**Lab 13**

Download the programs Employee.java and TestEmployee.java from Moodle.

Create separate packages for each exercise and copy the skeleton classes in each time.

**Exercise 1**

Write Java code to sort the employees by age in descending order. Implement the Comparable interface in order to do this.

Sample Output:

First Name: Noel Last Name: Brady Age:69

First Name: John Last Name: Donnell Age:34

First Name: Tony Last Name: Collins Age:18

First Name: Jim Last Name: Greally Age:16

**Exercise 2**

Write the Java code to sort the employees by surname in alphabetical order. Create a separate class called NameComparator to do the sort

Sample Output:

First Name: Noel Last Name: Brady Age:69

First Name: Tony Last Name: Collins Age:18

First Name: John Last Name: Donnell Age:34

First Name: Jim Last Name: Greally Age:16

**Exercise 3**

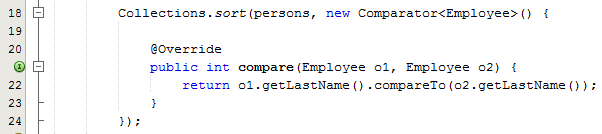
Write the Java code to sort the employees by surname in alphabetical order. This time use the Employee class to implement the Comparator interface.

**Exercise 4**

Sort again like exercise 3 but use an anonymous inner class this time – see slide 20 of the notes for guidance

**Exercise 5**

We could also implement the anonymous class in our test class as follows. Try this by putting the following code in the test class..



**Exercise 6 – Static Nested Classes**

You are required to create a class MagicHat that will define an array called rabbits, containing a variable number of Rabbit objects. You will put the definition for the class Rabbit inside the definition of the class MagicHat. The basic structure of MagicHat.java is shown below – create this structure first. Sample output is also shown below:

public class MagicHat {

// Definition of the MagicHat class...

// Nested class to define a rabbit

static class Rabbit {

// Definition of the Rabbit class...

}

}

**Sample Output**

Gray Hat contains:

Jack1 Fred1 Jack2 Jack3 Joe1

Black Hat contains:

Joe2 Jack4

Red Hat contains:

Fred2 Joe3 Joe4

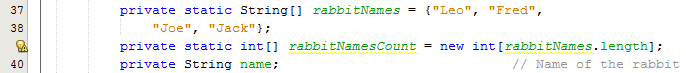
The MagicHat class requires 4 variables:

* Name of the hat (String)
* An array of type Rabbit to hold the rabbit objects
* A static int to hold the maximum number of rabbits (5)
* A static int to hold a random number of rabbits

Next you will build the static class Rabbit

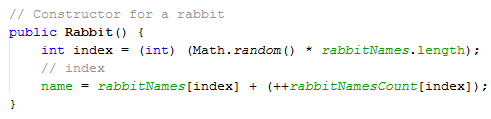
This class has 3 members:

* A static String array to hold the names allowed for a rabbit
* A static int array to hold the number id for a rabbit
* Name of the rabbit (String)

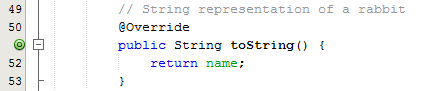


Constructor

Initialised the rabbit name using the two static arrays:

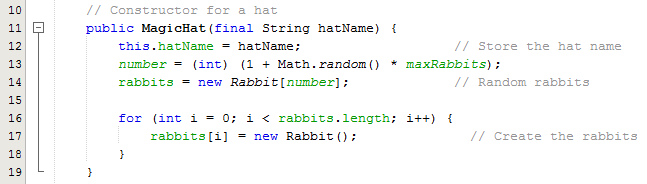


toString method

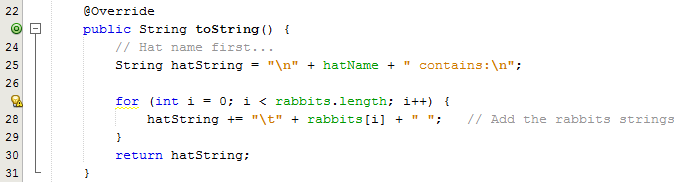


Now we go back to the MagicHat class

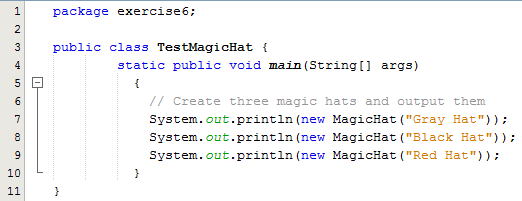
Constructor



toString method



TestMagicHat



**Explanation**

Each MagicHat object will contain a random number of Rabbit objects. The constructor for a MagicHat object stores the name of the hat in its private member hatName, and generates a Rabbit array with at least one, and up to maxRabbits elements. The array created is then filled with Rabbit objects.

The MagicHat class also has a method toString() method which returns a String object containing the name of the hat and the names of all the rabbits in the hat. This assumes that the Rabbit class also has a toString() method defined. We will be able to use the toString() implicitly in an output statement when we come to create and display MagicHat class objects.

The base names that we use to generate rabbit names are defined in the static array rabbitNames[] in the Rabbit class. The count for each base name, which we will append to the base name to produce a unique name for a rabbit, is stored in the static array rabbitNamesCount[]. This has the same number of elements as the rabbitNames array, and each element stores a value to be appended to the corresponding name in the rabbitNames array. The Rabbit class has the data member, name, to store a name that is initialized in the constructor. A random base name is selected from the rabbitNames[] array using an index value from 0 up to one less than the length of this array. We then append the current count for the name incremented by 1, so successive uses of any base name such as Gnasher, for example, will produce names Gnasher1, Gnasher2, and so on. The toString() method for the class returns the name for the Rabbit object.

The method main() in TestMagicHat creates three MagicHat objects and outputs the string representation of each of them. Putting the object as an argument to the println() method will call the toString() method for the object automatically, and the String object that is returned will be output to the screen.