write name at top

DO NOT OPEN. DO NOT START.

Scratch: rip off if you like, but turn in

If you are the first person to see this and you STAND UP and raise your hand while I am passing out exams, I will give you an extra two points*

CS211 Fall 2013

Dr. Kinga Dobolyi

Final Examination

Total points possible: 83

Name: _____

* You should NOT open an exam that says do not open on it, especially when the instructor tells you beforehand not to open it. So the set of students from the previous page should be a null set. Want two points? Think of one happy thing you will do after this exam, or after all of your finals, and draw a smiley face in the space below ☺

1. Examine the code below, and answer the following questions. **You may not add any new methods or change any method signatures.** (21 points)

```
public                    class                    FinalExam {
                int age;
                String name;
      public FinalExam(int age){




      }

      public              equals(                  other){









      }
}
```

   A. Modify the class (write it above) so that the invariant that the `age` is always a positive number is maintained. (4 pts)
   B. Modify the class (write it above) so that the class cannot be extended. (2 pts)
   C. Modify the class (write it above) so that the invariant that the `name` is always at least two characters long, and never more than 8 characters long, is always maintained. (4 pts)
   D. Modify the class (write it above) so that it uses good object-oriented design. (2 points)
   E. Complete the `equals` method above, such that two `FinalExam` objects are equal when they have the same `age` and `name`. Your method must never raise an exception. (9 pts)

2. Examine the code below, and answer the following questions. **If your answer is the same as another, you can write SAME AS QUESTION <question number>!** (35 points)

```
public abstract class Exam implements Comparable{

        private int age;
        public ArrayList list;
        public static String type;
        private static boolean flag;

        public Exam(int ageIn){
             age = ageIn;
        }

        private void func1(){
        }

        private static int func2(){
             return 0;
        }

        public static void func3(){
        }

        public int func4(){
             return 1;
        }

        public boolean something (Exam o){
             return false;
        }
   }
```

a. List the attributes and methods of **`Exam`** that a direct subclass of **`Exam`** could access directly

b. List the attributes and other methods of **`Exam`** that **`func1`** could access/use directly

c. List the attributes and other methods of **`Exam`** that **`func2`** could access directly

d. List the attributes and methods of **`Exam`** that **`func4`** could access directly

e. Imagine I had the standard **`main`** method *inside this class*. Which attributes and methods could the **`main`** method call from itself?

f. Imagine I have the following inside a main method in another class called `Driver`, that is NOT a child class of `Exam`. In the same problem, imagine `Exam` has been extended by `Exam2`, which has a default constructor that uses super to call `Exam`'s constructor and initializes the `age` to be 3. Assume both `Exam` and `Exam2` compile. Cross out all lines below that will NOT compile (inside the main method of `Driver`):

```
Object o1 = new Exam(3);
Exam ex1 = new Exam2();
Exam.func3();
Exam2.func3();
Exam.func2();
ex1.func3();
ex1.func1();
Exam.func1();
Exam2.type = "an exam";
Exam2.flag = false;
ex1.list = new ArrayList();
System.out.println(ex1.age);
```

g. Does the class `Exam` compile as written? Why or why not?

3. Examine the following classes, and give the output. Assume that the classes are in the same folder and compile. (9 points)

```
public class FinalExam {
     private String location;
     private static int count;

     public FinalExam(String l, int c){
          location = l;
          count = c;
     }

     public String calculate(int times){
          System.out.println(location + " timesParent: " + times +
               ", count: " + count);
          count = count + 1;
          if (count < times)
               calculate(times - 1);
          return "" + times;
     }

     public String toString(){ return location + " " + count;}
}

public class ChildExam extends FinalExam{
     private static int counter = 1;

     public ChildExam(String l, int c){
          super(l,c);
     }

     public void calculate(int times, int cutoff){
          System.out.println("timesChild: " + times);
          while (cutoff >= counter){
               super.calculate(times);
               cutoff--;
          }
     }

     public String toString(){
          return super.toString() + " " + counter;
     }
}
```

```
public class Driver {

	public static void main(String[] args){
		int x = 12;
		String string = "yellow";
		FinalExam f1 = new FinalExam(string, x);
		string = "pink";
		x = 3;
		ChildExam c1 = new ChildExam(string, x);
		x++;
		ChildExam c2 = c1;
		string = f1.calculate(5);
		c1 = new ChildExam("blue", 3);
		c1.calculate(2);
		c1.calculate(2,2);
		System.out.println(f1.toString());
		System.out.println(c1.toString());
		System.out.println(c2.toString());
	}
}
```

4. Complete the following recursive method below. It takes as argument an **`ArrayList`** of **`String`** objects, and returns a **`String`** that is made up of all pieces of the strings inside in the list, in order, that match the regular expression described below. Your method must: (18 points)
    a. Be recursive. You cannot use any for or while loops. Non-recursive methods will lose significant points.
    b. It must collect all strings that match the following regular expression: a name, which is made up of at least two and at most seven lower and uppercase letters, followed by two or more whitespace characters, followed by any positive integer, followed by a period, followed by either {two of the uppercase Q, or as many lowercase q-s as you would like}. Recall that the **`String`** class has a **`public boolean matches(String regex)`** method. Each element should be separated by a newline in the string you return.
    c. Your method must rely and use generics as indicated by the method signature below. You may not change the method signature.
    d. You may assume that the method, called from main below, would have the output below:

```
public static void main(String[] args){
     ArrayList<String> list = new ArrayList<String>();
     list.add("Kinga  1.QQ");
     list.add("Kingaaaa  1.QQ");
     list.add("2Kinga  1.QQ");
     list.add("i 1.");
     list.add("Kinga  100.QQ");
     list.add("Kinga  100.Qq");
     list.add("Kinga  100.qq");

     System.out.println(collect(list,0));
}
```

with output:

```
Kinga  1.QQ
Kinga  100.QQ
Kinga  100.qq
```

As we did in class, make sure to test your code before you turn in your exam! Think of some other test cases as well…

```
public static String collect (ArrayList<String> list, int ctr){
```

Scratch paper: write your name at top and turn in (or do not rip off)