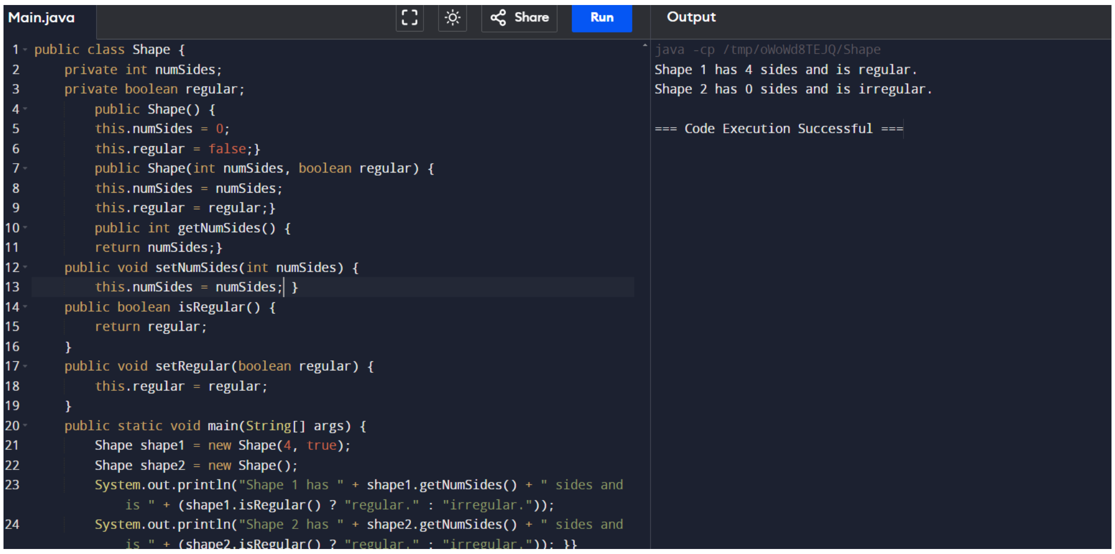
Java Fundamentals 7-1: Classes, Objects, and Methods Practice Activities

Vocabulary Definitions

1. **Class**: A template used for making Java objects.
2. **this**: An optional keyword used to access the members and methods of a class.
3. **Object**: An instance of a class.
4. **new**: The operator used to create an instance of a class.
5. **Garbage Collection**: A built-in function of the Java VM that frees memory as objects are no longer needed or referenced.
6. **Mutator Method**: A method that changes the state of an object.
7. **Accessor Method**: A method that returns information about an object back to the calling program.
8. **Method**: A procedure (changes the state of an object) or function (returns information about an object) that is encapsulated as part of a class.
9. **Instantiate**: A verb used to describe the act of creating a class object using the keyword `new`.
10. **Initialization**: The process of assigning a default value to a variable.
11. **Null**: An object reference that has not been instantiated.
12. **Finalizer**: An optional method that is called just before an object is removed by the garbage collector.
13. **Instance Variable**: The name of a variable that is associated with an object.
14. **Constructor**: A special method used to create an instance of a class.

