Name: ____________________ Andrew Id: ____________________

15-121 Sample Assessment 2

Up to 50 minutes. No calculators, no notes, no books, no computers. Show your work!

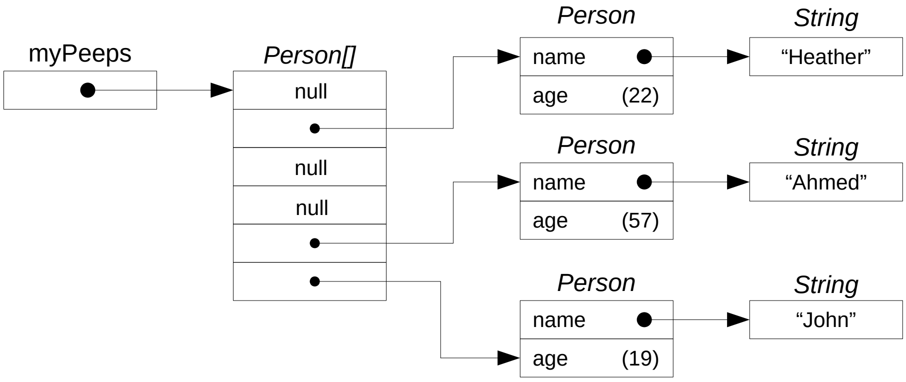
1. (6 points) **Code Tracing**: Indicate what the following program prints. Place your answer (and nothing else) in the box under the code.

```
public class Quiz4CT {
    private int a;
    private int b;
    private int c;

    public Quiz4CT(int a, int b) {
        this.b = a;
        this.a = b;

        for (int i = a; i < b; i++) {
            c += i;
        }
    }

    public void mystery(int d, int a) {
        c = a / b + d;
        int t = a;
        a = b;
        b = t;
    }

    public String toString() {
        return "a: " + this.a + ", b:" + this.b + ", c:" + this.c;
    }

    public static void main(String[] args) {
        Quiz4CT q1 = new Quiz4CT(3, 5);
        System.out.println(q1);
        q1.mystery(7, 15);
        System.out.println(q1);
    }
}
```

2. (6 points) **Code Tracing**: Consider the following two classes, and indicate what is printed when the main method of the B class is executed. Place your answer (and nothing else) in the box under the code.

```
public class A {
    protected int a;
    protected int b;

    public A(int a, int b) {
        this.a = a;
        this.b = b;
    }

    public int mystery() {
        return a+b;
    }

    public void printVals() {
        System.out.println("Y");
    }
}

public class B extends A {
    public B(int a, int be) {
        super(be, a);
        System.out.println("P1: " + a);
        System.out.println("P2: " + b);
        b = a - 2;
        a = be + 2;
        System.out.println("P3: " + this.a);
        System.out.println("P4: " + this.b);
    }

    public void printVals() {
        System.out.println("R");
    }

    public static void main(String[] args) {
        B bob = new B(15, 20);
        System.out.println(bob.mystery());

        A andrew = bob;
        andrew.printVals();
    }
}
```

3. (4 points) **Code From Picture**: Consider the following picture:



Assuming the `Person` class exists and has a constructor that takes `name` and `age` as arguments, write the appropriate lines of code to create the correct variables and objects that match what you see pictured above. Your code does not need to be inside of a method or a class. (This is a short code snippet.)

4. (10 points) **Free Response:** Consider a file containing lines of words separated by commas. The number of words per line is not fixed. Write the public static method `uniqueWords` which, given a filename, returns an `ArrayList` of all of the *unique* words in the file (meaning no duplicates).

   If there is an error opening the file, return `null`.

   Consider this example. If `input.txt` contains the following:

   ```
   cat,dog,bear,tiger
   spider,ant,cockroach
   thing
   alien,spider,bear,tiger,human
   ```

   Then `uniqueWords("input.txt")` would return an `ArrayList` containing:

   ```
   [cat, dog, bear, tiger, spider, ant, cockroach, thing, alien, human]
   ```

5. **Free Response:** Animal Names

Consider the following code that uses the `AnimalNames` class:

```
// Constructor takes a string of comma separated names
AnimalNames a = new AnimalNames("Fido,Jacky Poo,Salk");
System.out.println(a);
a.addName("Izzy");
System.out.println(a);
a.removeName("Jacky Poo");
System.out.println(a);
```

When executed, this code provides the following output:

```
AnimalNames: Fido Jacky Poo Salk
AnimalNames: Fido Jacky Poo Salk Izzy
AnimalNames: Fido Salk Izzy
```

(Notice the strange ambiguity in the `toString` result: It is not possible to distinguish between names with spaces in them and two different names. This is unfortunate, but it is what it is.)

You will write the code for four methods of this class. Other than those four methods, you cannot modify other things about the class (such as by adding new methods or instance variables). The entire class can be seen here:

```
public class AnimalNames {
        private ArrayList<String> names;

        public AnimalNames(String nameStr) {
                // Your code here
        }

        public void removeName(String name) {
                // Your code here
        }

        public void addName(String name) {
                // Your code here
        }

        protected String getNames() {
                // Your code here
        }

        public String toString() {
                return "AnimalNames: " + getNames();
        }
}
```

This problem continues on the next page. Do not write any answers on this page.

(a) (4 points) Write the Constructor:

```
public AnimalNames(String nameStr) {
```

(b) (1 point) Write the following method:

```
public void removeName(String name) {
```

(c) (1 point) Write the following method:

```
public void addName(String name) {
```

(d) (2 points) Write the following method:

```
protected String getNames() {
```

6. (6 points) **Free Response:** *Really* Cute Dog Names

The class CuteDogNames is a child of the AnimalNames class, but it only allows names that would be *really* cute for a dog. A cute dog name is defined as one that starts with “Mister”.

Consider the following code that uses the CuteDogNames class:

```
CuteDogNames c = new CuteDogNames("Mister Poofy,Jack,Salk,MisterRuff,MisterFuzzyWuzzy");
System.out.println(c);
c.addName("Izzy");
c.addName("MisterSnuffles");
System.out.println(c);
```

When executed, this code provides the following output:

```
CuteDogNames: Mister Poofy MisterRuff MisterFuzzyWuzzy
CuteDogNames: Mister Poofy MisterRuff MisterFuzzyWuzzy MisterSnuffles
```

Write the CuteDogNames class.

Important Notes and Hints:

1. Your CuteDogNames class may not declare any instance variables.
2. The names variable in AnimalNames is private, so you can’t directly access it from the child. You also aren’t allowed to change the visibility of names, or add any new methods or instances variables to AnimalNames. That means solving this problem will require some creative problem solving.
3. The first line of AnimalNames’s constructor should be a call to the constructor of the superclass. (Java only allows the super class constructor to be called from the *first* line of the child class constructor. It can’t be called later in the constructor.)
4. The String class has a startsWith method that returns a boolean and does what it sounds like.