CS159 – Advanced Programming **James Madison University**
Sample Examination 3 **Fall 2024**

Name:_____

Instructions. Answer all of the following questions. This is a "closed book" exam and you must work entirely on your own. You may not use a computing or communications device of any kind. 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. (5 points) Given the attached `Question` and `ExtraCreditQuestion` classes and the following declarations:

```
Question q;
Question r;
Question s;
ExtraCreditQuestion e;
ExtraCreditQuestion f;
```

indicate whether each of the following will compile (**C**) or not compile (**N**).

1.1____ `q = new Question(10.0);`

1.2____ `r = new Question(10.0);`

1.3____ `s = new ExtraCreditQuestion(10.0);`

1.4____ `e = new Question(10.0);`

1.5____ `f = new ExtraCreditQuestion(10.0);`

2. (10 points) Given the attached `Question`, `ExtraCreditQuestion`, and `Summarizer` classes, what would be printed by each of the following fragments?

2.1

```
Question s;
s = new ExtraCreditQuestion(10.0);
s.grade(0.5);
System.out.println(s.toString());
```

_____

2.2

```
ExtraCreditQuestion f;
f = new ExtraCreditQuestion(10.0);
f.grade(true);
System.out.println(f.toString());
```

_____

2.3

```
ExtraCreditQuestion g;
g = new ExtraCreditQuestion(10);
g.grade(0.5);
System.out.println(g.toString());
```

_____

2.4

```
Question s;
s = new ExtraCreditQuestion(10.0);
s.grade(0.5);
Summarizer.display(s);
```

_____

2.5

```
ExtraCreditQuestion s;
s = new ExtraCreditQuestion(10.0);
s.grade(0.5);
Summarizer.display(s);
```

_____

3. (5 points) Given the following classes and interfaces:

```
public interface Sortable {
  //Details omitted
}
```

```
public interface Animated {
  // Details omitted
}
```

```
public class Versatile implements Sortable, Animated {
  // Details omitted
}
```

```
public class Utilities {
  public static void add(Animated item) {
    // Details omitted
  }

  public static void add(Sortable item) {
    // Details omitted
  }

  public static void add(Versatile item) {
    // Details omitted
  }
}
```

indicate whether each of the following fragments will compile (**C**) or not compile (**N**). (Note: A fragment only compiles if all of the statements in it compile.)

3.1____
```
Versatile v;
v = new Versatile();
Utilities.add(v);
```

3.2____
```
Animated a;
a = new Versatile();
Utilities.add(a);
```

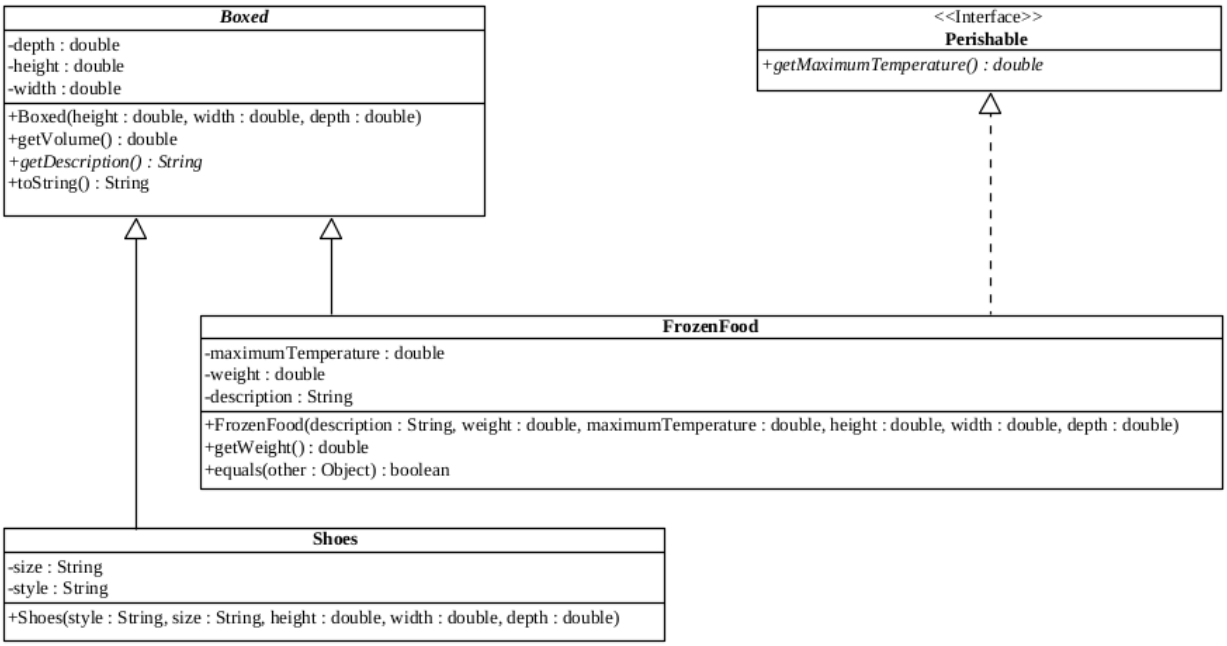
3.3____
```
Sortable s;
s = new Versatile();
Utilities.add(s);
```

3.4____
```
Animated a;
a = new Animated();
Utilities.add(a);
```

3.5____
```
Animated a;
a = new Animated();
Utilities.add(a);
```

4. (5 points) In the space below, create a UML class diagram for the attached `Question` and `ExtraCreditQuestion` classes.