Practice Activities

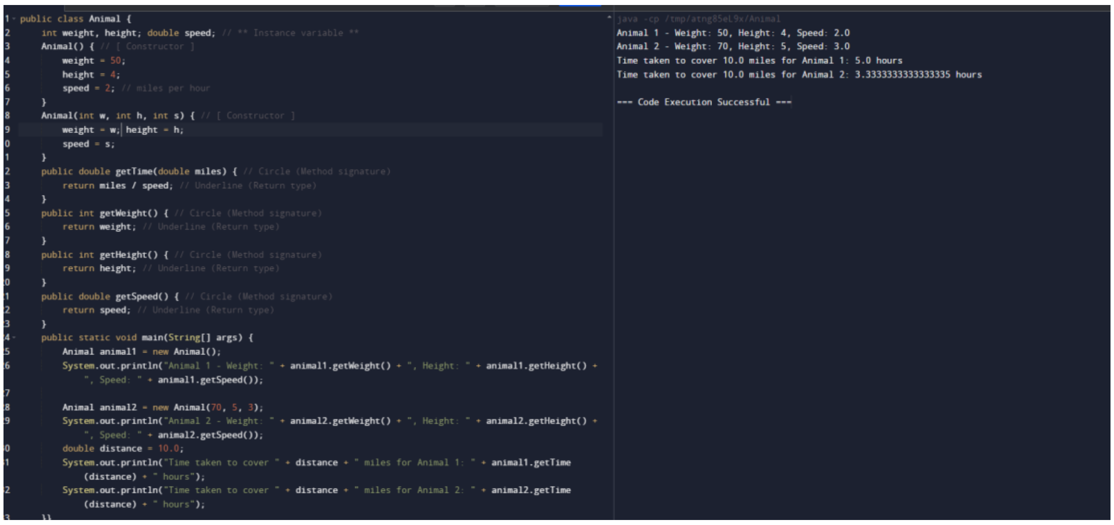
1. **Shape Class**



```java
public class Shape {
    private int numSides;
    private boolean regular;
        public Shape() {
        this.numSides = 0;
        this.regular = false;}
        public Shape(int numSides, boolean regular) {
        this.numSides = numSides;
        this.regular = regular;}
        public int getNumSides() {
        return numSides;}
    public void setNumSides(int numSides) {
        this.numSides = numSides; }
    public boolean isRegular() {
        return regular;
    }
    public void setRegular(boolean regular) {
        this.regular = regular;
    }
    public static void main(String[] args) {
        Shape shape1 = new Shape(4, true);
        Shape shape2 = new Shape();
        System.out.println("Shape 1 has " + shape1.getNumSides() + " sides and
            is " + (shape1.isRegular() ? "regular." : "irregular."));
        System.out.println("Shape 2 has " + shape2.getNumSides() + " sides and
            is " + (shape2.isRegular() ? "regular." : "irregular.")); }}
```

```
java -cp /tmp/oWoWd8TEJQ/Shape
Shape 1 has 4 sides and is regular.
Shape 2 has 0 sides and is irregular.

=== Code Execution Successful ===
```

