Laboratório 14 - Herança e Polimorfismo II (21/10/2014)

1. .

```
package lab14;

interface Worker {
	public boolean startWork();
	public boolean stopWork();
}
```

```
package lab14;

public class Employee implements Worker, Displayable {
	private String name;
	private String company;
	private static boolean working = false;  //Quando o objeto é criado ainda não
começou a trabalhar
	private static boolean stopWorking = false;
	private static boolean startWorking = false;

	public Employee(String empName, String empCompany) { //Construtor
	   this.name = empName;
	   this.company = empCompany;
	}

	@Override
	public boolean startWork() {
	   if (working == false)
	   {
	       working = true;  // O empregado começou a trabalhar
	       startWorking = true;
	   }
	   else
	       startWorking = false;
	   return startWorking;
	}

	@Override
	public boolean stopWork() {
	   if (working == true)
	   {
	       stopWorking = true; // O empregado deixa de trabalhar
	       working = false;
	   }
	   else                    // se não estava a trabalhar não irá parar
	       stopWorking = false;

	   return stopWorking;
	}

	/**
	 * @return name of the employee
	 */
	public String getName() {
	   return name;
	}

	/**
	 * @return company where the employee works
```

```
         */
        public String getCompany() {
               return company;
        }

        @Override
        public String display() {
           String display = "[" + name + "," + company + "]";
           return display;
        }
    }
```

```
package lab14;

import static org.junit.Assert.*;

import org.junit.Assert;
import org.junit.Test;

public class EmployeeTest1 {

       private static final String EMP_NAME = "nome do empregado";
       private static final String EMP_COMPANY = "nome da empresa";

       @Test
       public void testWorkerOperations()
       {
          Worker worker = new Employee(EMP_NAME, EMP_COMPANY);
          // Attention: At this point, the employee has not started
          // working yet!
          Assert.assertFalse(worker.stopWork()); // Should be false.
          Assert.assertTrue(worker.startWork()); // Should be true.
          Assert.assertFalse(worker.startWork()); // Should be false.
          Assert.assertTrue(worker.stopWork()); // Should be true.
          Assert.assertFalse(worker.stopWork()); // Should be false.
       }

       @Test
       public void testDisplayableOperations()
       {
          Displayable displayable = new Employee(EMP_NAME,EMP_COMPANY);
          Assert.assertEquals("[" + EMP_NAME + "," + EMP_COMPANY + "]",
       displayable.display());
       }

       @Test
       public void testEmployeeOperations()
       {
          Employee emp = new Employee(EMP_NAME, EMP_COMPANY);
          Displayable displayable = new Employee(EMP_NAME,EMP_COMPANY);
          // Attention: At this point, the employee has not started
          // working yet!
          Assert.assertFalse(emp.stopWork()); // Should be false.
          Assert.assertTrue(emp.startWork()); // Should be true.
          Assert.assertFalse(emp.startWork()); // Should be false.
          Assert.assertTrue(emp.stopWork()); // Should be true.
          Assert.assertFalse(emp.stopWork()); // Should be false.
```

```
            Assert.assertEquals("[" + EMP_NAME + "," + EMP_COMPANY + "]",
        displayable.display());
            Assert.assertEquals(EMP_NAME, emp.getName());
            Assert.assertEquals(EMP_COMPANY, emp.getCompany());
        }
    }
```