CS211 Summer 2012

Dr. Kinga Dobolyi

Final Examination

1. Examine the code below, and answer the following questions. (53 points)

```
import java.io.*;
import java.util.ArrayList;
public abstract class MyClass{
   private static int age;
   public ArrayList salaries;
   public int compareTo(Object o){
     MyClass other = (MyClass)o;
     if (other.name == null)
           return 1:
     else
           return (Math.abs(name.compareTo(other.name)));
   }
   private boolean find(Integer wage){
     for(int i = 0; i < salaries.size(); i++)</pre>
           if (salaries.contains(wage))
                 return true;
     return false:
   }
   private static void encode(int result){
     System.out.println(name);
                                       //1
     System.out.println(age);
                                       //2
     System.out.println(salaries);
                                       //3
     System.out.println(result);
                                       //4
   protected void openFile(){
     BufferedReader buf = null;
     try {
           buf = new BufferedReader(new FileReader("input.txt"));
           System.out.println("great success!");
     catch (Exception e) {
           System.out.println("bad file!");
     finally {
           System.out.println("close buffer!");
     }
   }
}
```

CS211 Summer 2012

Dr. Kinga Dobolyi

Final Examination

Student Name:		
Student G#:		
Student signature for Honor Code:		

а.	List the fields of MyClass that a direct subclass of MyClass could access directly (2 points)
b.	List the fields of MyClass that any class could directly access, or state that this is impossible (2 points)
c.	Is the compareTo method above recursive? Circle YES or NO, and in ONE SENTENCE explain your answer. (3 points)
d.	Assume that salaries always contains the wage you're looking for. What is the runtime of the find method for the (support your answer with ONE SENTENCE), assuming a list size of n: (6 points) a. Best possible case?
	b. Worst possible case?
	c. Average case?

e.	. Circle YES or NO whether the statements in the encode method (numbered 1 through 4) would cause a compilation error, and in ONE SENTNCE explain why (4 points)			
	Statement 1	YES	NO	
	Statement 2	YES	NO	
	Statement 3	YES	NO	
	Statement 4	YES	NO	
f.	What will be the output	of the openF i	ile method when input.txt does not exist? (2 points)	
g.	What will be the output opoints)	of the openF i	ile method when it is able to open the file input.txt? (2	
h.	 If I wanted to change the find method to return the Integer found, rather than a boolea value, and I had the expression salaries.get(i) 			
	What type would this ever runtime? (2 points)	aluate to at co	mpile time? What type would this evaluate to at	

i.	Override the equals method from the base class Object for MyClass in such a way that two instances are equal when all of their fields have the same values (which does not necessarily mean they are the same object in memory in terms of memory references). Your method must never throw an exception. Also, you must use at least one conditional in your method that makes use of short-circuiting as we have discussed in class. (10 points)
j.	Is it possible to write a constructor for MyClass and still have the code compile? If YES, then explain why we would ever want to do this. If NO, then explain why not. Each explanation should be ONE SENTENCE. (4 points)
k.	If I were to write a main method for MyClass, list all of the methods that currently exist in the code above that such a main method could call. (2 points)

2. Give the output for the following code (you can assume the **toString** method of **Object** returns a **String** such as "[0x0123456]" which refers to a memory address – we don't care what the actual value of the memory is):

a. Write the output below: (3 points)

b. Explain what polymorphism is in Java using the example above. (4 points)

- 3. Read all three parts below before starting to write your code. Your code must compile for full credit.
 - a. Write an interface called **Item**. An **Item** should have a void method called *print* to print out the item that uses the default **toString** methods to print out the fields of the class. (8 points)

b. Now write an abstract class called **Computer** that implements the interface from question 3-a. This class should store the brand of the computer as a **String**, as well as the price of a **Computer** as a primitive floating-point number. Any object that is a **Computer** must maintain the following invariants: the brand must be at least 3, and no more than 20 characters long, and the price must be a positive number at or below a thousand. Your code must implement these invariants for full credit. These fields should be mutable by other clases, however, the **Computer** class must exhibit the object-oriented properties of encapsulation and information hiding as we have discussed in class. (15 pts)

C.	Finally, write a concrete class MacBook which extends the Computer class with an additional field called applications that stores a list of them (each application can be stored as a String). You must write a constructor that initializes all fields of the MacBook (including inherited ones), and you may not shadow any variables. Write a toString method to print out all of the fields of the MacBook (9 pts)

(continue answer to b here if needed)

4.	Circle TRUE or FALSE for the following statements, and defend your choice in ONE SENTENCE for
	full credit. (20 points)

a. The regular expression "a*b*c*" will match an empty String

TRUE FALSE

b. The final keyword can be applied to a class, and doing so means you cannot further extend the class

TRUE FALSE

c. Using generics in Java helps eliminate ClassCastExceptions

TRUE FALSE

d. You can run bytecode on an operating system that does not have a Java compiler installed

TRUE FALSE

e. An abstract class must have at least one abstract method

TRUE FALSE

 $f. \quad \ \ \text{Java uses static binding of methods, as it is a compiled language}$

TRUE FALSE

g. Overloading a method and overriding a method are mutually exclusive

TRUE FALSE