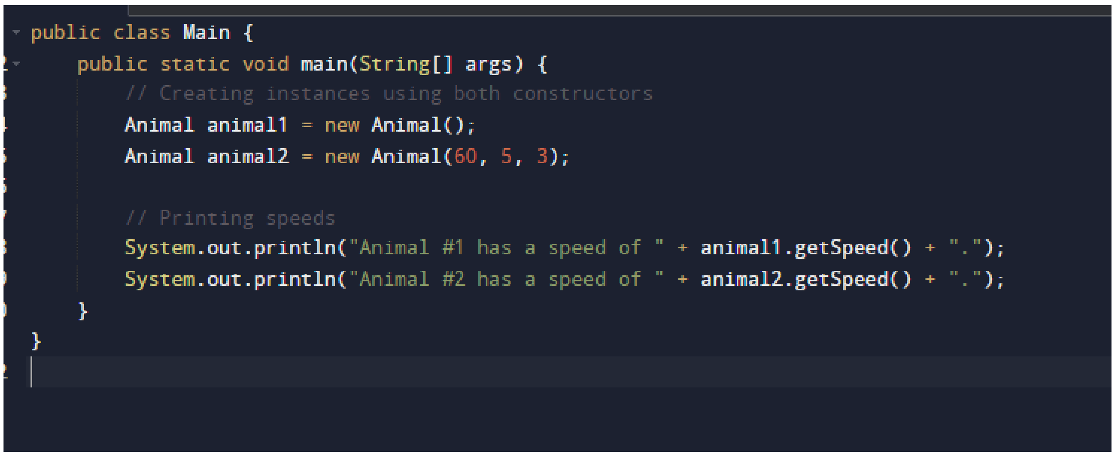
2. Identifying Key Parts of the Java Class



```java
public class Animal {
    int weight, height; double speed; // ** Instance variable **
    Animal() { // [ Constructor ]
        weight = 50;
        height = 4;
        speed = 2; // miles per hour
    }
    Animal(int w, int h, int s) { // [ Constructor ]
        weight = w; height = h;
        speed = s;
    }
    public double getTime(double miles) { // Circle (Method signature)
        return miles / speed; // Underline (Return type)
    }
    public int getWeight() { // Circle (Method signature)
        return weight; // Underline (Return type)
    }
    public int getHeight() { // Circle (Method signature)
        return height; // Underline (Return type)
    }
    public double getSpeed() { // Circle (Method signature)
        return speed; // Underline (Return type)
    }
    public static void main(String[] args) {
        Animal animal1 = new Animal();
        System.out.println("Animal 1 - Weight: " + animal1.getWeight() + ", Height: " + animal1.getHeight() +
            ", Speed: " + animal1.getSpeed());

        Animal animal2 = new Animal(70, 5, 3);
        System.out.println("Animal 2 - Weight: " + animal2.getWeight() + ", Height: " + animal2.getHeight() +
            ", Speed: " + animal2.getSpeed());
        double distance = 10.0;
        System.out.println("Time taken to cover " + distance + " miles for Animal 1: " + animal1.getTime
            (distance) + " hours");
        System.out.println("Time taken to cover " + distance + " miles for Animal 2: " + animal2.getTime
            (distance) + " hours");
    }}
```

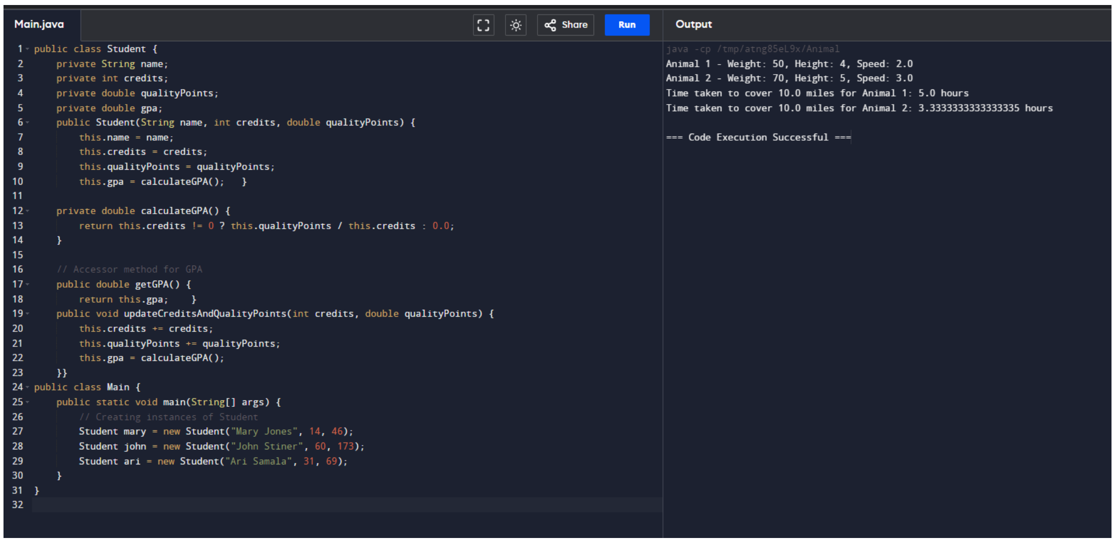
```
java -cp /tmp/atng85eL9x/Animal
Animal 1 - Weight: 50, Height: 4, Speed: 2.0
Animal 2 - Weight: 70, Height: 5, Speed: 3.0
Time taken to cover 10.0 miles for Animal 1: 5.0 hours
Time taken to cover 10.0 miles for Animal 2: 3.3333333333333335 hours

=== Code Execution Successful ===
```

