UNBC Laboratory Assignment
Computer Science 101—Winter 2022

Static and non-static Members

Due Date:

This assignment is due Friday, January 14 *by the beginning of lecture*.

Please submit via `learn.unbc.ca`. See `learn.unbc.ca` for more instructions on what to submit.

Persons

⇒ Implement a `Person` class that has all of the methods shown in Figure 1.

⇒ Implement a `PersonTester` class whose `public static void main()` method uses the `Person` class methods to test the class and simultaneously tells a story.

The `allSayHello` static member function should cause every currently living person to say hello. The tricky part of this is finding all of the currently living persons. To accomplish this use static member variable(s) to keep track of living people, and ensure that these variable(s) are updated by appropriate non-static functions. One technique is to use a member variable like

```
private static ArrayList<Person> thePeople;
```

Another is to use a class like

```
class PersonLink { private Person myPerson ; private PersonLink next ;  ...
```

and a static member variable of type `PersonLink`. Choose whatever you are comfortable with.

Additional Coding Requirements

All member variables **must be** `private`.

The `public` methods of the `Person` class must be exactly those described in Figure 1. You may add as many private methods and fields as you see fit.

The `PersonTester` class must contain a `public static void main()` method that runs your program. Content is otherwise up to you.

Method	*Meaning*
constructors	
`public Person(String n)`	Creates a living person with name `n`.
attributes	
`public Person murderer()`	Returns the `Person` that murdered this person. Returns `null` if this person has not been murdered.
`public String name()`	Returns this person's name. Should end with ", `deceased`" if the person is dead.
`public boolean isAlive()`	*obvious.*
actions	
`public void die()`	Causes `this` Person to die, unless they are already dead, when it has no effect.
`public void murder(Person victim)`	Causes `victim` to die, and the murderer to be known to the victim.
`public void sayHello()`	causes this `Person` to print "`Hello, I'm` ***`name`***`.`" on `System.out`.
class attributes	
`public static int numberLiving()`	
`public static int numberDead()`	
class actions	
`public static void allSayHello()`	causes every living person to say hello as described above.

Figure 1: Properties and Actions of `Persons`

```
1  /**
2   * This file is part of a solution to
3   *     CPSC 110 Lab 7 Problem 2 Winter 2010
4   *
5   * Implements a Roman numeral calculator
6   *
7   * @author David Casperson
8   * Student Number: 783030901
9   * @version 1
10  */
```

Figure 2: Format of file header comment

The `Person` class and the `PersonTester` class must be implemented in separate `.java` files.

Each (and every) `.java` file should have a header comment that looks substantially similar to that shown in Figure 2. In particular, the comment

- should be a `javadoc` comment (start with "`/**`"),
- should have an @author line,
- should have your student number, and
- should say why the file exists (lines 2–3)
- and what it does (line 5).

Checklist

Here are some things to check.

- Make sure that `die()` applied to a dead person doesn't cause the population to decrease.
- Make sure that a person's murderer's name prints correctly if the murderer herself is dead.
- Make sure that murdering a living person causes them to die.
- Decide what happens when either a murderer or a victim is already dead.