Lab 10 – Even more on Classes, Objects, and Methods

Introduction

At this point, you should be familiar with classes, objects, constructors, instance variables, and methods, and you should be able to create your own classes having these components. The present lab requires you to use these (and so you should review material from previous labs if you need a reminder); however, it also requires you to incorporate *static* variables and *static* methods in your program.

When an instance variable is defined in a class, each instance of the class has its own values for that variable. That is, it is possible to assign a value to the variable for one object, and this has no effect on the values of the variable in other objects. A static variable, however, is associated with the class itself and not any particular instance of the class. It is accessible to all instances of the class, but its value can be accessed and altered without ever even creating an instance of the class. This is why static variables are also sometimes called *class variables*. Static variables and methods are defined using the keyword **`static`** (which usually goes after the access modifier in a method or variable declaration—e.g., **`public static int counter`**…).

In this lab, you will create a class called **`Person`**. Some variables and methods of the class (for instance, **`age`**, **`name`**) will be instance-level. However, others will be defined at the level of the class. These will be used to keep track of how many instances of the **`Person`** class have been created as well as to store and allow access to other information not directly related to any particular instance of the **`Person`** class.

Lab Objectives

By the end of the lab, you should have gained further experience working with class constructors, access modifiers, instance variables and methods (including accessor and mutator methods). However, by the end of the lab, you should also be able to work with static variables and methods, and you should understand the difference between them and instance variables and methods.

What to Submit

The **`Person.java`** file should be submitted to our course website for grading.

Instructions

Part 1

- Create a class called **`Person`**. Include the standard header comment stating your name, the date, program purpose, and containing the statement of academic honesty. When writing, you must also abide by the Java formatting and style conventions.
- Unless otherwise specified, you must also follow the normal conventions pertaining to encapsulation.
- In the class, you should define the following instance variables:
    - **`age`**, an **`int`** which specifies a person's age;
    - **`name`**, a **`String`** which holds a person's name;
    - **`isFemale`**, a **`boolean`** variable specifying whether a person is female.
- The following static variables should also be defined.
    - **`totalPersons`**, an **`int`** which records the number of new **`Person`** objects created during the current run of the program.
    - **`totalFemales`**, an **`int`** which records the number of **`Person`** objects that are female. Since we will allow the gender of a person to be changed, the value for **`totalFemales`** might increase or decrease over time.

Lab 10 – Even more on Classes, Objects, and Methods

    - **`totalAge`**, an **`int`** which keeps track of the sum of the ages of all persons. This value might also increase or decrease over time.
- A constructor for the **`Person`** class should be defined, accepting a name, age, and whether the person is female or male. The values of these parameters should be used to initialize the instance variables specified above.
- Importantly, the constructor should validate the values of the parameters before assigning them to the instance variables. Specifically, the constructor must ensure that **`age`** is greater than or equal to 0 (if the value passed to the constructor is less than 0, then **`age`** should be assigned 0). If a null reference is passed to the constructor for **`name`**, then the constructor should set **`name`** to **`"Unknown".`**
- The constructor should also update the static variables appropriately: It should increment **`totalPersons`** and **`totalAge`** when a new **`Person`** object is created, and also **`totalFemales`** if that person is specified to be female.
- Appropriate accessor (getter) and mutator (setter) methods for the instance variables should be defined. These should follow the usual naming conventions for accessors and mutators: **`getAge()`**, **`setAge(int age)`**, **`getIsFemale()`**, **`setIsFemale(boolean isFemale)`**, etc. The return types of the getter methods should match the data types of the underlying variables.
- Note that the setter methods, like the constructor, should also validate the input: **`setAge`** should set the age to the new value passed as parameter only when this new value is greater than or equal to 0. If the String reference passed to the method **`setName`** is **`null`**, then the name of the **`Person`** object should be left unchanged.
- The setter methods should also update the static variables when appropriate. Note that if the gender of a person is changed, then **`totalFemales`** should be updated. Similarly, if a person's age is changed, then this should affect **`totalAge`**.
- The class should also include an **`equals()`** method that accepts another **`Person`** object as a parameter. The method should return true if and only if this other Person object's instance variables are equal to (ignoring case where appropriate) the calling object's instance variables. Otherwise, the method returns false.
- The instance method **`isYounger()`** should also be defined, using another **`Person`** object as a parameter. It should return true if and only if the calling object's age is less than the parameter object's age. Otherwise, the method returns false.
- A **`toString()`** method should be implemented, returning the **`name`**, **`age`**, and **`isFemale`** variables in a human readable String format as shown below.

```
Name: Carol Marcus
Age:  25
Female
```

- In addition to the above instance methods, the following static methods should also be defined. Each has no parameters.
    - **`avgAge()`** returns the average age (as a **`double`**) of all **`Person`** objects created;
    - **`howManyPeople()`**: returns the total number (an **`int`**) of **`Person`** objects created;
    - **`howManyFemales()`**: returns the total number (an **`int`**) of females.