3. Creating Instances of Animal



```java
public class Main {
    public static void main(String[] args) {
        // Creating instances using both constructors
        Animal animal1 = new Animal();
        Animal animal2 = new Animal(60, 5, 3);

        // Printing speeds
        System.out.println("Animal #1 has a speed of " + animal1.getSpeed() + ".");
        System.out.println("Animal #2 has a speed of " + animal2.getSpeed() + ".");
    }
}
```

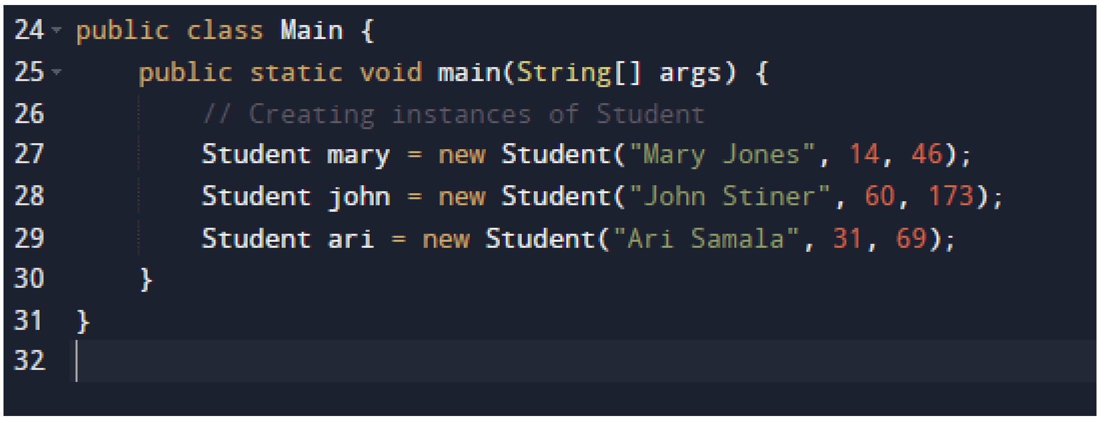
4. Student Class



```java
public class Student {
    private String name;
    private int credits;
    private double qualityPoints;
    private double gpa;
    public Student(String name, int credits, double qualityPoints) {
        this.name = name;
        this.credits = credits;
        this.qualityPoints = qualityPoints;
        this.gpa = calculateGPA();   }

    private double calculateGPA() {
        return this.credits != 0 ? this.qualityPoints / this.credits : 0.0;
    }

    // Accessor method for GPA
    public double getGPA() {
        return this.gpa;    }
    public void updateCreditsAndQualityPoints(int credits, double qualityPoints) {
        this.credits += credits;
        this.qualityPoints += qualityPoints;
        this.gpa = calculateGPA();
    }}
public class Main {
    public static void main(String[] args) {
        // Creating instances of Student
        Student mary = new Student("Mary Jones", 14, 46);
        Student john = new Student("John Stiner", 60, 173);
        Student ari = new Student("Ari Samala", 31, 69);
    }
}
```

