**CE00527-5 Further Object Oriented Programming**
**Tutorial Week 6**

In this tutorial you will

- override methods specified in the Object class.
- create a collection of DVDs and use JUnit to investigate some classes and methods of the Java Collections framework.

In Tutorial 5 you created and tested a system which used DVD and Person objects. You need to have finished Questions 4 and 5 of Tutorial 5 before attempting the exercises in this Tutorial.

Your DVDTest class should have an attribute d1 of type DVD. In the setup() method, d1 should be constructed with the following attributes:

- Title: Inception
- Lead Actor: A Person with first name “Leonardo”, last name “DiCaprio”
- Number of stars: 5

## Part 1 Overriding Object methods – Basic Exercise

In this section we will continue to develop the system by overriding some Object methods of the DVD and Person classes.

1. Add the following test to your DVDTest class:

```
@Test
public void testEquals() {
    Person p2 = new Person("Leonardo", "DiCaprio");
    DVD d2 = new DVD("Inception", p2, 5);
    DVD d3 = new DVD("Gone with the Wind", p2, 4);
    assertEquals(d2, d1);
    assertNotSame(d2, d1);
    assertEquals(d1, d1);
    assertSame(d1, d1);
    assertFalse(d3.equals(d1));
}
```

As you have not yet overridden the default Object equals() method, this test should fail. Can you explain why?
Override the equals() method in both Person and DVD so that the test passes. Your equals() methods should test all attributes for equality.

2. Now test the default hashCode() methods of DVD and Person using the test below. The Object hashCode() method returns the memory location of the object.

```
@Test
public void testHashCode() {
    Person p2 = new Person("Leonardo", "DiCaprio");
    DVD d2 = new DVD("Inception", p2, 5);
    assertEquals(d2.hashCode(), d1.hashCode());
    assertEquals(p2.hashCode(), d1.getLeadActor().hashCode());
}
```

Again this test should fail – why? What relationship should the hashCode() method have to the equals() method? Why does this test method not include a test of whether two non-equal DVD objects have different hashcodes?

Override the Object hashCode() method in Person and DVD so that the test passes. You may use NetBeans to autogenerate a hashCode() method as follows:

> Right-click anywhere within code for the class and choose **Insert Code** from the pop-up menu. Choose to generate **hashCode()** and tick all the fields to be included.

3. Implement the Cloneable interface in the DVD and Person classes. To start with, write a default clone() method in each class which simply calls the superclass (Object) clone() method (you can autogenerate this in NetBeans).

Add the following test to your DVDTest class

```
 @Test
public void testClone() throws CloneNotSupportedException {
    DVD d2 = (DVD)d1.clone();
    assertEquals(d2, d1);
    assertNotSame(d2, d1);
}
```

Verify that the test passes.

Add the lines in bold to the testClone() method of DVDTest:

```
@Test
public void testClone() throws CloneNotSupportedException {
    DVD d2 = (DVD)d1.clone();
    assertEquals(d2, d1);
    assertNotSame(d2, d1);
    // change the lead actor of the original DVD object
     Person p1 = d1.getLeadActor();
     p1.setFirstName("Clark");
     p1.setLastName("Gable");
     assertFalse(d2.equals(d1));
     assertEquals("Clark Gable", d1.getLeadActor().toString());
     assertEquals("Leonardo DiCaprio", d2.getLeadActor().toString());
}
```

The test should now fail – the objects d1 and d2 are still equal, even though the name of the leadActor object associated with d1 has been changed. Why is this so? You might want to use some additional assertions to test your hypothesis.

The default Object clone() method is a shallow clone. Override it in the DVD class to implement a deep clone. Run the tests again. What is the difference between a shallow and a deep clone?

Note: In this section we have tested the equals(), hashCode() and clone() methods of Person in DVDTest, while testing the equals(), hashCode() and clone() methods of DVD. Normally we would write separate unit tests of these methods within the PersonTest class, to ensure these methods work as expected independently of any class (such as DVD) which makes use of them.

4. In this question you will implement the Comparable interface in the Person and DVD classes.

First add a method called testCompareTo() in your PersonTest class, which verifies that the Person compareTo() methods orders Person objects by lastName, then firstName. The method should return 0 if the two Person objects hold the same first and last names.

Now implement the Comparable interface in the Person class so that the test passes.

Do the same (write a test method, then implement Comparable to pass the test) for the DVD class. The DVDs should be ordered by title, then lead actor, then number of stars (highest first).

**For Part 1, put in your portfolio:**
**the code listings for your final versions of the DVD and Person equals(), hashcode(), clone() and compareTo() methods**
**Explain in your portfolio, with reference to the DVD class:**
Why you must override the hashCode() method if you override the equals() method
What the relationship the hashCode() method should have to the equals() method
The difference between a shallow and a deep clone

**Part 2 Collections of DVDs – Advanced Exercise**

In this section we will use JUnit to investigate some classes and methods of the Java Collections framework.

We will use the DVD/Person system developed in Tutorial 5 and Part 1 of this tutorial – you need to have completed these exercises to attempt this section.

5. Right click on the package which holds the project's test classes. Select new -> other -> JUnit -> JUnit Test. Call the class DVDCollectionTest. This will create a skeleton test class with the necessary imports.

   Note that you could have used this wizard to create a Test for Existing Class, or a Test Suite (a class which runs a suite of Test Classes)

   Add the following attribute to DVDCollectionTest

   ```
   private ArrayList<DVD> theDVDs;
   ```

   In the setUp() method, set up the test fixture so that theDVDs holds DVDs with the following values

   Person class:

| Person Object Reference | First name | Last name |
|---|---|---|
| p1 | Leonardo | DiCaprio |
| p2 | Fay | Wray |
| p3 | Naomi | Watts |
| p4 | Cary | Grant |

   DVD class

| Position in theDVDs | Title | Number of Stars | Person object reference |
|---|---|---|---|
| 0 | Inception | 5 | p1 |
| 1 | King Kong | 5 | p2 |
| 2 | King Kong | 4 | p3 |

| | | | |
|---|---|---|---|
| 3 | Indiscreet | 3 | p4 |
| 4 | Ellie Parker | 4 | p3 |

Write a test method that verifies that theDVDs holds the correct information in the correct order. Your test should call toString() on each element in the in turn, verifying that an expected String equals the actual String returned. It should also print out the list in order. Run the test and make sure it passes.

6. Add a method testShuffle() to the DVDCollectionTest class. This test should call the Collections.shuffle() static method, with theDVDs as a parameter, and print out the list in the new order. You do not need to use any assertion methods in testShuffle().

7. Add a method testSort()to the DVDCollectionTest class. This test should call the Collections.sort() static method, with theDVDs as a parameter, and verify that the ArrayList has been sorted in order of title, then actor, and print out the list in the new order. Run the test and make sure it passes.

8. Add a new class to the DVD project called DVDComparator, which implements the Comparator interface. This compare() method of this class should order DVDs by the number of stars (highest first), followed by the lead actor and then the title.

   Add a method testSortByStars() to the DVDCollectionTest class. This test should call the Collections.sort() static method, with theDVDs and a DVD comparator object as parameters, and verify that the ArrayList has been sorted in order of number of stars (highest first), then lead actor, then title. The test should also print out the list in the new order. Run the test and make sure it passes.

**For Part 2, put in your portfolio:**

**the code listings for the DVDCollectionTest and DVDComparator classes**

**the output from running the DVDCollectionTest class**