Computer Science II — Spring 2023

Exam #1

This exam should have five pages. Closed book and notes.
No computers allowed.

Problem 1: [24 points]

a) In lab we discovered that if you extend a class that implements an interface, the subclass is automatically considered to implement the interface as well. (`TrickyGame` extended `AdventureGame`, which implemented `Controllable`, for example.) How does the compiler know that the subclass necessarily implements the *interfaces* its parent does?

b) Interfaces seem like a weaker version of inheritance — they allow polymorphism (e.g. `Controllable` allowed the Autopilot class's `find` method to be made polymorphic), but classes implementing an interface don't inherit any functionality like they would if they inherited from a parent class. Why do we need interfaces in addition to inheritance?

These class definitions are used by the questions on the next two pages. Feel free to tear this page out of the exam so you can be looking at it as you read the questions below.

```
public class Base
{
    protected String name = "Marvin";

    public void report() {
        System.out.println("Hi, I'm "+name);
    }
}


public class Thingy extends Base
{
    public Thingy(String who) {
        name = who;
    }

    public void report() {
        System.out.println("Reporting:");
        super.report();
    }

    public void rename(String who) {
        name = who;
    }
}


public class Other extends Thingy
{
    public Other() {
        super("Eleanor");
    }

    public void report() {
        System.out.println("I think my name is "+name);
    }

    public void nameCheck(String who) {
        if (name.equals(who)) {
            System.out.println("Yep, I'm "+who);
        }
        else {
            System.out.println("Nope, that's not me");
        }
    }
}
```

Problem 2: [28 points]

a) Is the following code legal? If so, what output would it produce? If not, explain why not.

```
Base x = new Thingy("Brad");
x.report();
```

b) Is the following code legal? If so, what output would it produce? If not, explain why not.

```
Thingy x = new Base();
x.report();
```

c) Is the following code legal? If so, what output would it produce? If not, explain why not.

```
Base x = new Other();
x.report();
```

d) Is the following code legal? If so, what output would it produce? If not, explain why not.

```
Base x = new Other();
x.nameCheck("Brad");
```

The questions below refer to this method, which makes use of the code on page 2:

```
public void renameSome(ArrayList<Thingy> things) {
    for(int i=0; i<things.size(); i++) {
        if (i % 2 == 0) {      // That's the mod operator
            Thingy item = things.get(i);
            item.rename("Brad");
        }
    }
    System.out.println("Done");
}
```

Problem 3: [28 points]

a) Is the `renameSome` method polymorphic? Explain.

b) Can the input list contain instances of `Base`? Explain why or why not.

c) Write a *T(n)* function that represents the worst-case number of computational steps required to execute `renameSome` on a list of length *n*. Explain your answer.

d) What's *O(n)* for `renameSome` when passed a list of length *n*? For full credit, justify your answer by providing appropriate values for *c* and n_0 given your *T(n)* from above.

Problem 4: [20 points]

The following method takes two die instances, rolls them 1000 times, and returns the winner. Rewrite this method in the space below so that if one of the die instances throws an exception (*any* sort of exception) while being rolled, it's disqualified and the other die is returned as the winner.

```
public static BasicDie compareDice(BasicDie die1, BasicDie die2)
{
     int total1 = rollRepeatedly(die1, 1000);
     int total2 = rollRepeatedly(die2, 1000);

     if (total1 > total2) {
          return die1;
     }
     else {
          return die2;
     }
}
```