

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 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.)

#	Answers	Code
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. Given the following Node class, draw the contents of memory after each of the statements below. The statements should NOT be considered independently. Each statement takes place after the one before it.

```
public class Song{
    // Notice that these are public!
    public String title;
    public Song next;

    public Song(String title, Song next){
        this.title = title;
        this.next = next;
    }
}
```

4.1

```
Song sx = new Song("Bingo", null);
Song head = new Song("Roar", sx
```

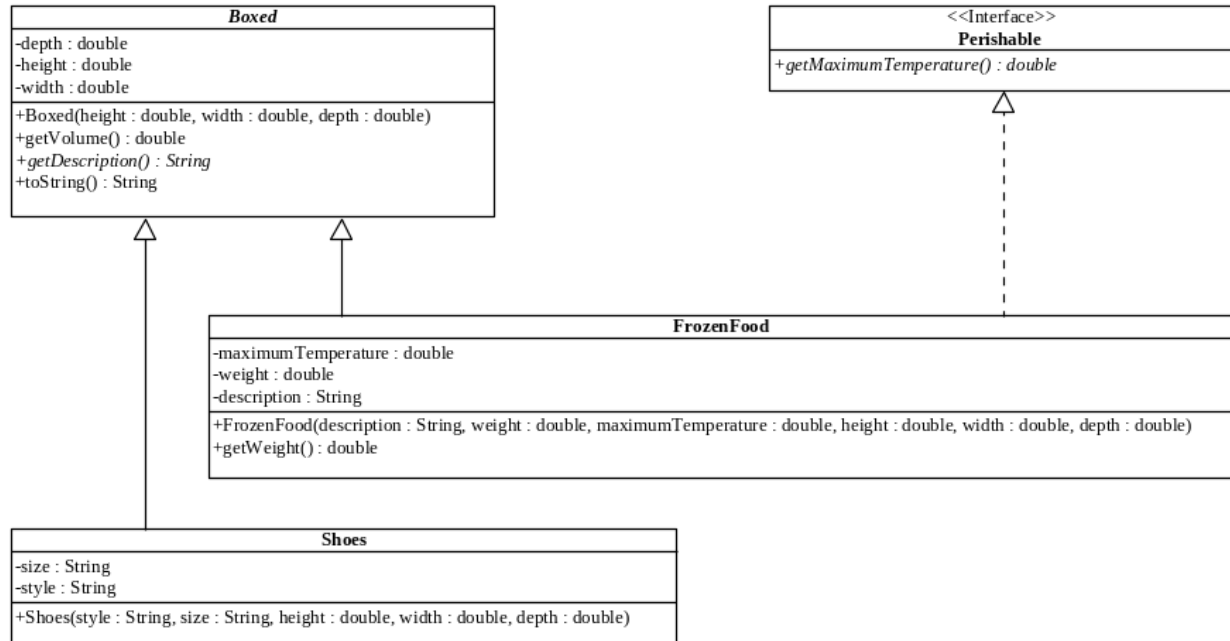
4.2

```
Song sy = new Song(null, "Believe");
sx.next = sy;
```

4.3

```
head.next.title = "Hello";
```

5. The system you must implement is summarized in the following UML class diagram.



In the space that follows, implement all of the following classes and interfaces. You must explicitly include all visibility/accessibility modifiers and the abstract modifier in all relevant declarations.

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  $\text{depth} * \text{height} * \text{width}$ .
- 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. Other Accessors/Getters

- 3.1. Any other accessors/getters that are specified in the UML diagram must return the corresponding attributes.

```
// The implementation of Boxed
```

```
// The implementation of Perishable
```

```
// The implementation of FrozenFood
```

```
// The implementation of Shoes
```



## 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());  
    }  
}
```