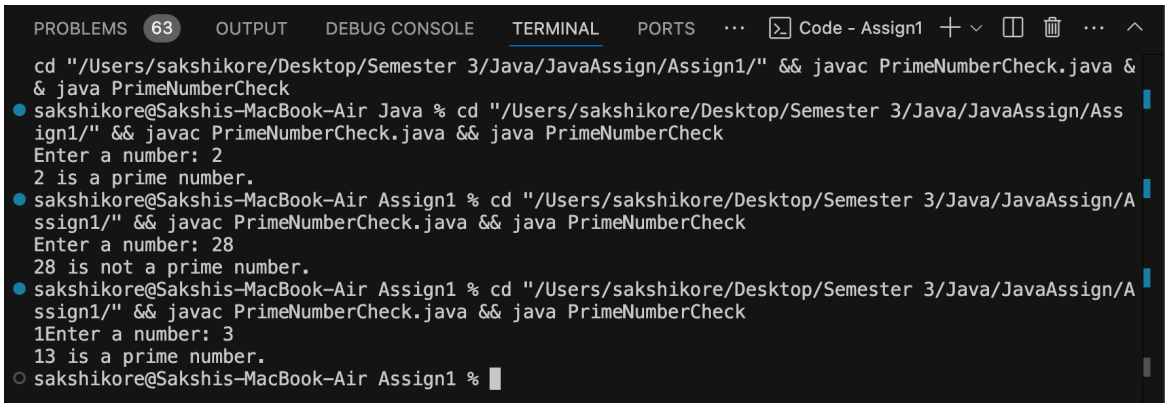
————————————————————XXX————————————————————

Question 9: WAP to find whether a number is prime or not.

Code:

```
import java.util.Scanner;

public class PrimeNumberCheck 
{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();

        boolean isPrime = true;
        if (num <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= num / 2; i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        if (isPrime) {
            System.out.println(num + " is a prime number.");
        } else {
            System.out.println(num + " is not a prime 
number.");
        }
        sc.close();
    }
}
```

Output:



——————————————————————XXX——————————————————————

Question 10: Write a Java Program find out Students Grades using Switch Case.

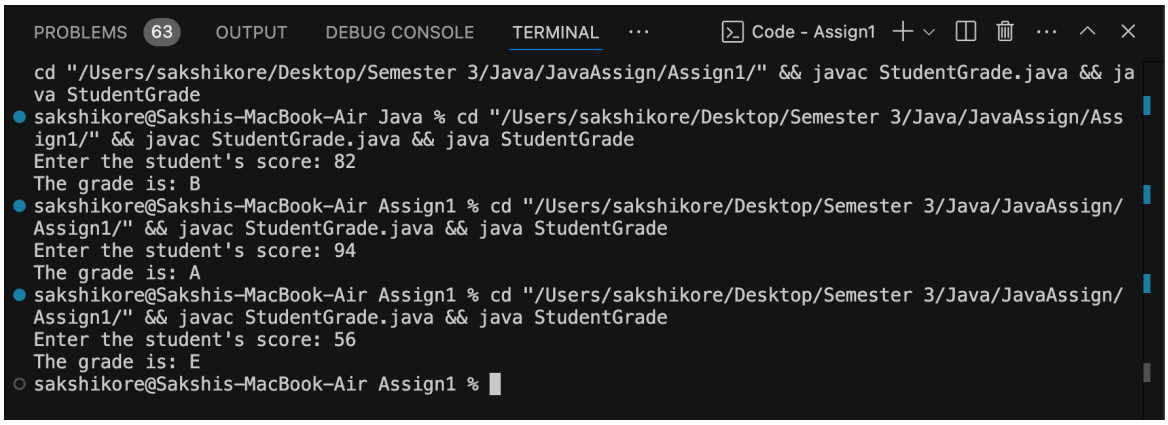
Score in subject	Grade
>=90	A
80-89	B
70-79	C
60-69	D
50-59	E
<50	F

Code:

```
import java.util.Scanner;

public class StudentGrade
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the student's score: ");
        int score = sc.nextInt();
        String grade;

        switch (score / 10)
        {
            case 10:

            case 9:
                grade = "A";
                break;

            case 8:
                grade = "B";
                break;

            case 7:
                grade = "C";
                break;

            case 6:
```

```
                grade = "D";
                break;

            case 5:
                grade = "E";
                break;

            default:
                grade = "F";
        }

        System.out.println("The grade is: " + grade);

        sc.close();
    }
}
```

Output:



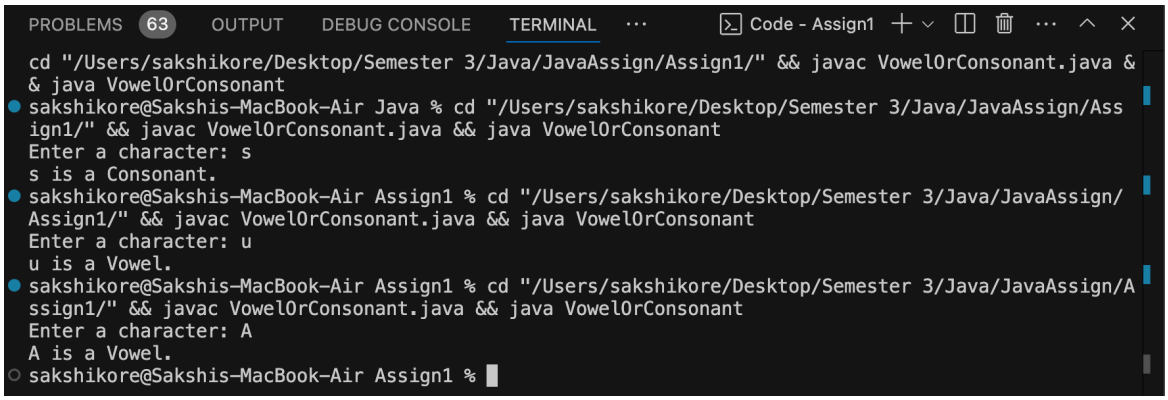
——————————————————————XXX——————————————————————

Question 11: WAP to check whether the inputted character is Vowel or Consonant.

Code:

```
import java.util.Scanner;

public class VowelOrConsonant
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a character: ");
        char ch = sc.next().charAt(0);

        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o'
|| ch == 'u' ||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O'
|| ch == 'U')
            {
            System.out.println(ch + " is a Vowel.");
        }
        else
        {
            System.out.println(ch + " is a Consonant.");
        }

        sc.close();
    }
}
```

Output:



————————————————————XXX————————————————————

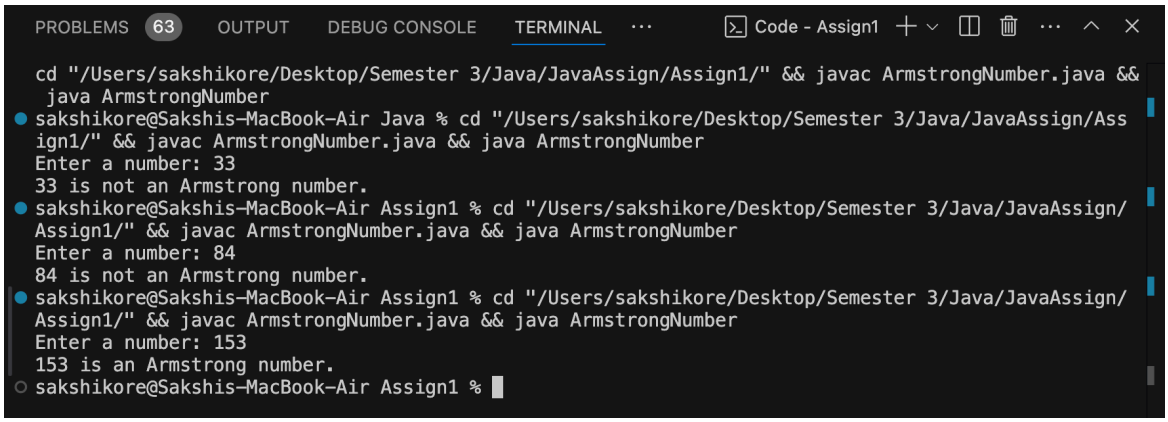
Question 12: WAP to check whether the inputted number is Armstrong Number or not. (Hint: An Armstrong number is a positive m-digit number that is equal to the sum of the mth powers of their digits. It is also known as pluperfect, or Plus Perfect, or Narcissistic number.
153: 13 + 53 + 33 = 1 + 125+ 27 = 153 (An Armstrong Number),
125: 13 + 23 + 53 = 1 + 8 + 125 = 134 (Not an Armstrong Number)
Hint: use Math.pow(num1, num2) to calculate power

Code:

```
import java.util.Scanner;

public class ArmstrongNumber
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int number = sc.nextInt();
        int originalNumber = number;
        int result = 0;
        int n = 0;

        while (originalNumber != 0)
        {
            originalNumber /= 10;
            ++n;
        }

        originalNumber = number;

        while (originalNumber != 0)
        {
            int digit = originalNumber % 10;
            result += Math.pow(digit, n);
            originalNumber /= 10;
        }

        if (result == number)
        {
            System.out.println(number + " is an Armstrong
number.");
        }
```

```
        else
            {
                System.out.println(number + " is not an
Armstrong number.");
            }

            sc.close();
        }
}
```

Output:



——————————————————XXX——————————————————

Question 13: Write a program that generates a random number between 1 to 100 and asks the user to guess what the number is. If the user's guess is higher than the random number, the program should display "Too high, try again." If the user's guess is lower than the random number, the program should display "Too low, try again." The program should use a loop that repeats until the user correctly guesses the random number (Hint: use Math.random() for generating random number. Eg. number = (int) (Math.random() * 100) + 1;).

Code:

```
import java.util.Scanner;
public class GuessTheNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int number = (int) (Math.random() * 100) + 1;
        int guess = 0;
        while (guess != number) {
            System.out.print("Guess the number (between 1
and 100): ");
            guess = sc.nextInt();
            if (guess < number) {
                System.out.println("Too low, try again.");
            } else if (guess > number) {
                System.out.println("Too high, try again.");
            } else {
                System.out.println("Correct! The number was
" + number);
            }
        }
        sc.close();
    }
}
```

Output:

————————————————————XXX————————————————————

Question 14: WAP to find average of consecutive N Odd numbers and even numbers.

