CS211 Summer 2012

Dr. Kinga Dobolyi

Final Examination

Name and G#: ____

1. Examine the code below, and answer the following questions. (16 points)

```
public abstract class MyClass{
   private static int count = 0;
   public static String store;
   private boolean available;
   public Integer num;

   public MyClass(){
        count++;
   }

   private void func1(){
        //do nothing
   }

   public void func2(){
        //do nothing
   }

   private static void func3(){
        //do nothing
   }

   public static void func4(){
        //do nothing
   }
```

a. List the fields and methods of **`MyClass`** that a direct subclass of **`MyClass`** could access directly (4 points)

b. List the fields of **`MyClass`** that you would be able to use if you were writing code for the func1 method (4 points):

c. List the fields of **`MyClass`** that you would be able to use if you were writing code for the func2 method (4 points):

d. List the fields of **`MyClass`** that you would be able to use if you were writing code for the func3 method (2 points):

e. List the fields of **`MyClass`** that you would be able to use if you were writing code for the func4 method (2 points):

2. Imagine I have a data structure that uses links (like a binary tree or linked list) to connect nodes to each other. Each node has 2 links coming out of it: one to the next node, and one to the previous node (assume getters and setters for these links). The list is **<u>not sorted</u>**. There is a link to the start and end of the data structure, but no other information is stored/maintained. For reasons of absurdity, there is an **`add`** method for this data structure, but it adds elements only to the middle of the list. Answer the following questions in one sentence or less (using the variable *N* if you need to – you may NOT make assumptions or declarations about this variable *N*, that is, your answers must generalize to any *N*): (10 points)
    a. What is the best case (in terms of time) for finding some element in this list of size *N*?

    b. What is the worst case (in terms of time) for finding some element in this list of size *N*?

    c. What is the best case (in terms of time) for adding an element to this list of size *N*?

    d. What is the worst case (in terms of time) for adding an element to this list of size *N*?

e. Would adding a (static) variable to the data structure that keeps track of the size of the list improve the times for c and d?

3. Imagine I have a regular expression as follows. In the questions below, circle whether or not the string matches the regular expression. (5 points)

   Regular expression: ^Hel.o(W|w+)olrd\.\d\d.+

a.	HelloWorld.22	YES	NO
b.	Helooworld.222	YES	NO
c.	Hel.owworld.22k	YES	NO
d.	HelloWWorldk222	YES	NO
e.	Helloolrd.22k	YES	NO

4. Examine the following three classes (all in appropriately named files in the same folder):

```
public class Animal{
     private int weight;
     public void print() {
          System.out.println("Animal: " + weight);
     }
}


public class Dog extends Animal implements Comparable{
     private String breed = "some dog";
     public int print(int num) {
          System.out.println("Dog: " + breed);
          return num;
     }
     public String toString() { return breed; }
}
```

```
import java.util.*;
public class Final{
     public static void main(String[] args){
          ArrayList listBasic = new ArrayList();
          ArrayList<Animal> listAnimal = new
ArrayList<Animal>();
          ArrayList<Dog> listDog = new ArrayList<Dog>();

          Object o1 = new Object();
          Animal a1 = new Animal();
          Dog d1 = new Dog();

          listBasic.add(o1);
          listBasic.add(a1);
          listBasic.add(d1);
          listAnimal.add(o1);
          listAnimal.add(a1);
          listAnimal.add(d1);
          listDog.add(o1);
          listDog.add(a1);
          listDog.add(d1);


          for(int i=0; i < listBasic.size(); i++)
               System.out.println(listBasic.get(i));
          for(int i=0; i < listAnimal.size(); i++)
               listAnimal.get(i).print();
     }
}
```

a. Why will the **`Dog`** class not compile? Fix the **`Dog`** class so it would compile (you can comment out things). (2 points)
b. Why will the **`Final`** class not compile? Fix the **`Final`** class so it would compile (you can comment out things) (2 points)
c. After your fixes in parts b and c, what would the output of the code be above? (5 points)

5. Imagine that instead of a binary tree, we have a tree that has three child nodes instead of just two. The nodes in this tree are unsorted. Complete the recursive method below to print out all nodes of the tree, assuming that you call the method on the root of the tree. You the parent nodes should always be printed before any of their child nodes. (6 points)

```
public void printNode(Node curr){
```

6. Circle TRUE or FALSE for the following statements, and defend your choice in ONE SENTENCE for full credit. (20 points)
   a. Unchecked exceptions do not have to be inside of a try-catch block.
      TRUE FALSE
   b. A NullPointerException is an example of a checked exception
      TRUE FALSE
   c. Method overriding implies you can have different arguments for the conceptually same method
      TRUE FALSE

d. Method overloading is when the child class re-implements the method in the parent class

TRUE FALSE

e. The following code will always throw a runtime exception:

```
String badString = null;
if ( 1 == 1 || badString.toString())
        System.out.println("nonsense");
```

TRUE FALSE

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

TRUE FALSE

g. Whitespace is used as a delimiter in command line arguments

TRUE FALSE

h. Java uses static binding of methods, as it is a compiled language

TRUE FALSE

i. The number 23, as a base 4 number, is equivalent to 13 in base 10.

TRUE FALSE

j. The size of a primitive array in Java cannot be changed after it is declared.

TRUE FALSE

7. Complete the following method to find the minimum value of the objects in the list **`arrayList`**. The method should return the element with the minimum value. You may NOT use the **`Collections.sort`** method, or any other sorting algorithm, for this question (i.e. implementing Bubble Sort or anything is both not allowed and only makes things harder). Note that you don't know what type of objects are in the **`arrayList`**, only that they implement the **`Comparable`** interface. This question is potentially non-obvious – save it until the end. (10 points)

```
public Object findMin(ArrayList<Comparable> arrayList){













}
```

THE END! Thanks for a great semester!