Week 5 Homework Q16

Telmen Enkhbold

San Fransico Bay University

CE480 - Java and Internet Application

Dr. Chang, Henry

10/12/2023

Author Note

The Question

A. class Square
    A. Attributes
        - int s ; // side
    B. Member functions
        - Helping function
            - int square(int i);

              Return the square of i

        - Manager functions
            - Constructor
        - Implementor
            - void enLarge(int ds);

              s is increased by ds

            - int area();
            - Call the method square() to compute area of a square

            - int perimeter();

              Return the perimeter of a square

        - Access functions
            - 1 get functions
            - 1 set functions
            - Predicate
                - isLarge();

                  A square is large if its side is greater than 10

B. class Circle
    - Attributes
        - double r; // radius
    - Member functions
        - Helping function
            - double pi();

              Return the value 3.1416

- Manager functions
  - Constructor
- Implementor
  1. void enLarge(double dr);

     r is increased by dr

  2. double area();

     Call the method pi() to compute area of a circle

  3. double circumference();

     Call the method pi() to compute the cicumference of a circle

- Access functions
  1. 1 get function
  2. 1 set function
  3. Predicate
     - isLarge();

       A circle is large if its radius is greater than 10

     - isAPoint();

       A circle is a point if r == 1;

C. class Coin
   - Attributes
     - Cicle circleObj;
     - Square squareObj;
   - Member functions
     - Helping function
       1. calcCircleArea();

          Return the area of the coin's circle portion

       2. calcSquareArea();