Code:

```
import java.util.Scanner;
public class AverageOddEven
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of terms (N): ");
        int n = sc.nextInt();

        int sumOdd = 0;
        int sumEven = 0;
        for (int i = 1; i <= n; i++)
        {
            sumOdd += 2 * i - 1;
            sumEven += 2 * i;
        }
        double avgOdd = sumOdd / (double) n;
        double avgEven = sumEven / (double) n;

        System.out.println("Average of first " + n + " odd
numbers: " + avgOdd);
        System.out.println("Average of first " + n + " even
numbers: " + avgEven);
        sc.close();
    }
}
```

Output:

```
PROBLEMS 63  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ···  Code - Assign1

cd "/Users/sakshikore/Desktop/Semester 3/Java/JavaAssign/Assign1/" && javac AverageOddEven.java &&
 java AverageOddEven
sakshikore@Sakshis-MacBook-Air Java % cd "/Users/sakshikore/Desktop/Semester 3/Java/JavaAssign/Ass
ign1/" && javac AverageOddEven.java && java AverageOddEven
Enter the number of terms (N): 4
Average of first 4 odd numbers: 4.0
Average of first 4 even numbers: 5.0
sakshikore@Sakshis-MacBook-Air Assign1 % cd "/Users/sakshikore/Desktop/Semester 3/Java/JavaAssign/
Assign1/" && javac AverageOddEven.java && java AverageOddEven
Enter the number of terms (N): 13
Average of first 13 odd numbers: 13.0
Average of first 13 even numbers: 14.0
sakshikore@Sakshis-MacBook-Air Assign1 %
```

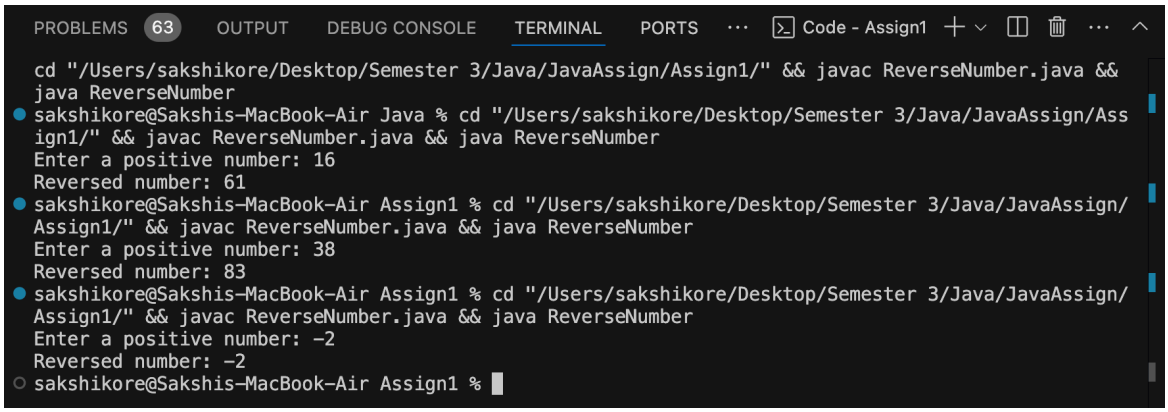
————————————————————XXX————————————————————

Question 15: WAP to reverse a positive number.

Code:

```
import java.util.Scanner;

public class ReverseNumber
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a positive number: ");
        int number = sc.nextInt();
        int reversedNumber = 0;

        while (number != 0)
        {
            int digit = number % 10;
            reversedNumber = reversedNumber * 10 + digit;
            number /= 10;
        }

        System.out.println("Reversed number: " +
reversedNumber);

        sc.close();
    }
}
```

Output:



————————————————————XXX————————————————————

Module 2

Question 1: WAP to create a class called Circle**.** It contains:

- Private instance variable: radius (of the type double) with default value of 1.0.
- Two overloaded constructors - a default constructor with no argument, and a constructor which takes a double argument for radius.
- Two public methods: getRadius(), calculateArea(), calculateCircumference() which return the radius, calculate and return area, and circumference respectively.

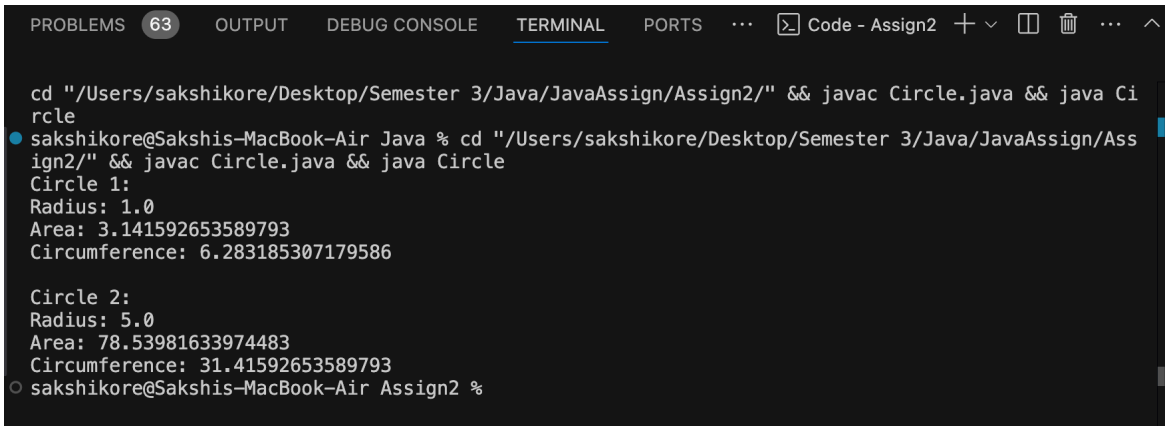
Hint: Use Math.PI for calculating area and circumference

Code:

```
class Circle {
    private double radius;

    public Circle() {
        this.radius = 1.0;
    }

    public Circle(double radius) {
        this.radius = radius;
    }

    public double getRadius() {
        return this.radius;
    }

    public double calculateArea() {
        return Math.PI * Math.pow(this.radius, 2);
    }

    public double calculateCircumference() {
        return 2 * Math.PI * this.radius;
    }

    public static void main(String[] args) {
        Circle circle1 = new Circle();
        System.out.println("Circle 1:");
        System.out.println("Radius: " +
circle1.getRadius());
```

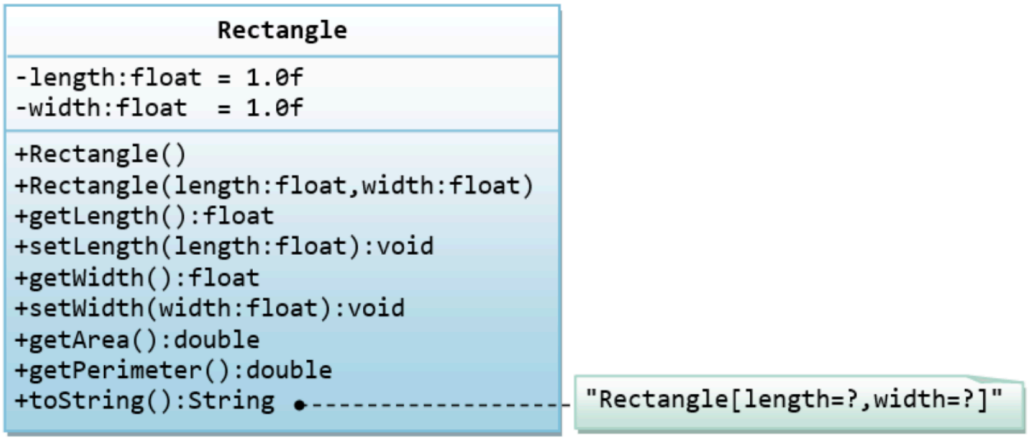
```
        System.out.println("Area: " +
circle1.calculateArea());
        System.out.println("Circumference: " +
circle1.calculateCircumference());

        Circle circle2 = new Circle(5.0);
        System.out.println("\nCircle 2:");
        System.out.println("Radius: " +
circle2.getRadius());
        System.out.println("Area: " +
circle2.calculateArea());
        System.out.println("Circumference: " +
circle2.calculateCircumference());
    }
}
```

Output:



————————————————————XXX————————————————————

Question 2: A class called Rectangle, which models a rectangle with a length and a width (in float), is designed as shown in the following class diagram. Write the Rectangle class as per UML diagram.

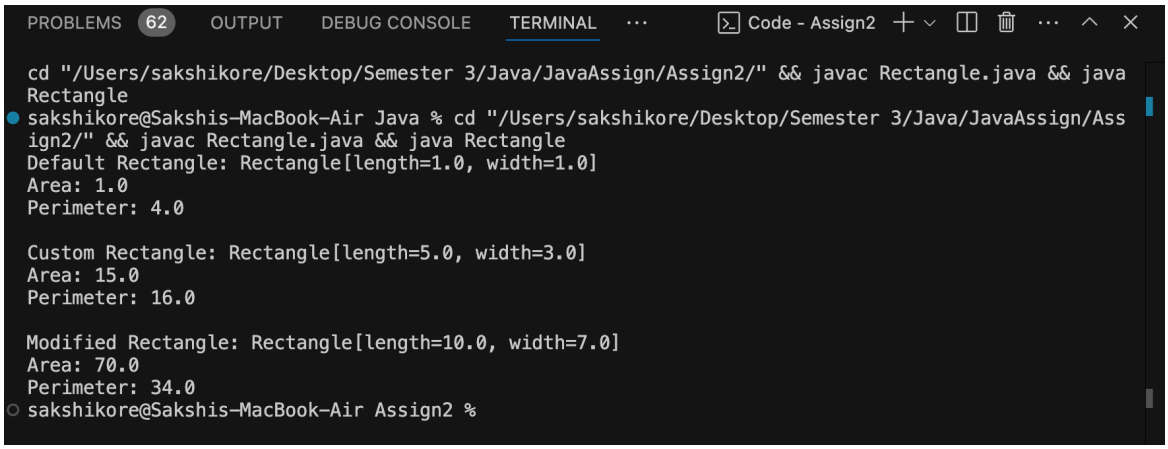


Code:

```
public class Rectangle {
    private float length = 1.0f;
    private float width = 1.0f;

    public Rectangle() {
    }

    public Rectangle(float length, float width) {
        this.length = length;
        this.width = width;
    }

    public float getLength() {
        return length;
    }

    public void setLength(float length) {
        this.length = length;
    }

    public float getWidth() {
        return width;
    }

    public void setWidth(float width) {
        this.width = width;
    }

    public double getArea() {
        return length * width;
```

