CS159 – Advanced Programming **James Madison University**
Sample Examination 2 **Fall 2024**

Name:_____

Instructions. Answer all of the following questions. This is a “closed book” examination and you must work entirely on your own. Your answers need not conform to the course style guide. All questions that involve code or are about code use the Java programming language.

This work complies with the JMU Honor Code.

Signature:_____

1. (8 points) Given the attached UML class diagram for the `Coin` class, write a statement that does each of the following. **You answer must not do more than what is asked for**.

1.1 Declares a variable named `inheritance` that is (a reference to) an array of `Coin` objects.

_____

1.2 Instantiates (i.e., allocates memory for) the array (referred to by) `inheritance` in such a way that it contains exactly 20 (references to) `Coin` objects.

_____

1.3 Instantiates a `Coin` object with a `weight` attribute of `10.0` and a `metal` attribute of `"Silver"`, and assigns (the reference to) it to the initial element of `week`.

_____

1.4 Calls the `getMetal()` method that belongs to the object referred to by the inital element of `inheritance` and prints the result on `System.out`.

_____

2. (8 points) Given the attached implementations of the `Size`, `EarRing`, `LipRing`, `NavelRing`, and `NoseRing` classes, what will be printed by each of the following main classes? Note: None of the classes contain any syntax errors.

```
// The main class for question 2.1
public class EarRingDriver {

    public static void main(String[] args) {
        Size d;
        d = new Size();
        d.radius = 5;
        d.thickness = 8;

        EarRing earRing;
        earRing = new EarRing(d);

        d.radius = 11;
        d.thickness = 14;

        System.out.println(earRing.toString());
    }
}
```

2.1 _____

```
// The main class for question 2.2
public class LipRingDriver {

    public static void main(String[] args) {
        Size d;
        d = new Size();
        d.radius = 5;
        d.thickness = 8;

        LipRing lipRing;
        lipRing = new LipRing(d);

        d.radius = 11;
        d.thickness = 14;

        System.out.println(lipRing.toString());
    }
}
```

2.2 _____

```
// The main class for question 2.3
public class NavelRingDriver {

    public static void main(String[] args) {
        Size d;
        d = new Size();
        d.radius = 5;
        d.thickness = 8;

        NavelRing navelRing;
        navelRing = new NavelRing(d);

        d.radius = 11;
        d.thickness = 14;

        System.out.println(navelRing.toString());
    }
}
```

2.3 _____

```
// The main class for question 2.4
public class NoseRingDriver {

    public static void main(String[] args) {
        Size d;
        d = new Size();
        d.radius = 5;
        d.thickness = 8;

        NoseRing noseRing;
        noseRing = new NoseRing(d);

        d.radius = 11;
        d.thickness = 14;

        System.out.println(noseRing.toString());
    }
}
```

2.4 _____

3. (5 Points) Given the attached `FractionTokenizer` class, what would be printed by the following fragment?

```
FractionTokenizer ft;
ft = new FractionTokenizer("2/4,3/4");

try {
    while (ft.hasNext()) {
        double answer;
        answer = ft.next();
        System.out.println(answer);
    }
} catch (NoSuchElementException nse) {
    System.out.println("That does not compute!");
}
```

4. (5 Points) Given the attached `FractionTokenizer` class, what would be printed by the following fragment?

```
FractionTokenizer ft;
ft = new FractionTokenizer("2/4,3/4");

for (int i=0; i<5; i++) {
    try {
        double answer;
        answer = ft.next();
        System.out.println(answer);
    } catch (NoSuchElementException nse) {
        System.out.println("That does not compute!");
    }
}
```

5. (8 Points) Complete `thefindSeniority()` method in the following `Seniority` enumerated type. Your answer must return the `Seniority` that is associated with the given number of credits if there is one. Otherwise it must return `null`. Hint: Every `enum` has static `values()` method that returns an array containing all of the instances of the `enum`.

```
public enum Seniority {

    FIRSTYEAR( 0, 27, 1.500),
    SOPHOMORE(28, 59, 1.750),
    JUNIOR(60, 89, 1.900),
    SENIOR(90, Integer.MAX_VALUE, 2.000);

    private double probation, suspension;
    private int minCredits; // The min. number of credits assoc. with this Seniority
    private int maxCredits; // The max. number of credits assoc. with this Seniority

    Seniority(int minCredits, int maxCredits, double suspension) {
        this.minCredits = minCredits;
        this.maxCredits = maxCredits;
        this.suspension = suspension;
        this.probation = 2.0;
    }

    public boolean contains(int credits) {
        return (credits >= minCredits) && (credits <= maxCredits);
    }

    public static Seniority findSeniority(int credits) {
        // Your answer goes here









    }

    public boolean isOnProbation(double gpa) {
        return !isOnSuspension(gpa) && (gpa < probation);
    }

    public boolean isOnSuspension(double gpa) {
        return gpa < suspension;
    }
}
```

6. (8 points) Given the `Seniority` enumerated type in the previous question, complete the following JUnit test. Your solution must cover all statements and all branches in the `contains()` method.

