Name: _____ Andrew Id: _____

15-121 Fall 2023 Assessment 2

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

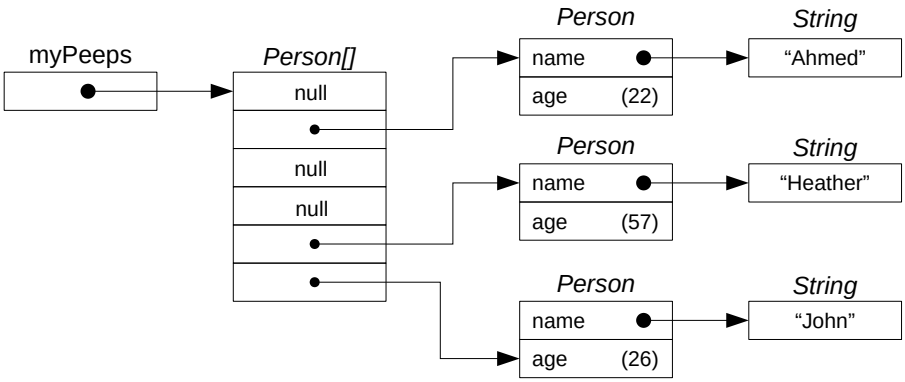
1. **Short Answer**

   Answer the following in 1 sentence, longer answers will be marked incorrect.

   (a) (2 points) What is the difference between using super() and using super?

   (b) (2 points) What is the relationship between an abstract data type and a data structure?

2. (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. (You are writing a short code snippet.)

3. (7 points) **Code Tracing**: Consider the following two classes, and indicate what is printed when the main method of the `Bill` class is executed. Place your answer (and nothing else) in the box under the code.

   Important Note: One of the lines of output contains a value in it that cannot be determined by hand, the actual value would only be known when the program runs on a real computer. When you need that value, simply write `??` in its place.

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

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

    public void mystery() {
        System.out.println("Bob");
    }

    public int magic() {
        return this.a + this.b;
    }
}

public class Bill extends A {
    public Bill(int c, int d) {
        super(d, c);
        goGoGo();
    }

    public void mystery() {
        this.a *= 2;
        this.b -= 15;
        System.out.println("Fred");
        super.mystery();
    }

    public int magic() {
        return this.a - this.b;
    }

    public void goGoGo() {
        System.out.println("A: " + this.a + "; B: " + this.b);
    }

    public static void main(String args[]) {
        Bill bob = new Bill(50, 75);
        A andrew = bob;
        bob.mystery();
        System.out.println(andrew.magic());
        bob.goGoGo();
        System.out.println(bob);
    }
}
```

4. **Free Response:** Dogs and Hamsters

Imagine that you are designing a program that will be used by a Veterinarian's office. This particular Veterinarian only sees two different kinds of pets: Dogs and Hamsters. The program you are writing has two different classes, `Dog` and `Hamster`, that can be described as follows:

- Each `Dog` has a name, age (integer number of months since birth) and a breed. The `toString` produces the following output:
  `Dog [name=Fido, ageInMonths=36, breed=Golden Retriever]`
- Each `Hamster` has a name, age (integer number of months since birth), and the dimensions of its cage (integer number of centimeters). The `toString` produces the following output:
  `Hamster [name=Feisty, ageInMonths=6, cageWidth=50, cageLength=100]`

In your program, you have already written the following `Pet` class:

```
public class Pet {
    private String name;
    private int ageInMonths;

    public Pet(String name, int ageInMonths) {
        this.name = name;
        this.ageInMonths = ageInMonths;
    }

    public String toString() {
        return "Pet [" + getPetInfo() + "]";
    }

    // A nice method to help children of this class build their toString methods
    protected String getPetInfo() {
        return "name=" + name + ", ageInMonths=" + ageInMonths;
    }
}
```

(a) (5 points) Write the `Dog` class, assuming that a `Dog` is-a `Pet`. You should use the proper principles of inheritance, and not needlessly duplicate instance variables or code from `Pet` when writing `Dog`. Your `Dog` class should include an appropriate constructor and `toString`.

(b) (5 points) Write the `Hamster` class, assuming that a `Hamster` is-a `Pet`. You should use the proper principles of inheritance, and not needlessly duplicate instance variables or code from `Pet` when writing `Hamster`. Your `Hamster` class should include an appropriate constructor and `toString`.

5. `PetList`

This question builds on the previous question. Consider the following text file (`pets.txt`) containing information about some pets seen by the local veterinarian. It contains information about four different pets: Two dogs and two hamsters.

```
D,Fido,36,Golden Retriever
H,Feisty,6,50,100
D,Saluk,125,Greyhound
H,Speedy,17,30,60
```

You need to write the `PetList` class that will operate as follows. When the following code is executed...

```
PetList myPetList = new PetList("pets.txt");

System.out.println("All pets:");
System.out.print(myPetList);

System.out.println("Just hamsters:");
ArrayList<Hamster> hList = myPetList.getHamsters();
for(Hamster h: hList) {
    System.out.println(h);
}
```

then the following output is generated...

```
All pets:
Dog [name=Fido, ageInMonths=36, breed=Golden Retriever]
Hamster [name=Feisty, ageInMonths=6, cageWidth=50, cageLength=100]
Dog [name=Saluk, ageInMonths=125, breed=Greyhound]
Hamster [name=Speedy, ageInMonths=17, cageWidth=30, cageLength=60]
Just hamsters:
Hamster [name=Feisty, ageInMonths=6, cageWidth=50, cageLength=100]
Hamster [name=Speedy, ageInMonths=17, cageWidth=30, cageLength=60]
```

Here is a copy of the `PetList` class:

```
public class PetList {
    private ArrayList<Pet> theList;

    public String toString() {
        String ret = "";
        for (int i = 0; i < theList.size(); i++) {
            ret += theList.get(i).toString() + "\n";
        }
        return ret;
    }

    /**
     * Read all dogs and hamsters from filename and add them to theList.
     *
     * @param filename The name of the file to read the pets from
     */
    public PetList(String filename) {
        // You will write this code
    }

    /**
     * Produce and return an ArrayList containing all of the hamsters in theList.
     *
     * @return An ArrayList of hamsters
     */
    public ArrayList<Hamster> getHamsters() {
        // You will write this code
    }
}
```

(a) (10 points) Write the constructor of the `PetList` class. You may not modify anything else about the class except its constructor.
Hint: You should *not* create any `Pet` objects. You should create `Dog` and `Hamster` objects.

```
public PetList(String filename) {




}
```

(b) (5 points) Write the `getHamsters` method which returns an `ArrayList` of all the `Hamster` objects found in `theList`. You may not modify anything else about the class except `getHamsters`.

```
public ArrayList<Hamster> getHamsters() {




}
```