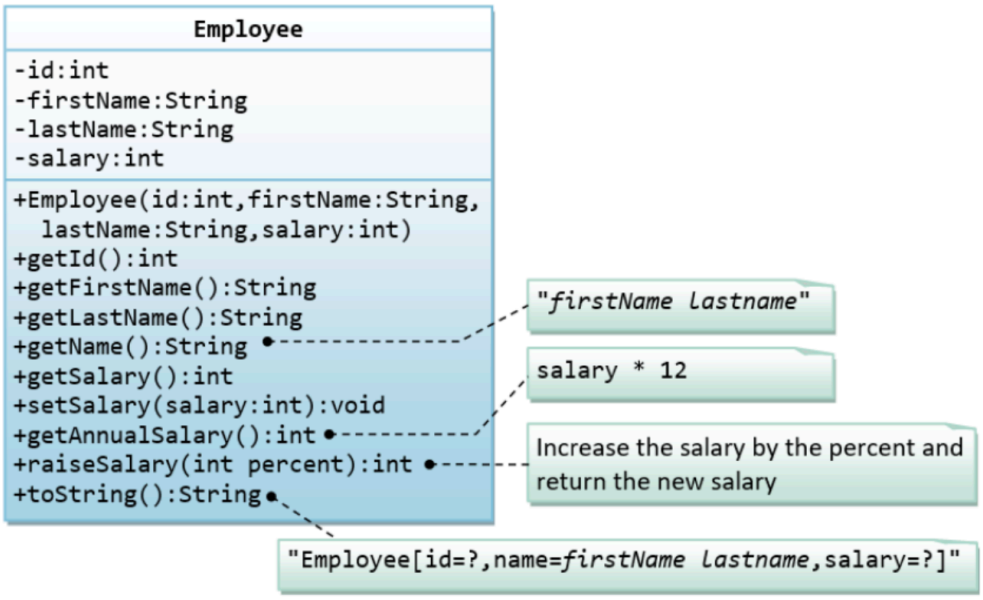
```
    }

    public double getPerimeter() {
        return 2 * (length + width);
    }

    @Override
    public String toString() {
        return "Rectangle[length=" + length + ", width=" +
width + "]";
    }

    public static void main(String[] args) {
        Rectangle rect1 = new Rectangle();
        System.out.println("Default Rectangle: " + rect1);
        System.out.println("Area: " + rect1.getArea());
        System.out.println("Perimeter: " +
rect1.getPerimeter());

        Rectangle rect2 = new Rectangle(5.0f, 3.0f);
        System.out.println("\nCustom Rectangle: " + rect2);
        System.out.println("Area: " + rect2.getArea());
        System.out.println("Perimeter: " +
rect2.getPerimeter());

        rect2.setLength(10.0f);
        rect2.setWidth(7.0f);
        System.out.println("\nModified Rectangle: " +
rect2);
        System.out.println("Area: " + rect2.getArea());
        System.out.println("Perimeter: " +
rect2.getPerimeter());
    }
}
```

Output:



————————————————————XXX————————————————————

Question 3: A class called Employee, which models an employee with an ID, name and salary, is designed as shown in the following class diagram. The method raiseSalary(percent) increases the salary by the given percentage. Write the Employee class and its driver class.



Code:

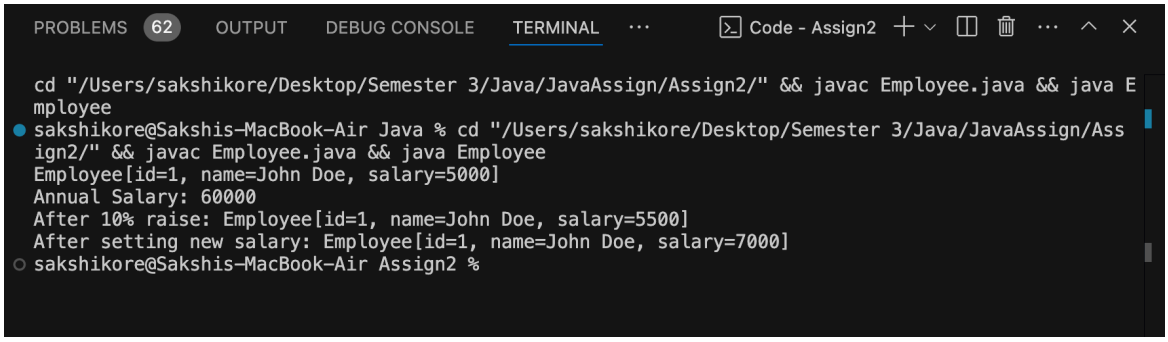
```
public class Employee {
    private int id;
    private String firstName;
    private String lastName;
    private int salary;
```

```
    public Employee(int id, String firstName, String
lastName, int salary) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.salary = salary;
    }
```

```
    public int getId() {
        return id;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
```

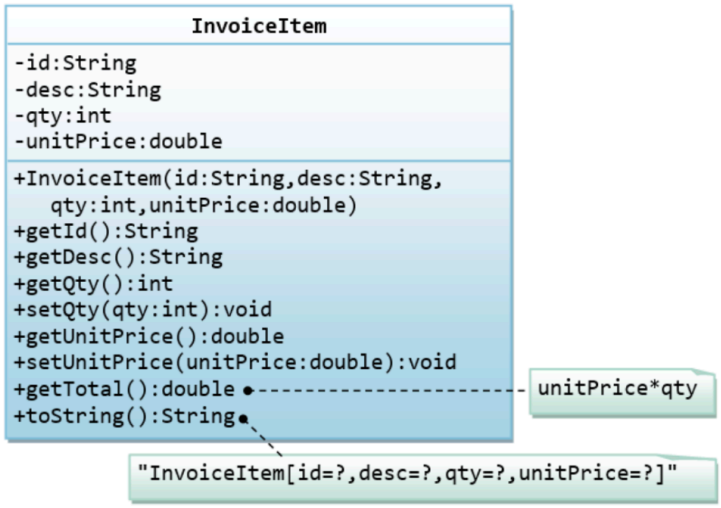
```
    public String getName() {
        return firstName + " " + lastName;
    }
    public int getSalary() {
        return salary;
    }
    public void setSalary(int salary) {
        this.salary = salary;
    }
    public int getAnnualSalary() {
        return salary * 12;
    }
    public int raiseSalary(int percent) {
        this.salary += (this.salary * percent) / 100;
        return this.salary;
    }

    @Override
    public String toString() {
        return "Employee[id=" + id + ", name=" + getName() +
", salary=" + salary + "]";
    }

    public static void main(String[] args) {
        Employee emp = new Employee(1, "John", "Doe", 5000);
        System.out.println(emp);
        System.out.println("Annual Salary: " +
emp.getAnnualSalary());
        emp.raiseSalary(10);
        System.out.println("After 10% raise: " + emp);
        emp.setSalary(7000);
        System.out.println("After setting new salary: " +
emp);
    }
}
```

Output:



————————————————————XXX————————————————————

Question 4: A class called InvoiceItem, which models an item of an invoice, with ID, description, quantity and unit price, is designed as shown in the following class diagram. It has a method getTotal which calculates total value (total=quantity*unit price). Write the InvoiceItem class and it's driver class.

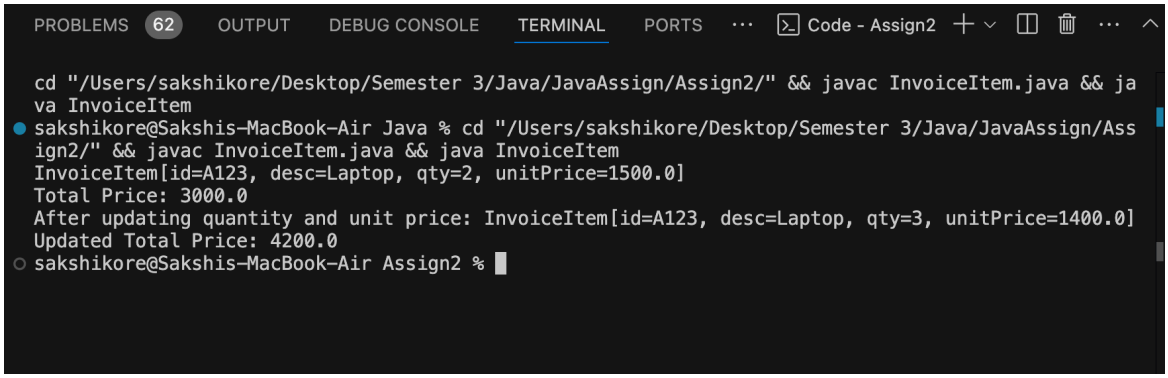


Code:

```
public class InvoiceItem {
    private String id;
    private String desc;
    private int qty;
    private double unitPrice;

    public InvoiceItem(String id, String desc, int qty,
double unitPrice) {
        this.id = id;
        this.desc = desc;
        this.qty = qty;
        this.unitPrice = unitPrice;
    }

    public String getId() {
        return id;
    }
    public String getDesc() {
        return desc;
    }
    public int getQty() {
        return qty;
    }
    public void setQty(int qty) {
        this.qty = qty;
    }
```

```
    public double getUnitPrice() {
        return unitPrice;
    }
    public void setUnitPrice(double unitPrice) {
        this.unitPrice = unitPrice;
    }
    public double getTotal() {
        return qty * unitPrice;
    }

    @Override
    public String toString() {
        return "InvoiceItem[id=" + id + ", desc=" + desc +
", qty=" + qty + ", unitPrice=" + unitPrice + "]";
    }

    public static void main(String[] args) {
        InvoiceItem item = new InvoiceItem("A123", "Laptop",
2, 1500.00);
        System.out.println(item);
        System.out.println("Total Price: " +
item.getTotal());
        item.setQty(3);
        item.setUnitPrice(1400.00);
        System.out.println("After updating quantity and unit
price: " + item);
        System.out.println("Updated Total Price: " +
item.getTotal());
    }
}
```

Output:



————————————————————XXX————————————————————

Question 5: A class called Author (as shown in the class diagram) is designed to model a book's author. It contains:

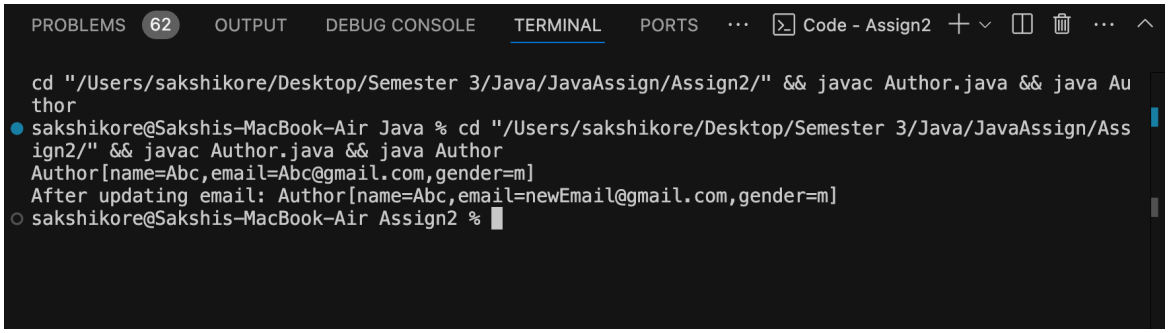
- Three private instance variables: name (String), email (String), and gender (char of either 'm' or 'f');
- One constructor to initialise the name, email and gender with the given values; public Author (String name, String email, char gender) {.....} (There is no default constructor for Author, as there are no defaults for name, email and gender.)
- Public getters/setters: getName(), getEmail(), setEmail(), and getGender(); (There are no setters for name and gender, as these attributes cannot be changed.)
- A toString() method that returns "Author[name=?,email=?,gender=?]", e.g., "Author[name=Abc ,email=Abc@gmail.com, gender=m]".