5. (15 Points) Implement the components summarized in the following UML class diagram.



You must write all of these classes/interfaces, **all of which must be in the `exam3` package**.

Note that `Boxed`, `getDescription()` and `getMaximumTemperature()` are all in italics in this diagram. Also, note that methods in concrete classes that are implied by the specializes/extends and realizes/implements arrows are omitted from the diagram.

In addition to the specifications that are contained in the UML diagram, your answer must comply with the following specifications.

1. The `Boxed` Class
   1.1. The `getVolume()` method must return `depth * height * width`.
   1.2. The `toString()` method must return the result of calling `getDescription()` and `getVolume()`, formatted using a format `String` of `"%s with a volume of %f"`.
2. The `Shoes` Class
   2.1. The `getDescription()` method must return information about the `style` and `size` (in that order), formatted using a format `String` of `"%s in size %s"`.
3. The `FrozenFood` Class
   3.1. The `equals()` method must only return true of other is an instance of `FrozenFood` and all of the attributes of the two objects are identical.
4. Other Accessors/Getters
   4.1. Any other accessors/getters that are specified in the UML diagram must return the corresponding attributes.

6. (10 Points) Given the following `Changer` class:

```
import java.util.ArrayList;

public class Changer {

    public static void change(FrozenFood food) {
        food.setWeight(100.00);
    }

    public static void change(FrozenFood a, FrozenFood b) {
        FrozenFood temp;
        temp = a;
        a = b;
        b = temp;
    }

    public static void change(ArrayList<FrozenFood> foods) {
        FrozenFood temp;
        temp = foods.get(0);
        foods.set(0, foods.get(1));
        foods.set(1, temp);
    }
}
```

what would be printed by each of the following fragments?

```
FrozenFood food = new FrozenFood("Pizza", 32.00, 20.00, 10.00, 10.00, 1.00);
Changer.change(food);
System.out.println(food.toString());
```

```
FrozenFood pizza = new FrozenFood("Pizza", 32.00, 20.00, 10.00, 10.00, 1.00);
FrozenFood fries = new FrozenFood("Fries", 16.00, 20.00,  6.00,  6.00, 6.00);
Changer.change(pizza, fries);
System.out.println(pizza.toString());
```

```
FrozenFood pizza = new FrozenFood("Pizza", 32.00, 20.00, 10.00, 10.00, 1.00);
FrozenFood fries = new FrozenFood("Fries", 16.00, 20.00,  6.00,  6.00, 6.00);
ArrayList<FrozenFood> foods = new ArrayList<FrozenFood>();
foods.add(pizza);
foods.add(fries);
Changer.change(foods);
System.out.println(foods.get(0).toString());
```

---

7. Consider the declarations of Boat class attached. Write a class called BoatUtils with two static methods as specified below.

a) Write saveBoatList method that will write out to a filename the names of all the boats in boats. Each name should be on it's own line in the file. If either filename or boats is null, return false. If the file does not exist, throw an IllegalArgumentException using a try-catch clause. Return true otherwise.

```
public static boolean saveBoatList(String filename,   ArrayList<Boat>
boats) throws IllegalArgumentException{
```

```

}
```

b) Write loadBoatList method so that it will read in boat names from filename and storing them in the ArrayList of Strings names. If either filename or names is null, return false. If the file does not exist, the method should throw a FileNotFoundException. Return true otherwise.

```
public static boolean loadBoatList(String filename,  ArrayList<String>
names) throws  FileNotFoundException{
```

}

Attachments

```
public class Question {
    private double credit;
    private double points;

    public Question(double points) {
        this.points = points;
        this.credit = 0.0;
    }

    public boolean fullCredit() {
        return credit >= 1.0;
    }

    public void grade(boolean correct) {
        if (correct) {
            credit = 1.0;
        } else {
            credit = 0.0;
        }
    }

    public void grade(double credit) {
        if      (credit > 1.0) {
            this.credit = 1.0;
        } else if (credit < 0.0) {
            this.credit = 0.0;
        } else {
            this.credit = credit;
        }
    }

    public double value() {
        return points * credit;
    }

    public String toString() {
        return String.format("%.1f", value());
    }
}
```

```
public class ExtraCreditQuestion extends Question {

    public ExtraCreditQuestion(double points)
    {
        super(points);
    }

    public double value() {
        if (fullCredit()) {
            return super.value();
        } else {
             return 0.0;
        }
    }
}
```

```
public class Summarizer {
    public static void display(Question q) {
        System.out.println("Q:" + q.toString());
    }

    public static void display(ExtraCreditQuestion q) {
        System.out.println("E:" + q.toString());
    }
}
```

```
public class Boat {

    private int capacity;
    private String name;
    private int idNum;

    public Boat(int max_people, String nickname, int num) {
        capacity = max_people;
        name = nickname;
        idNum = num;
    }

    public boolean equals(Object obj) {
        if (obj instanceof Boat) {
            Boat boat = (Boat) obj;
            if (this.idNum == boat.idNum) {
                return true;
            }
        }
        return false;
    }

    public String getName() {
        return name;
    }

    public int getCapacity() {
        return capacity;
    }

    public String toString() {
        return name + "-" + idNum + "-" + capacity;
    }
}
```