Exercise 4

References in more detail

By the end of this exercise you will be able to

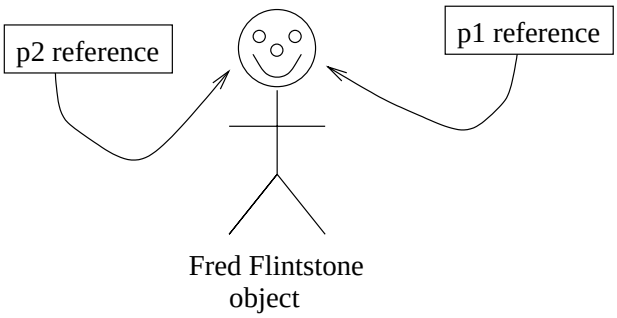
- Copy references so that two different reference variables refer to the same object.
- Pass object references from one method to another.
- Have one object referring in its properties to a different object.
- Understand how Java's dot operator can be applied several times in one statement.

Introduction

In this exercise, we look at references in more detail than we had looked at previously. You will be familiar with using references in the `main` method as a handle on the objects that were created. References are more powerful than this, for example you can copy references from one variable to another so that you can have more than one handle on the same object. The following code creates an object representing Fred Flintstone and two references `p1` and `p2` that are set to refer to the same Fred Flintstone object.

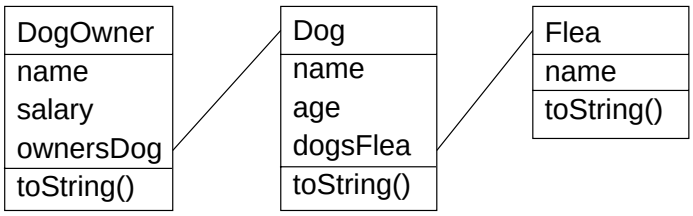
```
Person p1 = new Person("Fred Flintstone");
Person p2 = p1;
```

Here is a graphical representation of this relationship.



Also you can pass references as parameters to methods, so that a method can do work with an object that belongs to another class. Thirdly, you can store references as a property of an object. This enables you to set up complex relationships between objects.

In this example there are four classes involved: `Flea`, `Dog`, `DogOwner` and `DogTest`. If you look at the program listing now, you will see that the `DogOwner` class is missing. You will write this class as one of the questions of the exercise. The `Dog`, `Flea` and `DogOwner` classes represent the everyday concepts of dog, flea and dog owner. You will create instances of these classes and then set the values of the properties so that the `Dog` object refers to a `Flea` object (the dog has a flea) and the `DogOwner` object refers to a `Dog` object (the dog owner has a dog). Here is a UML diagram of the relationships between the three classes:



The `DogTest` class contains the `main` method so its job is to set up the objects as described above.

Questions

1. Fetch the file `DogTest.java`, which contains the code of the program listing.

2. In the `main` method of the `DogTest` class, write code to construct three fleas called Pop, Squeak and Zip. Look at the parameters of the `Flea` constructor to see how to create the `Flea` objects.

3. In the `main` method of the `DogTest` class, write code to construct three dogs called Rex, Jimbo and Fido. Note that the third parameter to the `Dog` class is of type `Flea`. Therefore you will need to supply a `Flea` reference for each dog. Make it so that Rex has a flea called Pop, Jimbo has a flea called Squeak, and Fido has a flea called Zip.

   HINT: If the flea called Pop is referenced by the variable name `p`, then this reference should appear as the third argument in one of the calls to the `Dog` constructor.

4. Write a `toString` method in the `Dog` class that works like the `toString` method in the `Flea` class. Then use this method to print out the full statistics of the three dogs that you have just created in question 3.

5. By copying the pattern of the `Flea` and `Dog` classes, write a class `DogOwner` that has three properties: `name`, `salary` and `ownersDog`. Also write a three-parameter constructor for the `DogOwner` class that sets these properties.

6. Add some code into the `main` method to construct three dog owners called Angus, Brian and Charles. Make it so that Angus has a dog called Rex, Brian has a dog called Jimbo, and Charles has a dog called Fido. Use the `Dog` references that you created in question 3 to achieve this.

7. Write a `toString` method for the class `DogOwner` and add some code to the `main` method to call it for Angus, Brian and Charles.

8. If your reference to the Angus object is called `a`, then what is the value of the following:

   ```
   a.ownersDog.dogsFlea.toString()
   ```

   Add some code to the `main` method to find out.

Program listing

```
class Flea {

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

   // Constructor of the class...
   public Flea(String aName) {
      name = aName;
   }

   // Methods of the class...
   public String toString() {
      return "I am a flea called " + name;
   }

}

class Dog {

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

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

}

class DogTest {

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

   }
}
```