Code:

```
public class Author {
    private String name;
    private String email;
    private char gender;

    public Author(String name, String email, char gender) {
        this.name = name;
        this.email = email;
        this.gender = gender;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public char getGender() {
        return gender;
    }
```

```
    @Override
    public String toString() {
        return "Author[name=" + name + ",email=" + email +
",gender=" + gender + "]";
    }

    public static void main(String[] args) {
        Author author = new Author("Abc", "Abc@gmail.com",
'm');

        System.out.println(author);

        author.setEmail("newEmail@gmail.com");
        System.out.println("After updating email: " +
author);
    }
}
```

Output:



——————————————————————XXX——————————————————————

Question 6: WAP to create a class time having default constructor, parameterised constructor whose specifications are as follows:

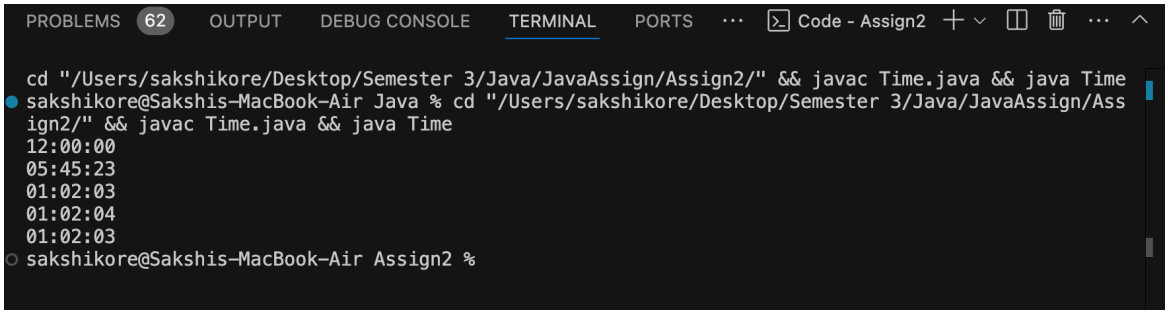
- Instance variable: hr, min, sec
- Constructors:
  - Default (with no parameters passed; should initialize the represented time to 12:0:0)
  - a constructor with three parameters: hours, minutes, and seconds.
  - a constructor with one parameter: the value of time in seconds since midnight (it should be converted into the time value in hours, minutes, and seconds)
- Instance methods:
  - setClock() with one parameter *seconds* since midnight (to be converted into the time value in hours, minutes, and seconds as above).
  - tick() with no parameters that increments the time stored in a Clock object by one second.
  - tickDown() which decrements the time stored in a Clock object by one second.
  - displaytime() displays the time in the format hr: min:sec e.g: 05:45:23

Code:

```
public class Time {
    private int hr;
    private int min;
    private int sec;

    public Time() {
        this.hr = 12;
        this.min = 0;
        this.sec = 0;
    }

    public Time(int hr, int min, int sec) {
        this.hr = hr;
        this.min = min;
        this.sec = sec;
    }
    public Time(int totalSeconds) {
```

```
        setClock(totalSeconds);
    }
    public void setClock(int totalSeconds) {
        this.hr = totalSeconds / 3600;
        this.hr = this.hr % 24;
        totalSeconds %= 3600;
        this.min = totalSeconds / 60;
        this.sec = totalSeconds % 60;
    }

    public void tick() {
        setClock(getTotalSeconds() + 1);
    }
    public void tickDown() {
        setClock(getTotalSeconds() - 1);
    }
    private int getTotalSeconds() {
        return hr * 3600 + min * 60 + sec;
    }
    public void displayTime() {
        System.out.printf("%02d:%02d:%02d\n", hr, min, sec);
    }

    public static void main(String[] args) {
        Time time1 = new Time();
        time1.displayTime();
        Time time2 = new Time(5, 45, 23);
        time2.displayTime();
        Time time3 = new Time(3723);
        time3.displayTime();
        time3.tick();
        time3.displayTime();
        time3.tickDown();
        time3.displayTime();
    }
}
```

Output:



————————————————————XXX————————————————————

Question 7: Write a Java class Complex for dealing with complex number. Your class must have the following features:

- Instance variables :
  - real for the real part of type double
  - imag for imaginary part of type double.
- Constructor:
  - public Complex (): A default constructor, it should initialize the number to 0, 0)
  - public Complex (double real, double imag: A constructor with parameters, it creates the complex object by setting the two fields to the passed values.
- Instance methods:
  - public Complex add (Complex n): This method will find the sum of the current complex number and the passed complex number. The methods returns a new Complex number which is the sum of the two.
  - public Complex subtract (Complex n): This method will find the difference of the current complex number and the passed complex number. The methods returns a new Complex number which is the difference of the two.
  - public void setReal(double real): Used to set the real part of this complex number.
  - public void setImag(double image): Used to set the imaginary part of this complex number.
  - public double getReal(): This method returns the real part of the complex number
  - public double getImag(): This method returns the imaginary part of the complex number
  - public String toString(): This method allows the complex number to be easily printed out to the screen

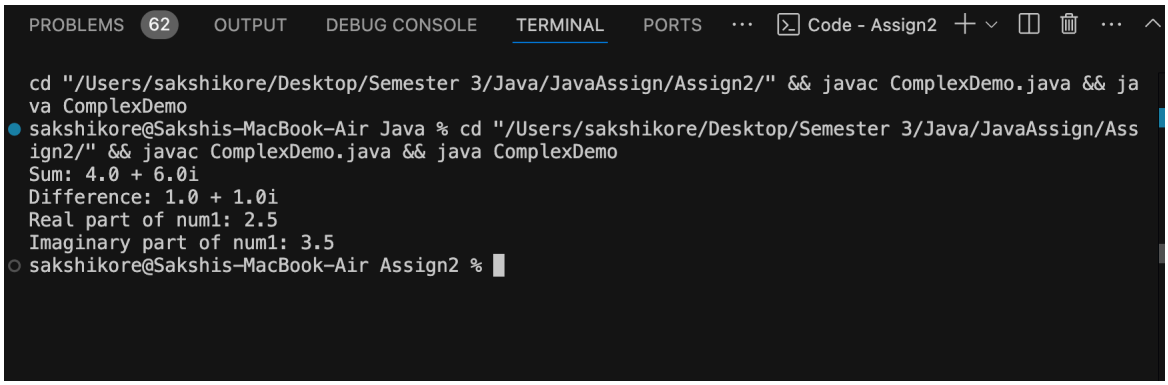
Write a separate class ComplexDemo with a main() method and test the Complex class methods.

Code:

```
class Complex {
    private double real;
    private double imag;

    public Complex() {
        this.real = 0;
        this.imag = 0;
    }

    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }

    public Complex add(Complex n) {
        return new Complex(this.real + n.real, this.imag +
n.imag);
    }

    public Complex subtract(Complex n) {
        return new Complex(this.real - n.real, this.imag -
n.imag);
    }

    public void setReal(double real) {
        this.real = real;
    }

    public void setImag(double imag) {
        this.imag = imag;
    }

    public double getReal() {
        return this.real;
    }

    public double getImag() {
        return this.imag;
    }

    @Override
    public String toString() {
        return this.real + " + " + this.imag + "i";
    }
}
```

```java
public class ComplexDemo {
    public static void main(String[] args) {
        Complex num1 = new Complex(2.5, 3.5);
        Complex num2 = new Complex(1.5, 2.5);

        Complex sum = num1.add(num2);
        System.out.println("Sum: " + sum);

        Complex difference = num1.subtract(num2);
        System.out.println("Difference: " + difference);

        System.out.println("Real part of num1: " +
num1.getReal());
        System.out.println("Imaginary part of num1: " +
num1.getImag());
    }
}
```

Output:



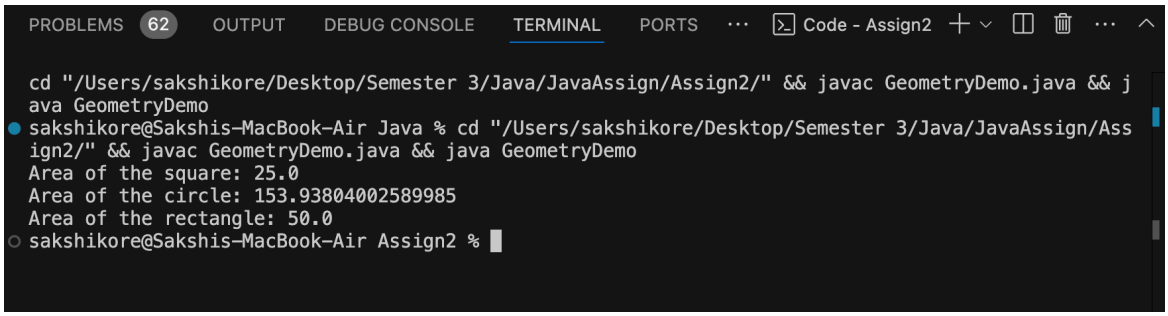
——————————————————XXX——————————————————

Question 8: Create a Geometry class and overload calculateArea method for square, circle, and rectangle.