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

public class SeniorityTest {









}
```

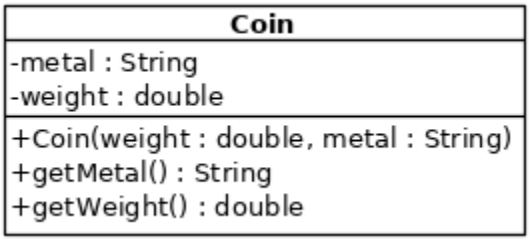
7. (4 points) Write a complete implementation of the explicit value for the `Coin` class in the attached UML diagram. Your implementation must construct a **deep copy**.

```
}
```

8. (4 points) Write a complete implementation of the a copy constructor for the `Coin` class. Your implementation must invoke the explicit value constructor.

```
}
```

Attachments




```
public class Size {

    public int thickness;
    public int radius;

    public Size() {
        thickness = 0;
        radius = 0;
    }
}
```

```
public class EarRing {

    private Size size;

    public EarRing(Size s) {
        this.size = new Size();
        this.size.radius = s.radius;
        this.size.thickness = s.thickness;
    }

    public String toString() {
        return String.format("%d, %d",
                this.size.radius, this.size.thickness);
    }
}
```

```
public class LipRing {

   private Size size;

    public LipRing(Size s) {
        this.size = new Size();
        this.size = s;
    }

    public String toString() {
        return String.format("%d, %d",
                this.size.radius, this.size.thickness);
    }
}
```

```
public class NavelRing {

    private Size size;

    public NavelRing(Size s) {
        this.size = s;
        this.size = new Size();
    }

    public String toString() {
        return String.format("%d, %d",
                this.size.radius, this.size.thickness);
    }
}
```

```
public class NoseRing {

    private Size size;

    public NoseRing(Size s) {
        this.size = s;
    }

    public String toString() {
        return String.format("%d, %d",
                this.size.radius, this.size.thickness);
    }
}
```

```
import java.util.NoSuchElementException;


public class FractionTokenizer {
    private static final String DEFAULT_DELIMITER = ",";

    private int next;
    private String[] tokens;

    public FractionTokenizer(String str) {
        if (str == null) {
            tokens = new String[0];
        } else {
            tokens = str.split(DEFAULT_DELIMITER);
        }
        next = 0;
    }

    public boolean hasNext() {
        return next < tokens.length;
    }

    public double next() throws NoSuchElementException {

        double denominator, numerator, result;
        String token;
        String[] parts;

        if (!hasNext()) {
            throw new NoSuchElementException();
        }

        token = tokens[next];
        next++;

        if ((token.length() < 3) || (token.indexOf('/') < 0)) {
            result = 0.0;
        } else {
            parts = token.split("/");
            numerator = Double.parseDouble(parts[0]);
            denominator = Double.parseDouble(parts[1]);

            result = numerator / denominator;
        }

        return result;
    }
}
```

JUnit Reference Card

This part of the reference card contains examples of JUnit tests that use static methods in the `org.junit.jupiter.api.Assertions` class.

An example of a JUnit test that uses the `assertEquals()` method:

```
@Test
public void getAtomicNumber() {
    Atom    o;
    o = new Atom("O", 8, 16);
    assertEquals(8, o.getAtomicNumber(), "Oxygen");
}
```

An example of a JUnit test that doesn't use a lambda expression to test for an exception:

```
@Test
public void constructor_IllegalArguments() {
    try {
      new Atom("O", -8, -16);
      fail("Should have thrown an IllegalArgumentException");
    } catch (IllegalArgumentException iae) {
      // The exception was thrown as expected
    }
}
```

An example of a JUnit test that uses a lambda expression to test for an exception:

```
@Test
public void constructor_IllegalArguments() {
    assertThrows(IllegalArgumentException.class, () ->
        {new Atom("O", -8, -16);});
}
```

This part of the reference card contains other useful static methods in the `org.junit.jupiter.api.Assertions` class.

```
assertFalse(boolean actual)
assertTrue(boolean actual)
assertNull(Object actual)
assertSame(Object expected, Object actual)
```