

## Static and non-static Members

---

### Due Date:

This assignment is due Friday, January 20 *by the beginning of lecture*.

Please submit via moodle [See there for more instructions on what to submit.](#)

---

### Persons

Here is a description of the code to implement:

#### The Person class

- ⇒ Implement a Person class that has all of the methods shown in Figure 1.
- More precisely, the public methods of your Person class must include all the methods shown and only the methods shown in Figure 1
  - In particular, the Person class does *not* have a main method.
  - In order to make the Person class work, you will need to add member variables, both static and *non-static*.  
**All member variables must be private.**
  - You may use additional private methods should you find it helpful.

#### The PersonTester class

- ⇒ Implement a PersonTester class whose public static void main(...) method tells a story about Persons. The story should exercise the methods of the Person class.
- A tiny PersonTester class is shown in Figure 3 on page 5. (The coding style has been contorted so that the code fits on one page.)
  - Your PersonTester should test more thoroughly.
  - Have fun inventing your story!

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

---

## More about Person methods

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.

---

## Coding Requirements

- All member variables **must be** private.
- The `Person` class and the `PersonTester` class must be implemented in separate `.java` files.
- The classes should be in package `lab2`. (If packages have not been covered, do your best.)

- 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).
- 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.

## 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.

---

```
1 // Put a proper file header comment here!
2 public class PersonTester
3 { private static void print (String s) {System.out.print(s) ;}
4   private static void println(String s) {System.out.println(s) ;}
5
6   private static void printCount()
7   {
8     final int livingCount = Person.numberLiving() ;
9     final int deadCount   = Person.numberDead() ;
10    final int totalCount  = livingCount + deadCount ;
11    print("The number of people is "+totalCount) ;
12    if (deadCount>0)
13      println(String.format(", but %d of them are dead!\n",deadCount)) ;
14    else println(".");
15  }
16
17  public static void main(String [] args)
18  {
19    Person bob = new Person ("Bob") ;
20    println("Bob says \"hi\"") ;
21    bob.sayHello() ;
22    printCount() ;
23
24    Person bill = new Person ("Bill") ;
25    println("Bill is born.") ;
26    bill.sayHello() ;
27    println("Everyone says hello") ;
28    Person.allSayHello() ;
29    printCount() ;
30
31    println("Bob dies.") ;
32    bob.die() ;
33    printCount() ;
34    Person.allSayHello() ;
35
36    Person sue = new Person("Sue") ;
37    println("Sue is born.") ;
38    sue.murder(bill) ;
39    Person.allSayHello() ;
40    printCount() ;
41    println("Oh Oh Oh!") ;
42    print(bill.name()+" is ") ;
43    println(bill.isAlive() ? "hiding." : "dead.") ;
44    if (bill.murderer()!=null)
45      {
46        println(bill.name()+" is murdered!") ;
47        println("The murderer of Bill is "+bill.murderer().name()+".  ") ;
48      }
49    print("\n\nOur exciting story ends suddenly.\n") ;
50  } }
```

---

Figure 3: A tiny cramped PersonTester class