**Instructions:** The midterm is 70 minutes long. Read each question before starting the test. Write your answers in the space provided. Write your name at the top of each page that you submit. **Closed book**. **No phones, calculators, etc.,** allowed. Good luck!

**Note:** Do not fold or crumple pages.

Student Name (print): ______________________________

Student ID:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

Tutorial Section A1 (Friday at 8:30 am ) ☐

Tutorial Section B2 (Friday at 10:00 am) ☐

Tutorial Section A2 (Friday at 11:35AM ) ☐

Tutorial Section B1 (Friday at 4:00 PM) ☐

## A: Creating a Class [10 marks]

Write a Java class called `Person`. Each person will have a name (String), a favourite kind of pet (String), and a list of every pet that this person has ever had (array of Strings). The pet list may contain the same kind of pet multiple times. Include a constructor that has three input parameters (for name, favourite kind of pet and pet list) and sets the initial state for the object. The constructor *must* make a new array for the list of pets and copy the data from the input array to this new one. Write a `superHappy()` method that returns `true` if at least 1/2 of all the pets that this person has had is their favourite kind of pet and returns `false` otherwise. Example code usage is given below.

Do **not** use information hiding for this class. Make *everything* accessible.

```
String[] petsHarmeet = new String[]{ "cat", "dog", "dog", "eel", "cat", "cat"};
Person p = new Person("Harmeet", "cat", petsHarmeet);
// assert: p.superHappy() is true
```

## B: Using a Class [10 marks]

Assume that the `Person` class from the last question is given to you. Complete the following two methods that are in another class that uses `Person` objects.

```
/* counts how many times anyone has had a given pet */
public static int countPet( String pet, Person[] people ){
```

```
/* returns the name of a person that has never had any pets that were their */
/* favourite kind of pet. If more than one person satisfies this return the */
/* FIRST one you find in the input                                          */
public static String unhappyPerson( String pet, Person[] people ){
```

## C: Class Inheritance [10 marks]

Consider the following QQ class:

```
public class QQ{
  public int q;
  public QQ(int q){
    this.q = q;
  }
}
```

The intention is that objects of this class have some state (the attribute q) that once set cannot be changed. First rewrite this class so that (a) the value of q can never be changed, and (b) it uses **information hiding**. Add any appropriate getters/setters. Next, write a new class WW that extends the class QQ. The new class should (c) introduce a new attribute (a String labeled w) and have a constructor that takes an integer (for q) and a String (for w) as input and sets the state of the object approriately.

## D: Loops and Arrays [10 marks]

Complete the following Java method. The method takes a list of numbers (int[]), a lower bound (int) and an upper bound (int) as input and returns a new list (array of ints) that consists of all the input numbers that are smaller than the lower bound or larger than the upper bound. The order of the numbers in the output array must be the same as they appeared in the input array. The output array must be the correct length.

```
public static int[] notInRange( int[] numbers, int lower, int upper ){
```

## E: Memory Model [10 marks]

Consider the following `main()` method of a class.

```
public static void main( String[] args ){
        byte      a = 7;
        String    b = "cow";
        int[]     c = new int[]{2,4,6,8,10,15,20};
        String[]  d = {"elk", "bat", "cow"};
        boolean[] e = new boolean[2];
}
```

Draw a picture (box and arrow diagram) of the memory model when this program is running at the time just before the programs ends. Align your diagram with the labels below.

```
---------------     -----------------------------     ----------------
     stack                      heap                     static area
```