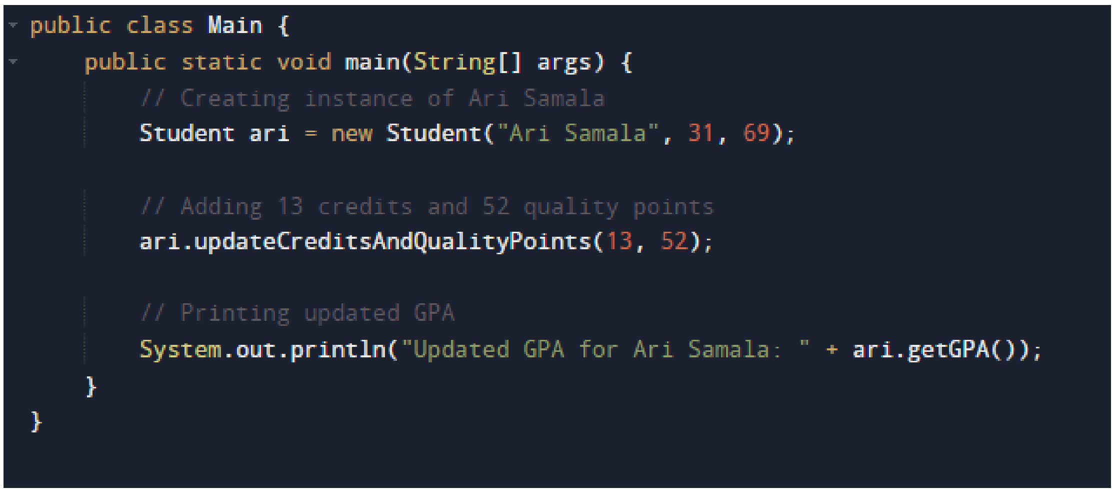
```
java -cp /tmp/atng85eL9x/Animal
Animal 1 - Weight: 50, Height: 4, Speed: 2.0
Animal 2 - Weight: 70, Height: 5, Speed: 3.0
Time taken to cover 10.0 miles for Animal 1: 5.0 hours
Time taken to cover 10.0 miles for Animal 2: 3.3333333333333335 hours

=== Code Execution Successful ===
```

5. Creating Instances of Student



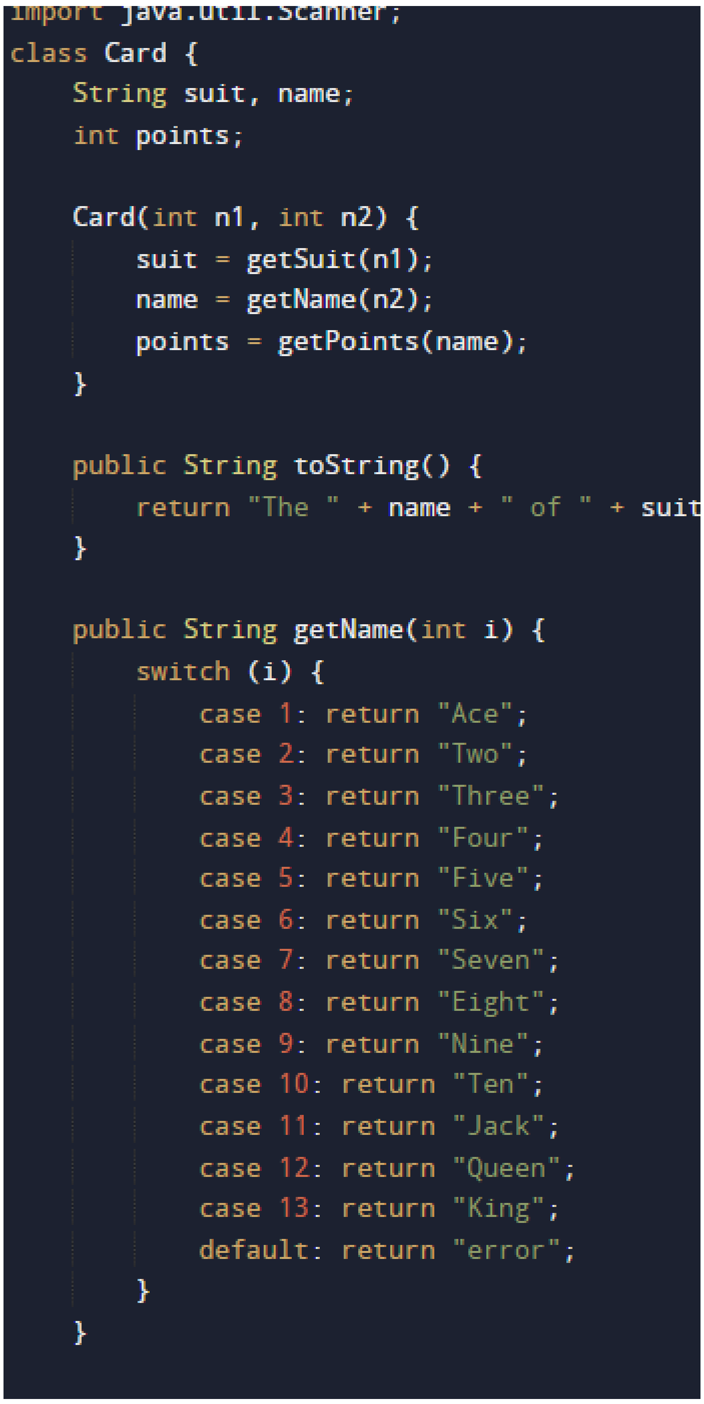
```java
public class Main {
    public static void main(String[] args) {
        // Creating instances of Student
        Student mary = new Student("Mary Jones", 14, 46);
        Student john = new Student("John Stiner", 60, 173);
        Student ari = new Student("Ari Samala", 31, 69);
    }
}
```