Code:

```
class Geometry {

    public double calculateArea(double side) {
        return side * side;
    }
    public double calculateArea(double radius, boolean
isCircle) {
        return Math.PI * radius * radius;
    }
    public double calculateArea(double length, double width)
{
        return length * width;
    }
}

public class GeometryDemo {
    public static void main(String[] args) {
        Geometry geometry = new Geometry();
        double squareArea = geometry.calculateArea(5);
        System.out.println("Area of the square: " +
squareArea);
        double circleArea = geometry.calculateArea(7, true);
        System.out.println("Area of the circle: " +
circleArea);
        double rectangleArea = geometry.calculateArea(10,
5);
        System.out.println("Area of the rectangle: " +
rectangleArea);
    }
}
```

Output:



————————————————————XXX————————————————————

Question 9: WAP to implement Box class (Hint: Refer Java PPT). Inherit Box class in BoxWt whose

a. instance variable is weight and
b. method is print_BoxWt()
c. constructors: default, parameterized and BoxWt(BoxWt ob)

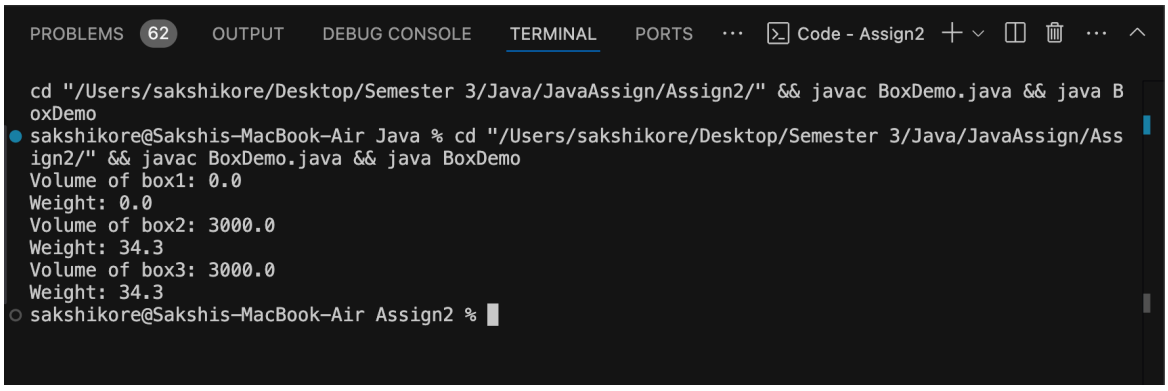
Use super() to invoke superclass constructors.

Code:

```
class Box {
    double width, height, depth;

    Box() {
        width = height = depth = 0;
    }

    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    Box(Box ob) {
        width = ob.width;
        height = ob.height;
        depth = ob.depth;
    }

    double volume() {
        return width * height * depth;
    }
}

class BoxWt extends Box {
    double weight;

    BoxWt() {
        super();
        weight = 0;
    }

    BoxWt(double w, double h, double d, double m) {
        super(w, h, d);
        weight = m;
    }
```

```
    BoxWt(BoxWt ob) {
        super(ob);
        weight = ob.weight;
    }

    void print_BoxWt() {
        System.out.println("Weight: " + weight);
    }
}

public class BoxDemo {
    public static void main(String[] args) {
        BoxWt box1 = new BoxWt();
        System.out.println("Volume of box1: " +
box1.volume());
        box1.print_BoxWt();

        BoxWt box2 = new BoxWt(10, 20, 15, 34.3);
        System.out.println("Volume of box2: " +
box2.volume());
        box2.print_BoxWt();

        BoxWt box3 = new BoxWt(box2);
        System.out.println("Volume of box3: " +
box3.volume());
        box3.print_BoxWt();
    }
}
```

Output:



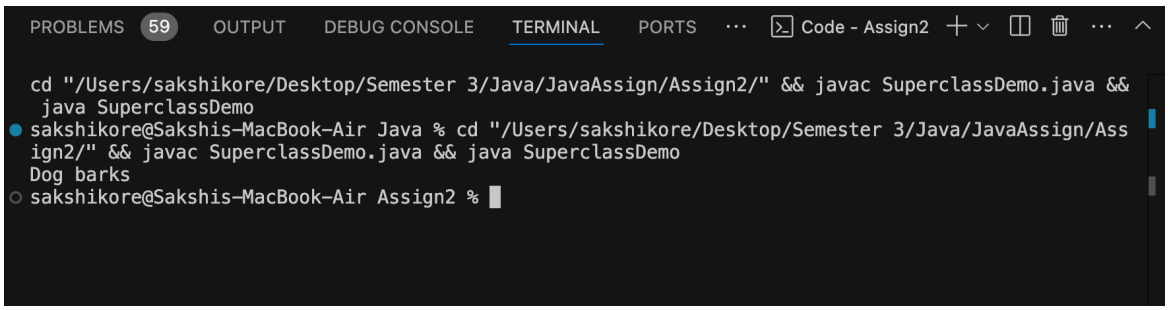
——————————————————XXX——————————————————

Question 10: WAP the demonstrate that super class variable can point to object of subclass.

Code:

```java
class Animal {
    void sound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}

public class SuperclassDemo {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        myDog.sound();
    }
}
```

Output:



————————————————————XXX————————————————————

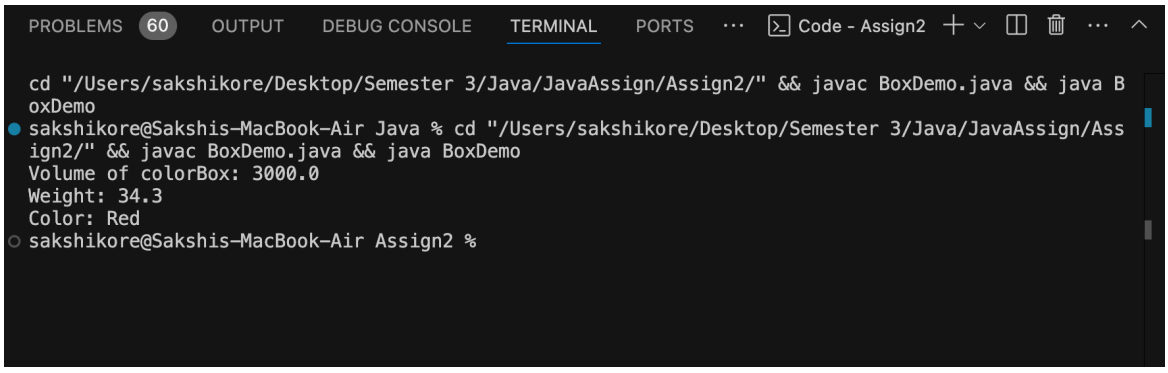
Question 11: WAP to demonstrate multilevel inheritance by creating a BoxColor class and inheriting BoxWt class. BoxColor class has an instance variable color of String type.

Code:

```
class Box {
    double width, height, depth;

    Box() {
        width = height = depth = 0;
    }

    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    Box(Box ob) {
        width = ob.width;
        height = ob.height;
        depth = ob.depth;
    }

    double volume() {
        return width * height * depth;
    }
}

class BoxWt extends Box {
    double weight;

    BoxWt() {
        super();
        weight = 0;
    }

    BoxWt(double w, double h, double d, double m) {
        super(w, h, d);
        weight = m;
    }

    BoxWt(BoxWt ob) {
        super(ob);
        weight = ob.weight;
    }
```

```
    void print_BoxWt() {
        System.out.println("Weight: " + weight);
    }
}

class BoxColor extends BoxWt {
    String color;

    BoxColor(double w, double h, double d, double m, String
c) {
        super(w, h, d, m);
        this.color = c;
    }

    void displayColor() {
        System.out.println("Color: " + color);
    }
}

public class BoxDemo {
    public static void main(String[] args) {
        BoxColor colorBox = new BoxColor(10, 20, 15, 34.3,
"Red");
        System.out.println("Volume of colorBox: " +
colorBox.volume());
        colorBox.print_BoxWt();
        colorBox.displayColor();
    }
}
```

Output:



————————————————————XXX————————————————————

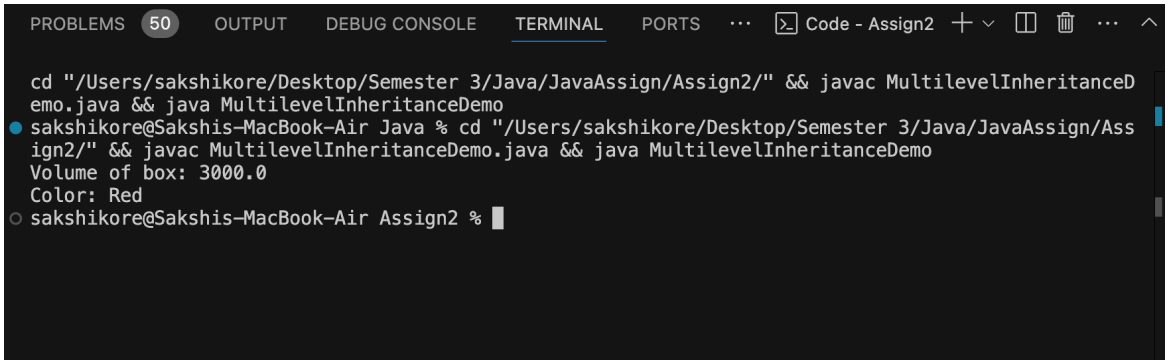
Question 12: WAP to demonstrate the sequence of execution of constructors in multilevel inheritance.

Code:

```
class Box
{
    double width, height, depth;

    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    double volume() {
        return width * height * depth;
    }
}

class BoxWt extends Box {
    double weight;

    BoxWt(double w, double h, double d, double m) {
        super(w, h, d);
        weight = m;
    }
}

class BoxColor extends BoxWt {
    String color;

    BoxColor(double w, double h, double d, double m, String
c) {
        super(w, h, d, m);
        color = c;
    }

    void displayColor() {
        System.out.println("Color: " + color);
    }
}

public class MultilevelInheritanceDemo {
    public static void main(String[] args) {
        BoxColor box = new BoxColor(10, 20, 15, 34.3,
"Red");
```

```
        System.out.println("Volume of box: " +
box.volume());
        box.displayColor();
    }
}
```

Output:



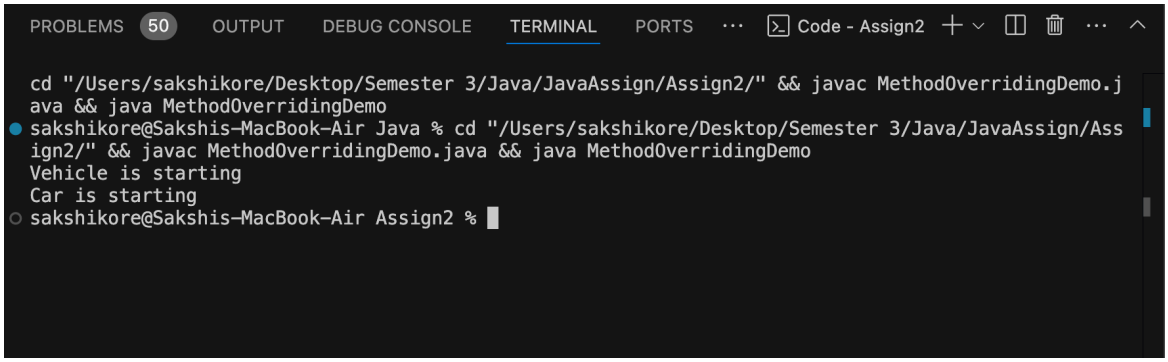
————————————————————XXX————————————————————

Question 13: WAP to demonstrate the concept of method overriding.

