IMPORTANT: Write your FULL NAME and SECTION on the back of every page

44-542 Object Oriented Programming Spring 2016

Exam 2 Part 2 Version A (50 pts)

1. 10 pts. **Implement an interface**

```
public interface Sendable {
    /**
     * @return the string of character that will be sent
     */
    String allTextToSend();

    /**
     * @return the distance to send the message
     */
    double getDistance();

    /**
     * @return The cost to send the message
     */
    double findCost();
}
```

You will create a class `SafeMessage` that implements the `Sendable` interface. The `SafeMessage` class will have two attributes:

- `basicText` – the basic message (String)
- `distance` – the distance to send the message (double)

You will need to implement a two argument constructor. You may assume that the distance value passed into the constructor is positive. The only methods you need to implement are the ones to fulfill your contract with the interface. In particular,

- allTextToSend will repeat basicText twice (just to be safe)
- findCost will compute the cost based on allTextToSend and is $1.00 for the first 3 letters and $0.10 for each letter after that.

The following shows an example of how a client might use this class.

```
Sendable safe = new SafeMessage("abcd", 2000.0);
System.out.println(safe.allTextToSend() + " " +
      safe.getDistance() + " " +
      safe.findCost());
```

It will print:

```
abcdabcd 2000.0 3.5
```

Write the entire `SafeMessage` class on the next page.

PRINT Your Name: ____

Your Section: ____

Your code for the `SafeMessage` class:

```
public class SafeMessage implements Sendable {

    private String basicText;
    private double distance;

    public SafeMessage(String text, double distance) {
        this.basicText = text;
        this.distance = distance;
    }

    public String allTextToSend() {
        return basicText + basicText;
    }

    public double getDistance() {
        return distance;
    }

    public double findCost() {
        double cost = 0.0;
        int letters = basicText.length() * 2;
        if (letters > 3) {
            cost = 3.0 + 0.1 * (letters - 3);
        } else {
            cost = letters;
        }
        return cost;
    }

} // end class SafeMessage
```

PRINT Your Name: ____

Your Section: ____

2. 10 pts. **Sorting**
Suppose we have the class Book as given in the following code.

```
public class Book {
      private String title;
      private int pages;
      private double cost;

      public Book(String title, int pages, double cost) {
            this.title = title;
            this.pages = pages;
            this.cost = cost;
      }

      public String toString(){
            return "Book: " + title;
      }
}
```

Which interface must you implement for Book to have a natural order?

```
Comparable
```

How will you need to change the declaration of the Book class?

```
Add implements Comparable<Book>
```

Implement the compareTo method in Book so that its natural order is based on the cost and is in ascending order.

```
public int compareTo (Book other) {
      if (cost < other.cost)
            return -1;
      else if (cost > other.cost)
            return 1;
      else
            return 0;
}
```

Show how you would sort an ArrayList <Book> named myBooks based on the natural order using the appropriate method from Collections.

```
Collections.sort(myBooks);
```

PRINT Your Name: ____

Your Section: ____

3. 10 pts. **Trace Exception**
What is the output of the following code: (You may assume that an OppsException class that extends RuntimeException has been defined and appropriate imports have been done. Furthermore, the file will be found and output will be printed.)

Contents of the file data.txt:

```
(&  5  -1.0  2  3.4  abc  -19  21.0  $)
```

```
public static void main(String[] args) throws FileNotFoundException {
      Scanner parser = new Scanner(new File("data.txt"));
      double sum = 0.0;

      while (parser.hasNext()) {
            try {
                  double data = parser.nextDouble();
                  if (data < 0) {
                        throw new OopsException("not this");
                  }
                  System.out.println(data);
                  sum += data;
            } catch (InputMismatchException e) {
                  System.out.println("A");
                  parser.next();
            } catch (OppsException e) {
                  System.out.println("B:" + e.getMessage());
                  parser.next();
                  }
      } // end while
      System.out.println("Sum is: " + sum);
} // end main
```

OUTPUT

```
A
5.0
B:not this
3.4
A
B:not this
A
Sum is: 8.4
```

PRINT Your Name: ____

Your Section: ____

IMPORTANT: Write your FULL NAME and SECTION on the back of every page

4. 5 pts. **ArrayList**

Answer all the parts. Please write neatly. Write the Java source code in the given spaces.

```
public ArrayList<String> longerThan(int min,
     ArrayList<String> data) {
```

// Part A. (1pt) Declare a variable `result` that will hold a list of Strings. Create an array list and assign it to that variable.

```
List<String> result = new ArrayList<String>();
```

// Part B. (3 pts) Use an enhanced for loop to determine which of the strings in data have a length that is greater than min and add them to result.

```
for (String value : data){
    if ( value.length() > min)
        result.add(value)
}
```

// Part C. (1 pt). Write the return statement for the longerThan method.

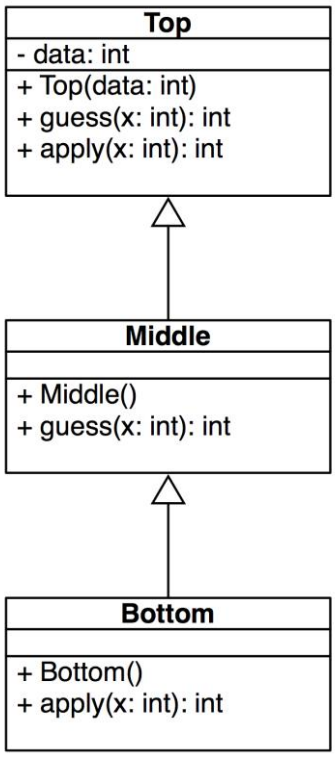
```
return result;
```

```
} // end of longerThan
```

PRINT Your Name: ____

Your Section: ____

5. 15 pts. **Trace class hierarchy**
Consider the following code for three classes Top, Middle, and Bottom. (A UML diagram is included for your convenience.)




```
public class Top {
    private int data;
    public Top(int data){
        this.data = data;
    }
    public int guess (int x){
        System.out.println("Top guess " + x);
        data = x;
        return data + x;
    }

    public int apply (int x){
        System.out.println("Top apply " + x);
        return guess(x + 2);
    }
} // end class Top

public class Middle extends Top {
    public Middle(){
        super(11);
    }
    public int guess(int x){
        System.out.println("Middle guess " + x);
        return super.guess(x)+11;
    }
}  // end class Middle

public class Bottom extends Middle {
    public int apply(int x){
        System.out.println("Bottom apply " + x);
        return -1;
    }
} // end class Bottom
```

PRINT Your Name: ____

Your Section: ____

IMPORTANT: Write your FULL NAME and SECTION on the back of every page

What is the output of the following code? Provide your answer in the box.

```
public static void main(String[] args) {
   Top high = new Top(8);
   System.out.println(high.guess(4));
   System.out.println(high.apply(7));

   Top medium = new Middle();
   System.out.println(medium.apply(9));

   Top low = new Bottom();
   System.out.println(low.guess(5));
   System.out.println(low.apply(8));

} // end of main
```

OUTPUT

```
Top guess 4
8
Top apply 7
Top guess 9
18

Top apply 9
Middle guess 11
Top guess 11
33

Middle guess 5
Top guess 5
21
Bottom apply 8
-1
```

END OF PART 2

PRINT Your Name: ____

Your Section: ____