6. Updating Credits and Quality Points for Ari Samala



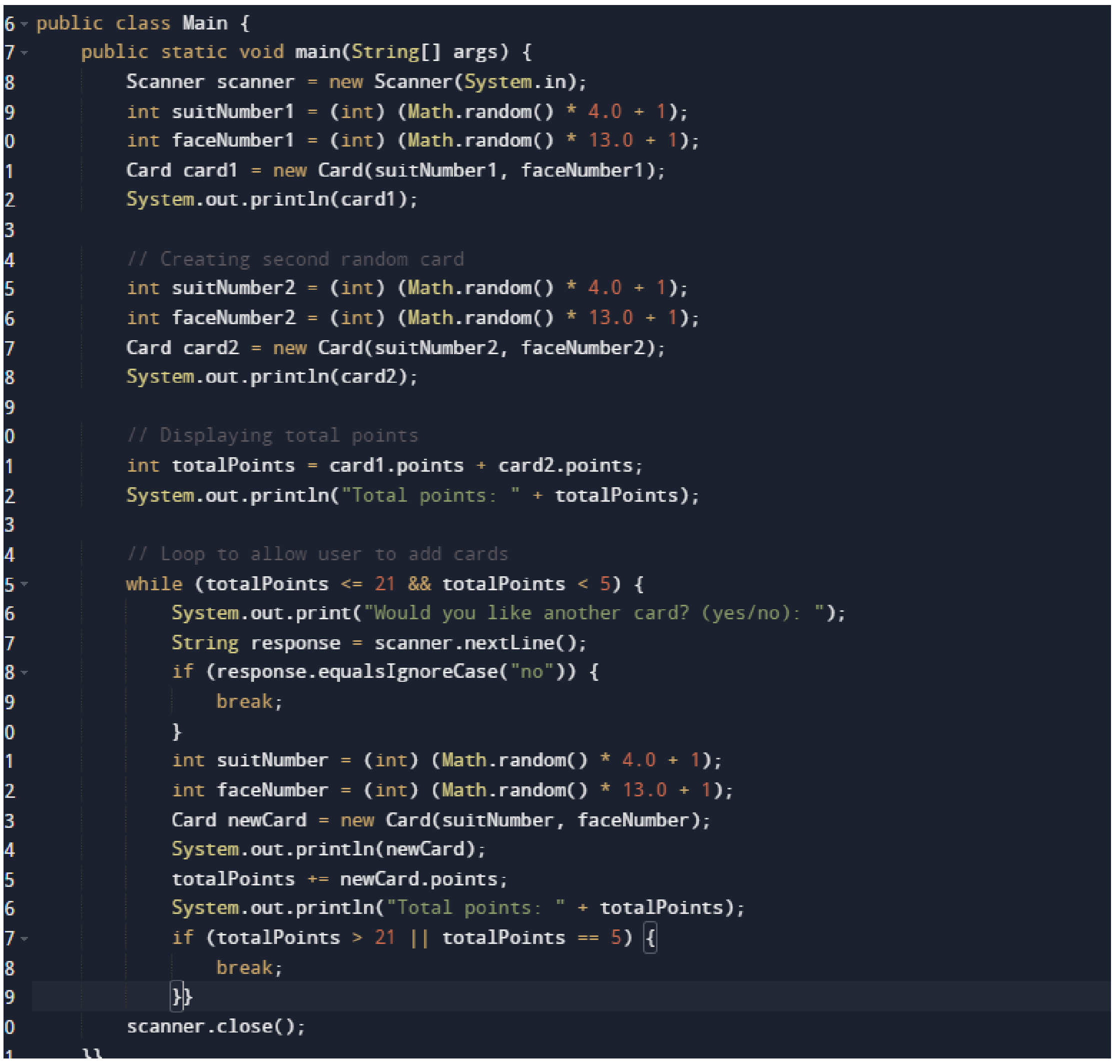
```java
public class Main {
    public static void main(String[] args) {
        // Creating instance of Ari Samala
        Student ari = new Student("Ari Samala", 31, 69);

        // Adding 13 credits and 52 quality points
        ari.updateCreditsAndQualityPoints(13, 52);

        // Printing updated GPA
        System.out.println("Updated GPA for Ari Samala: " + ari.getGPA());
    }
}
```