          Return the area of the coin's square portion

     - Manager functions

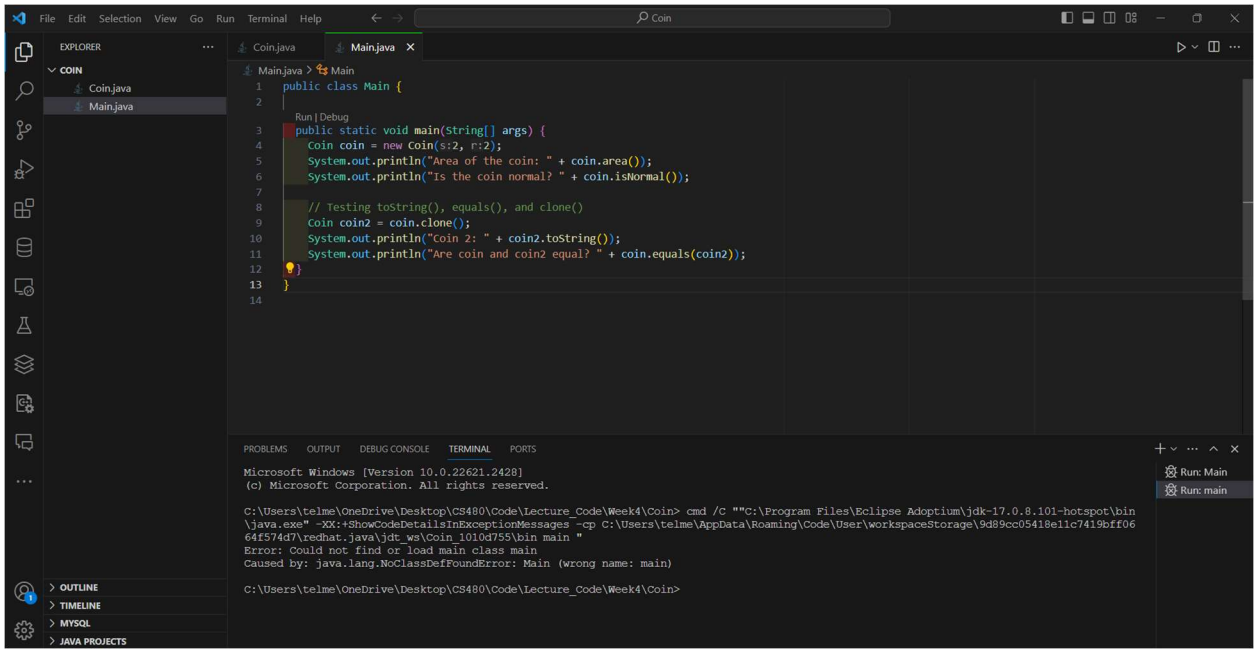
1. Two Constructors

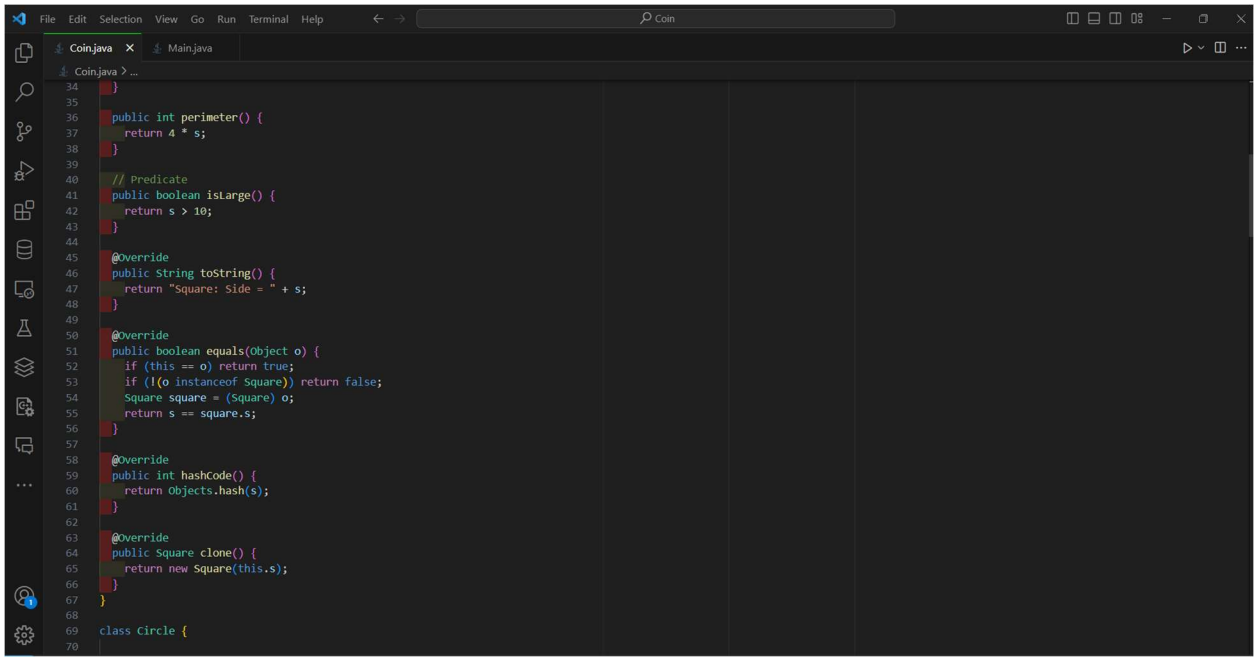
```
2. public Coin(int s1, double r1);
3. public Coin(Square squareObj1, Circle circleObj1);
```

- Implementor
  1. area(): Use the helping functions to compute the area of a coin which is equal to the substraction of the square's area from its circle's area.
- Access functions
  1. 2 get functions
  2. 2 set functions
  3. Predicate
     - isNormal();
     - `A coin is normal if its diameter is longer`
     - `than the diagnal of the square.`

D. Test your class by
   1. Create 1 object, coin
      - coin whose square's side is 2 and its circle's radius is 2
   2. Print the area of coin
   3. Check if coin is normal

d. Knowledge of four concepts are required in this question:
   - Aggregation
   - Summary of toString(), clone(), equals()
   - Clone (with and without object attributes)
   - toString()
   - equals()
   - References
     - 4 types of member functions
       - Geometry
     - Class structure
       - Class containing one layer of multiple simple objects attributes
     - Constructor Types
     - Summary of toString(), clone(), equals()