Lab 10 – Even more on Classes, Objects, and Methods

Part 2

- Create a new class called **`PersonTester`**. This will be a program with just a **`main`** method to test the above class, and it will not need to be submitted. The I/O of **`PersonTester.java`** should be consistent with the formatting as shown in the additional examples found at the end of this lab.
- Create four new **`Person`** objects (**`person1`**, **`person2`**, **`person3`**, **`person4`**) with the properties listed below.

person1	person2	person3	person4
Name: Marcus Smith Age: 23 Male	Name: Mary Brown Age: 21 Female	Name: Pat Age: 40 Male	Name: Marcus Smith Age: 25 Male

- Use the method **`toString()`** to display each of the **`Person`** objects.
- Increase the age of **`person1`** by 2.
- Print out the current age of **`person1`**.
- Set the age of **`person2`** to -12.
- Print out the age of **`person2`**. (Note: the age should not be -12; it should be 21).
- Call the **`setName`** method of **`person3`** using **`null`** as a parameter.
- Set the gender of **`person3`** to female.
- Use the method **`toString()`** to display **`person3`**.
- Print out whether **`person4`** is younger than **`person3`**.
- Print out whether or not **`person1`** and **`person4`** are the same person (have the same name, age, and gender).
- Print out the following:
    - the total number of **`Person`** objects created,
    - the total number of females, and
    - the average age of all the **`Person`** objects.
- Additional examples are found at this end of this lab for further testing purposes. These additional examples do not need to be submitted with your **`Person.java`** file. However, the I/O formatting of your class must be consistent with these examples.

Submission and Grading

After you have completed and thoroughly tested your program, upload and submit **`Person.java`** to our course webpage in order to receive credit for the lab (if you have any questions about how to submit this assignment, then ask your lab instructor for help at least 24 hours before this assignment is due). Always double check that your submission was successful on our course website.

The lab will be graded according to the following guidelines.

- A score between 0 and 100 will be assigned.

Lab 10 – Even more on Classes, Objects, and Methods

- If the source file(s) are not submitted before the specified deadline, if they do not compile, or if the submitting student has an unexcused absence for a single lab period devoted to the assignment, then a grade of 0 will be assigned. (Note: students who show up to their first lab period and show their lab TA that they finished and submitted the week's lab assignment may be excused from their second lab period for that week.)
- If the required comment for all labs describing the program and the academic honesty statement is not included at the top of the file, then 10 points will be deducted. Note: this required comment can be found in Lab 02.
- The program will be evaluated using a separate testing file where multiple instances of the **`Person`** class will be created and their methods invoked.

Additional Examples

The examples below are for additional testing purposes only, and they do not need to be submitted. However, the I/O of your submitted class should follow the same format as these examples.

Example 1
Example main method:

```
Person person1 = new Person("Roger", 23, false);
Person person2 = new Person("Dorothy", 21, true);

System.out.println("person1 = \n" + person1 + "\n");
System.out.println("person2 = \n" + person2 + "\n");

System.out.println("Avg Age: " + Person.avgAge());

person1.setAge(60);
System.out.println("person1's age is " + person1.getAge());

person1.setName("Norman");
System.out.println("person1 = \n" + person1);

System.out.println("Avg Age: " + Person.avgAge());
```

Example output:

```
person1 =
Name: Roger
Age: 23
Male

person2 =
Name: Dorothy
Age: 21
Female

Avg Age: 22.0
person1's age is 60
person1 =
Name: Norman
```

Lab 10 – Even more on Classes, Objects, and Methods

```
Age: 60
Male
Avg Age: 40.5
```

Example 2

Example main method:

```
Person person1 = new Person("Roger", 23, false);
Person person2 = new Person("Dorothy", 21, true);

System.out.println("person1 = \n" + person1 + "\n");
System.out.println("person2 = \n" + person2 + "\n");

System.out.println("Before changing person1:");
System.out.println("# of People:\t" + Person.howManyPeople());
System.out.println("# of Females:\t" + Person.howManyFemales());

person1.setAge(person2.getAge());
person1.setName(person2.getName());
person1.setIsFemale(person2.getIsFemale());

System.out.println("After changing person1:");
System.out.println("# of People:\t" + Person.howManyPeople());
System.out.println("# of Females:\t" + Person.howManyFemales());
System.out.println();
System.out.println("person1 equals person2? " + person1.equals(person2));
```

Example output:

```
person1 =
Name: Roger
Age: 23
Male

person2 =
Name: Dorothy
Age: 21
Female

Before changing person1:
# of People:      2
# of Females:     1
After changing person1:
# of People:      2
# of Females:     2

person1 equals person2? true
```

Example 3

Example main method:

```
Person person1 = new Person(null, -20, false);

System.out.println("person1 = \n" + person1 + "\n");
```

Example output:

```
person1 =
Name: Unknown
Age: 0
Male
```

Example 4

Example main method:

```
Person person1 = new Person("Jim", 20, false);
Person person2 = new Person("Nick", 22, false);

System.out.println("person1 = \n" + person1 + "\n");
System.out.println("person2 = \n" + person2 + "\n");

System.out.println("person1 younger than person2? " +
      person1.isYounger(person2));
```

Example output:

```
person1 =
Name: Jim
Age: 20
Male

person2 =
Name: Nick
Age: 22
Male

person1 younger than person2? true
```