Code:

```java
class Vehicle {
    void start() {
        System.out.println("Vehicle is starting");
    }
}

class Car extends Vehicle {
    @Override
    void start() {
        System.out.println("Car is starting");
    }
}

public class MethodOverridingDemo {
    public static void main(String[] args) {
        Vehicle myVehicle = new Vehicle();
        Vehicle myCar = new Car();

        myVehicle.start();
        myCar.start();
    }
}
```

Output:



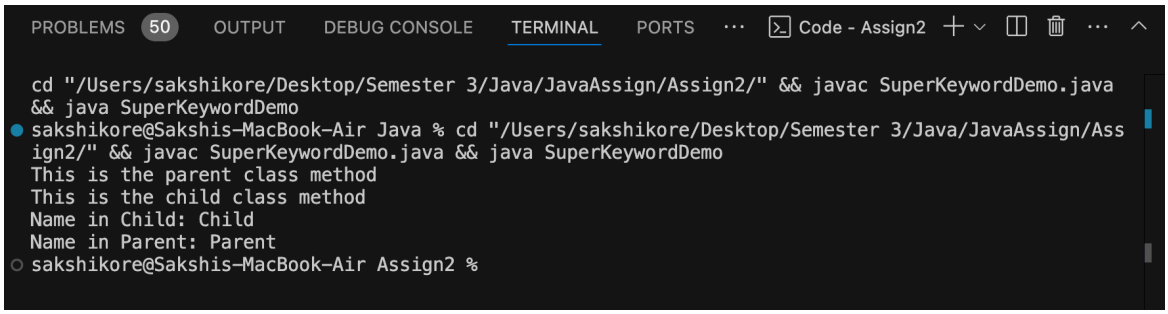
——————————————————XXX——————————————————

Question 14: WAP to demonstrate the use of super keyword to call overridden methods and instance variables.

Code:

```
class Parent {
    String name = "Parent";

    void show() {
        System.out.println("This is the parent class 
method");
    }
}
class Child extends Parent {
    String name = "Child";

    @Override
    void show() {
        super.show();
        System.out.println("This is the child class 
method");
    }

    void displayNames() {
        System.out.println("Name in Child: " + name);
        System.out.println("Name in Parent: " + super.name);
    }
}

public class SuperKeywordDemo {
    public static void main(String[] args) {
        Child child = new Child();
        child.show();
        child.displayNames();
    }
}
```

Output:



————————————————————XXX————————————————————

Question 15: Create a class Animal with

- instance variables:
  - boolean vegetarian
  - String food
  - int numOfLegs
- Create a no-argument constructor and parameterized constructor. (Use “this”)
- Create getters and setters
- Create a toString() method for animal class
- Create a subclass Cat with instance variable:
  - String color
  - Create a no-argument constructor and parameterized constructor which has all four parameters (Use this and super)
  - Create a toString() method for Cat class
- Create a subclass Cow with instance variable:
  - String breed
  - Create a no-argument constructor and parameterized constructor which has all four parameters. (Use this and super)
  - Create a toString() method for Cow class

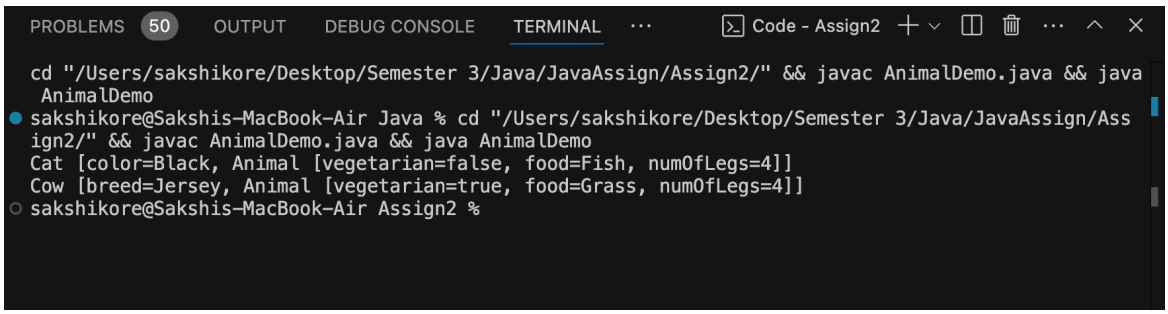
Code:

```
class Animal {
    private boolean vegetarian;
    private String food;
    private int numOfLegs;

    public Animal() {
        this.vegetarian = false;
        this.food = "unknown";
        this.numOfLegs = 0;
    }

    public Animal(boolean vegetarian, String food, int
numOfLegs) {
        this.vegetarian = vegetarian;
        this.food = food;
        this.numOfLegs = numOfLegs;
    }

    public boolean isVegetarian() {
```

```
        return vegetarian;
    }
    public void setVegetarian(boolean vegetarian) {
        this.vegetarian = vegetarian;
    }
    public String getFood() {
        return food;
    }
    public void setFood(String food) {
        this.food = food;
    }
    public int getNumOfLegs() {
        return numOfLegs;
    }
    public void setNumOfLegs(int numOfLegs) {
        this.numOfLegs = numOfLegs;
    }

    @Override
    public String toString() {
        return "Animal [vegetarian=" + vegetarian + ",
food=" + food + ", numOfLegs=" + numOfLegs + "]";
    }
}

class Cat extends Animal {
    private String color;
    public Cat() {
        super();
        this.color = "unknown";
    }
    public Cat(boolean vegetarian, String food, int
numOfLegs, String color) {
        super(vegetarian, food, numOfLegs);
        this.color = color;
    }
    public String getColor() {
        return color;
    }
    public void setColor(String color) {
        this.color = color;
    }

    @Override
    public String toString() {
        return "Cat [color=" + color + ", " +
super.toString() + "]";
    }
}
```

```java
class Cow extends Animal {
    private String breed;

    public Cow() {
        super();
        this.breed = "unknown";
    }
    public Cow(boolean vegetarian, String food, int
numOfLegs, String breed) {
        super(vegetarian, food, numOfLegs);
        this.breed = breed;
    }
    public String getBreed() {
        return breed;
    }
    public void setBreed(String breed) {
        this.breed = breed;
    }

    @Override
    public String toString() {
        return "Cow [breed=" + breed + ", " +
super.toString() + "]";
    }
}

public class AnimalDemo {
    public static void main(String[] args) {
        Cat cat = new Cat(false, "Fish", 4, "Black");
        System.out.println(cat);
        Cow cow = new Cow(true, "Grass", 4, "Jersey");
        System.out.println(cow);
    }
}
```

Output:



————————————————————XXX————————————————————

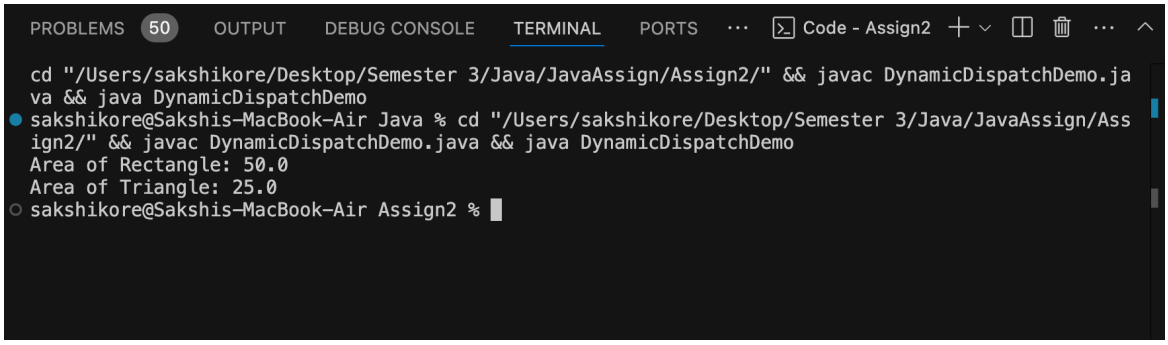
Question 16: Create a figure class with instance variable dim1, dim2 and method as area(). Create two derived class rectangle and triangle with constructors defined for initializing instance variable. Override area in both derived classes. WAP to demonstrate dynamic method dispatch (Run time polymorphism).

Code:

```
class Figure {
    double dim1, dim2;

    Figure(double dim1, double dim2) {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }

    double area() {
        return 0;
    }
}

class Rectangle extends Figure {
    Rectangle(double length, double breadth) {
        super(length, breadth);
    }

    @Override
    double area() {
        return dim1 * dim2;
    }
}

class Triangle extends Figure {
    Triangle(double base, double height) {
        super(base, height);
    }

    @Override
    double area() {
        return 0.5 * dim1 * dim2;
    }
}

public class DynamicDispatchDemo {
    public static void main(String[] args) {
        Figure figure;
```

```
        figure = new Rectangle(10, 5);
        System.out.println("Area of Rectangle: " +
figure.area());

        figure = new Triangle(10, 5);
        System.out.println("Area of Triangle: " +
figure.area());
    }
}
```

Output:



——————————————————————XXX——————————————————————

Question 17: Write a program to implement the concept of multiple inheritance through interface. Create a Figure and Draw interface. In Figure interface, define members PI, area(), perimeter(). In Draw interface, define members draw_shape(). Implement it in Circle class and Rectangle class.

Code:

```
interface Figure {
    double PI = 3.14159;

    double area();
    double perimeter();
}

interface Draw {
    void drawShape();
}

class Circle implements Figure, Draw {
    private double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    public double area() {
        return PI * radius * radius;
    }

    public double perimeter() {
        return 2 * PI * radius;
    }

    public void drawShape() {
        System.out.println("Drawing a Circle.");
    }
}

class Rectangle implements Figure, Draw {
    private double length, breadth;

    Rectangle(double length, double breadth) {
        this.length = length;
        this.breadth = breadth;
    }
```

```
    public double area() {
        return length * breadth;
    }

    public double perimeter() {
        return 2 * (length + breadth);
    }

    public void drawShape() {
        System.out.println("Drawing a Rectangle.");
    }
}

public class MultipleInheritanceDemo {
    public static void main(String[] args) {
        Circle circle = new Circle(5);
        circle.drawShape();
        System.out.println("Area of Circle: " +
circle.area());
        System.out.println("Perimeter of Circle: " +
circle.perimeter());

        Rectangle rectangle = new Rectangle(10, 5);
        rectangle.drawShape();
        System.out.println("Area of Rectangle: " +
rectangle.area());
        System.out.println("Perimeter of Rectangle: " +
rectangle.perimeter());
    }
}
```

Output:

```
PROBLEMS 50  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  ···  Code - Assign2

cd "/Users/sakshikore/Desktop/Semester 3/Java/JavaAssign/Assign2/" && javac MultipleInheritanceDem
o.java && java MultipleInheritanceDemo
sakshikore@Sakshis-MacBook-Air Java % cd "/Users/sakshikore/Desktop/Semester 3/Java/JavaAssign/Ass
ign2/" && javac MultipleInheritanceDemo.java && java MultipleInheritanceDemo
Drawing a Circle.
Area of Circle: 78.53975
Perimeter of Circle: 31.4159
Drawing a Rectangle.
Area of Rectangle: 50.0
Perimeter of Rectangle: 30.0
sakshikore@Sakshis-MacBook-Air Assign2 %
```

——————————————————XXX——————————————————

Question 18: Write a Java program that creates a class hierarchy for employees of a company. The base class should be Employee, with subclasses Manager and Developer.

