Exercise 2

Methods, properties and constructors

By the end of this exercise you will be able to

- Define and call a *method*.
- Define and access a *property*.
- Define and call a *constructor*.
- Understand how a class without a constructor is automatically given a zero-argument constructor.

Introduction

A `class` in Java is like a C language struct with the difference that in addition to being able to hold data, Java classes can also contain program code. A class is comprised of *properties* and *methods*. Properties are identical to the data members of a C struct, and methods are the Java analogue of C functions. Unlike C functions, Java methods always reside within a class.

This exercise will get you used to using classes in Java by introducing you to a class `Person` that represents the concept of a human being. In order to use the `Person` class we must first create *instances* or *objects* of that class. In the example listing we create two such objects, one to represent Luke Skywalker and the other to represent Winston Peters.

If you look at the program listing now, you will see that the the first two lines of the `main` method say `new Person(...)`. These cause the *constructor* of the `Person` class to be called which is a special method with the job of creating the `Person` objects and setting the initial values of the properties to sensible values.

The information inside the brackets are given to the constructor like arguments to a function so that each object is constructed in its own way according to what kind of object it is supposed to represent. For example, the Luke Skywalker object is constructed using the string `"Luke Skywalker"` which is copied into the `name` property and the number `34` which is copied into the `age` property.

The expressions `ls` and `wp` are *references* which are like handles on the objects so that if we want to use one of the methods or properties of the `Person` class we can say which object we are referring to, eg: `ls` to refer to the Luke Skywalker object, `wp` to refer to the Winston Peters object.

Execution of the Java program starts at the `main` method, so that is where you will make most of your changes to the program code, except where you are asked to define a new method or property of `Person`, in which case it belongs inside the `Person` class.

If you understand this introduction then you are now ready to start using the object-oriented style of programing that Java follows!

Questions

NOTE: At the end of each question, you should make sure that your program still compiles and that when you run it, it behaves as expected.

Program listing

```
class Person {

   // Properties of the class...
   private String name;
   private int    age;

   // Constructor of the class...
   public Person(String aName, int anAge) {
      name = aName;
      age  = anAge;
   }

   // Methods of the class...
   public void talk() {
      System.out.println("Hi, my name's " + name);
      System.out.println("and my age is " + age);
      System.out.println();
   }
   public void commentAboutAge() {
      if (age < 5) {
         System.out.println("baby");
      }
      if (age == 5) {
         System.out.println("time to start school");
      }
   }

}

class PersonTest {

   // The main method is the point of entry into the program...
   public static void main(String[] args) {

      Person ls = new Person("Luke Skywalker",34);
      Person wp = new Person("Winston Peters",48);

      ls.talk();
      wp.talk();

   }

}
```

1. Fetch the file `PersonTest.java`, which contains the code from the program listing and load it into your editor.
2. Think of a new property that a person can have and add it to the `Person` class. You should be able to do this by copying the pattern of the other properties of the `Person` class. Four changes to the program listing are needed:
   (a) Add the property name and its type to the "properties" section of the class.
   (b) Add an extra parameter to the the constructor so that the property gets initialised correctly.
   (c) Modify the `main` method so that the Luke Skywalker and Winston Peters objects get the appropriate value of the new property.
   (d) Modify the `talk` method so that the new property is printed to the screen when the `talk` method is called.
3. Modify the `commentAboutAge` method to print the message "old person" if the person's age is greater than sixty.
4. In the `main` method add two calls to `commentAboutAge`, one for Luke Skywalker and one for Winston Peters.
5. Write a method `growOlder` that adds one year to the person's age. Make Winston Peters get older by one year by putting a call in the `main` method to do this.

   HINT: Look at the `talk` and the `commentAboutAge` methods to see how to define a method.
6. Write a method `giveKnighthood` that adds the word "Sir" to the beginning of a person's name. Then make Winston Peters become "Sir Winston Peters" by calling this method from the `main` method.

   HINT: This is much like question 5, except you are adding strings instead of numbers.
7. How is it possible to print the value of Luke Skywalker's age directly from inside the `main` method without calling the `talk` method?

   HINT: The `talk` method has a line of code to print the person's age. Copy this code to the `main` method, but since you are in another class from the `age` property, you will need to mention the name of the object in front of the `age` property, so `age` becomes `ls.age`.

   Also, you will need to change the definition of the `age` property from `private` to `public` to get this change to work. Why you need to do this will be covered in more detail in another exercise.
8. Delete the calls to `commentAboutAge` that you added in question 4, and instead put a call to `commentAboutAge` in the `talk` method. Note that when you are calling a method of the same class you are in, you don't have to put the object name in front of the method.

   Your program should generate similar output to what it did before.
9. Write a `growOlderBy` method which is similar to the `growOlder` method, except that it has an `int` parameter called `years` which says how many years the person should grow older by. In the `main` method, place a call to `growOlderBy` to make Luke Skywalker age by ten years.
10. Delete the `Person` constructor and replace the lines that create the Luke Skywalker and Winston Peters objects with the following:

```
Person ls = new Person();
Person wp = new Person();
```

    When a class doesn't have a constructor, the compiler automatically gives it a zero-argument constructor that sets all numeric values to zero and all reference values to `null`, meaning "no object". Run the program again to see what is output.