7. Card Class



```java
import java.util.Scanner;
class Card {
    String suit, name;
    int points;

    Card(int n1, int n2) {
        suit = getSuit(n1);
        name = getName(n2);
        points = getPoints(name);
    }

    public String toString() {
        return "The " + name + " of " + suit
    }

    public String getName(int i) {
        switch (i) {
            case 1: return "Ace";
            case 2: return "Two";
            case 3: return "Three";
            case 4: return "Four";
            case 5: return "Five";
            case 6: return "Six";
            case 7: return "Seven";
            case 8: return "Eight";
            case 9: return "Nine";
            case 10: return "Ten";
            case 11: return "Jack";
            case 12: return "Queen";
            case 13: return "King";
            default: return "error";
        }
    }
```

8. Main Class with Additional Card Logic



```java
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int suitNumber1 = (int) (Math.random() * 4.0 + 1);
        int faceNumber1 = (int) (Math.random() * 13.0 + 1);
        Card card1 = new Card(suitNumber1, faceNumber1);
        System.out.println(card1);

        // Creating second random card
        int suitNumber2 = (int) (Math.random() * 4.0 + 1);
        int faceNumber2 = (int) (Math.random() * 13.0 + 1);
        Card card2 = new Card(suitNumber2, faceNumber2);
        System.out.println(card2);

        // Displaying total points
        int totalPoints = card1.points + card2.points;
        System.out.println("Total points: " + totalPoints);

        // Loop to allow user to add cards
        while (totalPoints <= 21 && totalPoints < 5) {
            System.out.print("Would you like another card? (yes/no): ");
            String response = scanner.nextLine();
            if (response.equalsIgnoreCase("no")) {
                break;
            }
            int suitNumber = (int) (Math.random() * 4.0 + 1);
            int faceNumber = (int) (Math.random() * 13.0 + 1);
            Card newCard = new Card(suitNumber, faceNumber);
            System.out.println(newCard);
            totalPoints += newCard.points;
            System.out.println("Total points: " + totalPoints);
            if (totalPoints > 21 || totalPoints == 5) {
                break;
            }}
        scanner.close();
    }}
```