- Employee class should have private attributes such as name, address, salary, and job title.
  - Implement public methods:
  - public double calculateBonus(): return 0.0
  - public String generatePerformanceReport(): return "No performance report available.”
  - Create getters
  - Create no arg and parameterized constructors
- Manager class has an attribute numberOfSubordinates. It has a method: mangeProject(), and overridden methods: calculateBonus() and generatePerformanceReport().
  - public double calculateBonus(): provides a custom implementation for bonus calculation for managers. In this case, it calculates the bonus as 15% of the manager's salary. (Hint: Use getter for salary)
  - public String generatePerformanceReport(): It returns a specific performance report message for managers, including the manager's name and an “Excellent" rating. (Hint: Use getter for name)
  - public void manageProject(): This is a custom method specific to the “Manager" class. It simulates the action of a manager managing a project by printing a message to the console. (Hint: Use getter for name)
  - Create no arg and parameterised constructors and getters
- Developer class has a private attribute programmingLanguage• public double calculateBonus(): provides a custom implementation for bonus calculation for developers. In this case, it calculates the bonus as 10% of the developer's salary. (Hint: Use getter for salary)
  - public String generatePerformanceReport(): It returns a specific performance report message for developers, including the developer's name and a "Good" rating. (Hint: Use getter for name)
  - public void writeCode(): This is a custom method specific to the "Developer" class. It simulates the action of a developer writing code in their specialized programming language by printing a message to the console. (Hint: Use getter for name)
  - Create no arg and parameterized constructors and getters

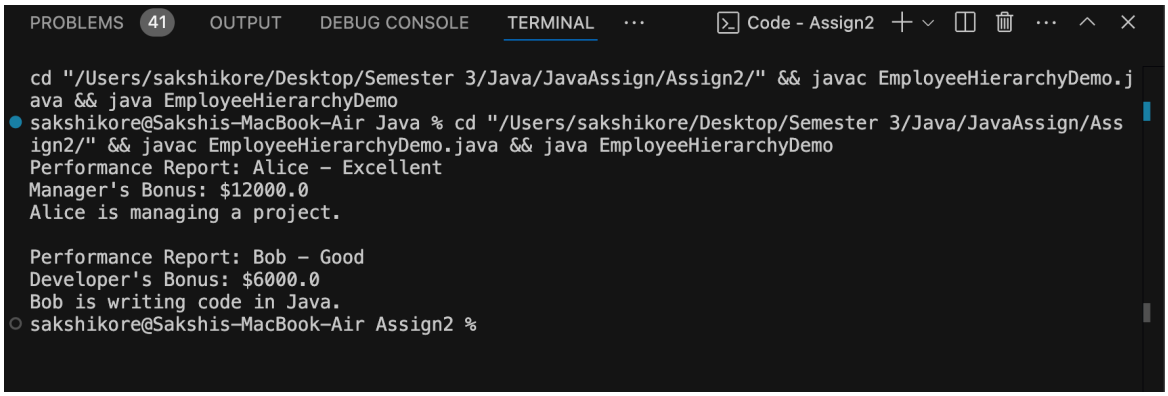
Code:

```
class Employee {
    private String name;
    private String address;
    private double salary;
    private String jobTitle;

    public Employee() {
        this.name = "Unknown";
        this.address = "Unknown";
        this.salary = 0.0;
        this.jobTitle = "Unknown";
    }

    public Employee(String name, String address, double
salary, String jobTitle) {
        this.name = name;
        this.address = address;
        this.salary = salary;
        this.jobTitle = jobTitle;
    }

    public double calculateBonus() {
        return 0.0;
    }
    public String generatePerformanceReport() {
        return "No performance report available.";
    }
    public String getName() {
        return name;
    }
    public String getAddress() {
        return address;
    }
    public double getSalary() {
        return salary;
    }
    public String getJobTitle() {
        return jobTitle;
    }
}

class Manager extends Employee {
    private int numberOfSubordinates;
    public Manager() {
        super();
        this.numberOfSubordinates = 0;
    }
```

```
    public Manager(String name, String address, double
salary, String jobTitle, int numberOfSubordinates) {
        super(name, address, salary, jobTitle);
        this.numberOfSubordinates = numberOfSubordinates;
    }

    @Override
    public double calculateBonus() {
        return getSalary() * 0.15;
    }
    @Override
    public String generatePerformanceReport() {
        return "Performance Report: " + getName() + " -
Excellent";
    }

    public void manageProject() {
        System.out.println(getName() + " is managing a
project.");
    }
    public int getNumberOfSubordinates() {
        return numberOfSubordinates;
    }
}

class Developer extends Employee {
    private String programmingLanguage;
    public Developer() {
        super();
        this.programmingLanguage = "Unknown";
    }
    public Developer(String name, String address, double
salary, String jobTitle, String programmingLanguage) {
        super(name, address, salary, jobTitle);
        this.programmingLanguage = programmingLanguage;
    }

    @Override
    public double calculateBonus() {
        return getSalary() * 0.10; // 10% of the salary
    }
    @Override
    public String generatePerformanceReport() {
        return "Performance Report: " + getName() + " -
Good";
    }
```

```
    public void writeCode()
{        System.out.println(getName() + " is writing code in
" + programmingLanguage + ".");
    }
    public String getProgrammingLanguage() {
        return programmingLanguage;
    }
}

public class EmployeeHierarchyDemo {
    public static void main(String[] args) {
        Manager manager = new Manager("Alice", "123 Main
St", 80000, "Manager", 5);
    System.out.println(manager.generatePerformanceReport());
        System.out.println("Manager's Bonus: $" +
manager.calculateBonus());
        manager.manageProject();
        System.out.println();
        Developer developer = new Developer("Bob", "456
Maple Ave", 60000, "Developer", "Java");
  System.out.println(developer.generatePerformanceReport());
        System.out.println("Developer's Bonus: $" +
developer.calculateBonus());
        developer.writeCode();
    }
}
```

Output:



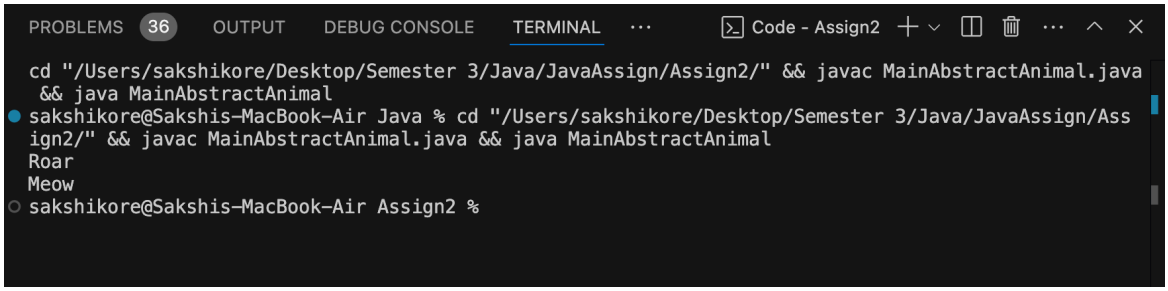
————————————————————XXX————————————————————

Question 19: Write a Java program to create an abstract class Animal with an abstract method called sound().

Code:

```
abstract class Animal {
    abstract void sound();
}

class Lion extends Animal {
    @Override
    void sound() {
        System.out.println("Roar");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Meow");
    }
}

public class MainAbstractAnimal {
    public static void main(String[] args) {
        Animal lion = new Lion();
        Animal cat = new Cat();

        lion.sound();
        cat.sound();
    }
}
```

Output:



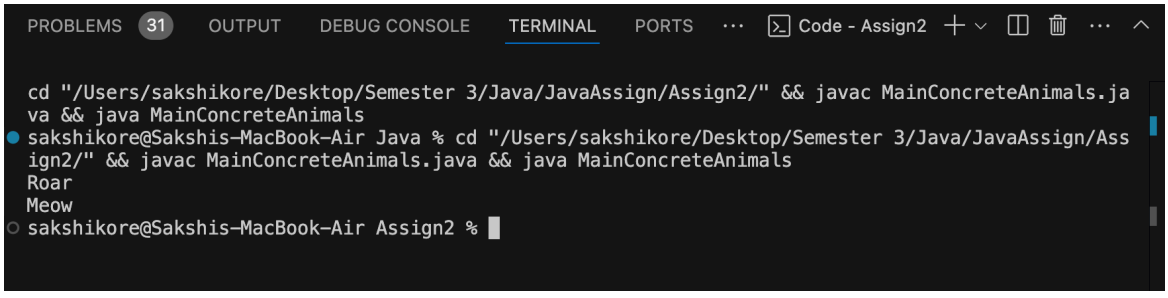
————————————————————XXX————————————————————

Question 20: Create subclasses Lion and Cat that extend the Animal class and implement the sound() method to make a specific sound for each animal.

Code:

```
abstract class Animal {
    abstract void sound();
}

class Lion extends Animal {
    @Override
    void sound() {
        System.out.println("Roar");
    }
}

class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Meow");
    }
}

public class MainConcreteAnimals {
    public static void main(String[] args) {
        Animal lion = new Lion();
        Animal cat = new Cat();

        lion.sound();
        cat.sound();
    }
}
```

Output:



————————————————————XXX————————————————————

Question 21: Create an Abstract Class “Shape” and 2 subclasses, SemiCircle and Circle that extend the Shape class and implement the respective methods to calculate the area and perimeter of each shape.