Source Code for Main

```
public class Main {
```

```
  public static void main(String[] args) {
    Coin coin = new Coin(2, 2);
    System.out.println("Area of the coin: " + coin.area());
    System.out.println("Is the coin normal? " + coin.isNormal());

    // Testing toString(), equals(), and clone()
    Coin coin2 = coin.clone();
    System.out.println("Coin 2: " + coin2.toString());
    System.out.println("Are coin and coin2 equal? " + coin.equals(coin2));
  }
}
```

Source Code for Coin

```
import java.util.Objects;

class Square {

  private int s; // side

  // Constructor
  public Square(int s) {
    this.s = s;
  }

  // Implementor
  public void enlarge(int ds) {
    s += ds;
  }

  // Access functions
  public int getSide() {
    return s;
  }

  public void setSide(int s) {
    this.s = s;
  }

  // Helper function
```

```
  public int square(int i) {
    return i * i;
  }

  // Manager functions
  public int area() {
    return square(s);
  }

  public int perimeter() {
    return 4 * s;
  }

  // Predicate
  public boolean isLarge() {
    return s > 10;
  }

  @Override
  public String toString() {
    return "Square: Side = " + s;
  }

  @Override
  public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Square)) return false;
    Square square = (Square) o;
    return s == square.s;
  }

  @Override
  public int hashCode() {
    return Objects.hash(s);
  }

  @Override
  public Square clone() {
    return new Square(this.s);
  }
}

class Circle {

  private double r; // radius
```

```
// Constructor
public Circle(double r) {
  this.r = r;
}

// Implementor
public void enlarge(double dr) {
  r += dr;
}

// Access functions
public double getRadius() {
  return r;
}

public void setRadius(double r) {
  this.r = r;
}

// Helper function
public double pi() {
  return 3.1416;
}

// Manager functions
public double area() {
  return pi() * square(r);
}

public double circumference() {
  return 2 * pi() * r;
}

// Predicates
public boolean isLarge() {
  return r > 10;
}

public boolean isAPoint() {
  return r == 1;
}

@Override
public String toString() {
```

```
    return "Circle: Radius = " + r;
  }

  @Override
  public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Circle)) return false;
    Circle circle = (Circle) o;
    return Double.compare(circle.r, r) == 0;
  }

  @Override
  public int hashCode() {
    return Objects.hash(r);
  }

  @Override
  public Circle clone() {
    return new Circle(this.r);
  }

  // Helper function
  private double square(double d) {
    return d * d;
  }
}

class Coin {

  private Circle circleObj;
  private Square squareObj;

  // Constructors
  public Coin(int s, double r) {
    squareObj = new Square(s);
    circleObj = new Circle(r);
  }

  public Coin(Square squareObj, Circle circleObj) {
    this.squareObj = squareObj;
    this.circleObj = circleObj;
  }

  // Implementor
  public double area() {
```

```
    return circleObj.area() - squareObj.area();
  }

  // Access functions
  public double getCircleArea() {
    return circleObj.area();
  }

  public double getSquareArea() {
    return squareObj.area();
  }

  public void setCircleRadius(double r) {
    circleObj.setRadius(r);
  }

  public void setSquareSide(int s) {
    squareObj.setSide(s);
  }

  // Predicates
  public boolean isNormal() {
    double coinDiameter = 2 * circleObj.getRadius();
    double squareDiagonal = squareObj.getSide() * Math.sqrt(2);
    return coinDiameter > squareDiagonal;
  }

  @Override
  public String toString() {
    return (
      "Coin: Square = " +
      squareObj.toString() +
      ", Circle = " +
      circleObj.toString()
    );
  }

  @Override
  public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Coin)) return false;
    Coin coin = (Coin) o;
    return (
      Objects.equals(circleObj, coin.circleObj) &&
      Objects.equals(squareObj, coin.squareObj)
```

```
    );
  }

  @Override
  public int hashCode() {
    return Objects.hash(circleObj, squareObj);
  }

  @Override
  public Coin clone() {
    return new Coin(this.squareObj.clone(), this.circleObj.clone());
  }
}
```