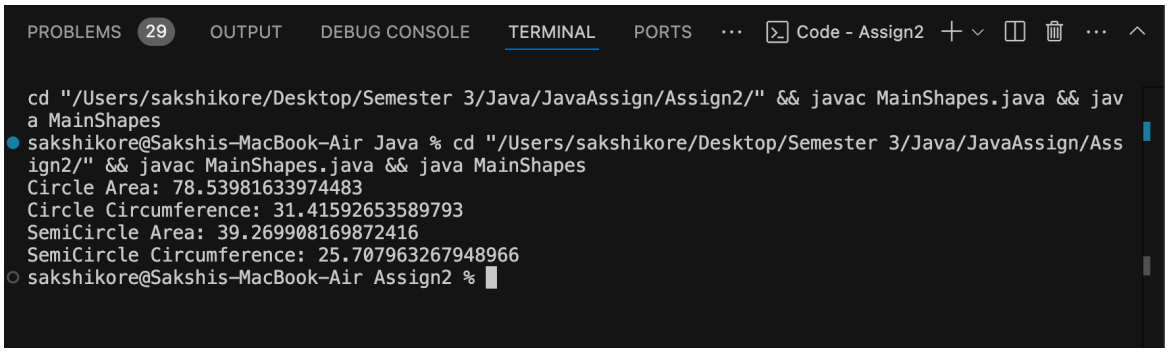
- Shape also defines 2 abstract methods calculateArea() and calculateCircumference(). Both return doubles representing the area and circumference of the shapes respectively.
- For each shape you will have to create the necessary values need to calculate area and circumference and getters and setters for each of the values. You will also need to create constructors for each class, and the instructions for those will be included with each class.

Code:

```
abstract class Shape {
    abstract double calculateArea();

    abstract double calculateCircumference();
}
class SemiCircle extends Shape {
    private double radius;
    SemiCircle(double radius) {
        this.radius = radius;
    }

    @Override
    double calculateArea() {
        return Math.PI * radius * radius / 2;
    }
    @Override
    double calculateCircumference() {
        return Math.PI * radius + 2 * radius;
    }

    public double getRadius() {
        return radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }
}

class Circle extends Shape {
    private double radius;
    Circle(double radius) {
```

```
        this.radius = radius;
    }

    @Override
    double calculateArea() {
        return Math.PI * radius * radius;
    }
    @Override
    double calculateCircumference() {
        return 2 * Math.PI * radius;
    }

    public double getRadius() {
        return radius;
    }
    public void setRadius(double radius) {
        this.radius = radius;
    }
}

public class MainShapes {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape semiCircle = new SemiCircle(5);
        System.out.println("Circle Area: " +
circle.calculateArea());
        System.out.println("Circle Circumference: " +
circle.calculateCircumference());
        System.out.println("SemiCircle Area: " +
semiCircle.calculateArea());
        System.out.println("SemiCircle Circumference: " +
semiCircle.calculateCircumference());
    }
}
```

Output:



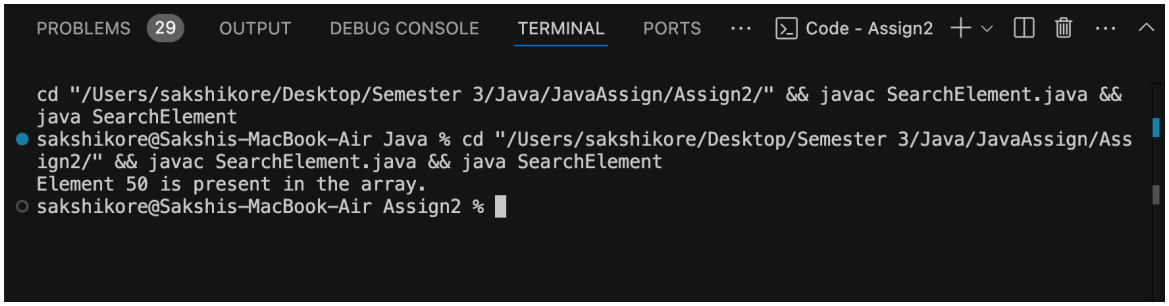
————————————————————XXX————————————————————

Question 22: Write a program to search an element in an array using for each loop.

Code:

```
public class SearchElement {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50, 60, 70, 80, 90,
100};

        int searchElement = 50;

        boolean found = false;

        for (int num : numbers) {
            if (num == searchElement) {
                found = true;
                break;
            }
        }

        if (found) {
            System.out.println("Element " + searchElement +
" is present in the array.");
        } else {
            System.out.println("Element " + searchElement +
" is not present in the array.");
        }
    }
}
```

Output:



————————————————————XXX————————————————————

Question 23: WAP to calculate sum and average of elements stored in one D array.

Code:

```
public class ArraySumAverage
{
    public static void main(String[] args)
    {
        int[] numbers = {10, 20, 30, 40, 50};

        int sum = 0;
        double average;

        for (int num : numbers)
        {
            sum += num;
        }

        average = (double) sum / numbers.length;

        System.out.println("Sum of elements: " + sum);
        System.out.println("Average of elements: " +
average);
    }
}
```

Output:

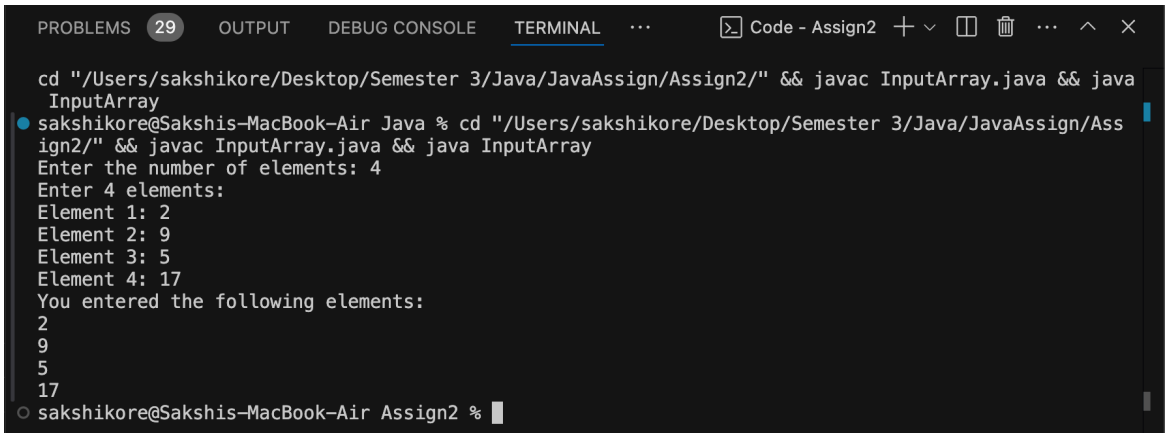
————————————————————XXX————————————————————

Question 24: WAP to input elements in one D array using scanner class.

Code:

```java
import java.util.Scanner;

public class InputArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of elements: ");
        int size = scanner.nextInt();

        int[] numbers = new int[size];

        System.out.println("Enter " + size + " elements:");
        for (int i = 0; i < size; i++) {
            System.out.print("Element " + (i + 1) + ": ");
            numbers[i] = scanner.nextInt();
        }

        scanner.close();

        System.out.println("You entered the following
elements:");
        for (int num : numbers) {
            System.out.println(num);
        }
    }
}
```

Output:



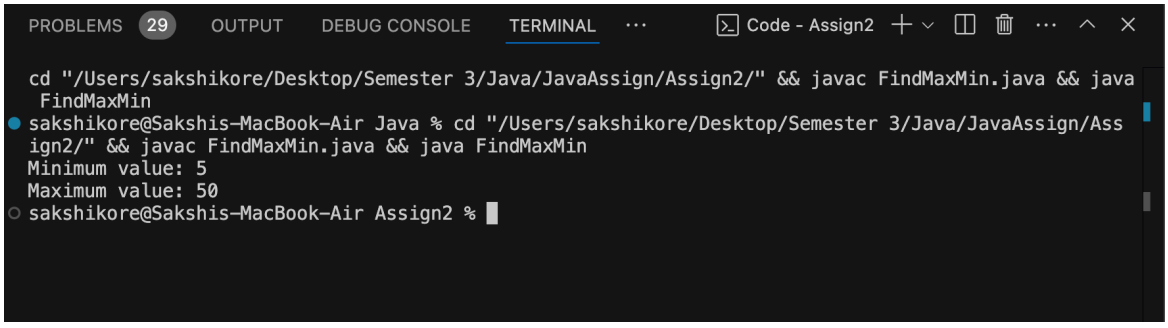
————————————————————XXX————————————————————

Question 25: Write a Java program to find the maximum and minimum value of an array.

Code:

```java
public class FindMaxMin
{
    public static void main(String[] args)
    {
        int[] numbers = {10, 20, 5, 40, 50, 15, 30};

        int min = numbers[0];
        int max = numbers[0];

        for (int num : numbers)
        {
            if (num < min)
            {
                min = num;
            }
            if (num > max)
            {
                max = num;
            }
        }

        System.out.println("Minimum value: " + min);
        System.out.println("Maximum value: " + max);
    }
}
```

Output:



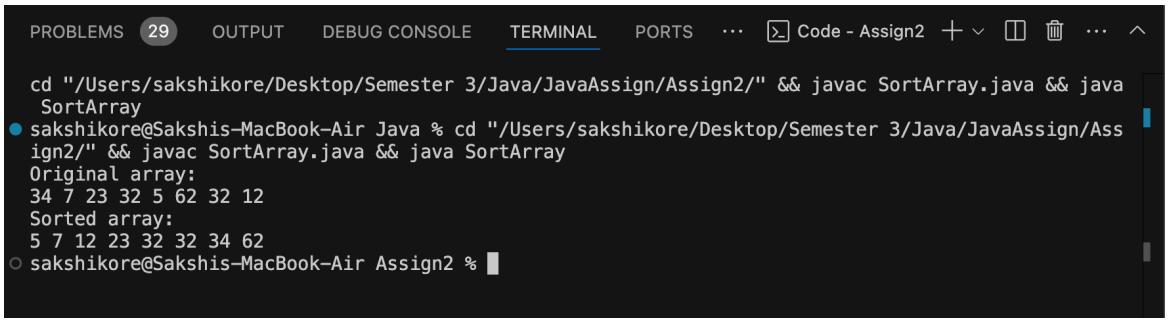
————————————————————XXX————————————————————

Question 26: Write a Java program to sort an array of integers.

Code:

```java
import java.util.Arrays;

public class SortArray
{
    public static void main(String[] args)
    {
        int[] numbers = {34, 7, 23, 32, 5, 62, 32, 12};

        System.out.println("Original array:");
        printArray(numbers);

        Arrays.sort(numbers);

        System.out.println("Sorted array:");
        printArray(numbers);
    }

    public static void printArray(int[] array)
    {
        for (int num : array)
        {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

Output:



————————————————————XXX————————————————————