Classes, Methods, Interfaces

CSC 210 Practice Exercises

Concept questions

1. What is the best practice for declaring instance variables?
2. What differentiates an instance variable from a class variable?
3. What are the keywords to declare a class constant?
4. What's overloading?
5. What do we call a method with the same name as the class and with no return type

JUnit

Given the class below, write a test class with the following JUnit assertions:

- `Assert.assertEquals()`
- `Assert.assertNotNull()`
- `Assert.assertNull()`
- `Assert.assertTrue()`

```
public class MyClass {

    private String x;
    private int y;

    public boolean isLonger(int size) {
      return x.length() > size;
    }

    public void setX(String x) {
        this.x = x;
    }
```

```
    public void setY(int y) {
        this.y = y;
    }

    public String getX() {
        return x;
    }

    public int getY() {
        return y;
    }

}
```

Here's what you need for import statements and class set up:

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class TestMyClass {

    @Test
    void test() {
        // write your assertion here
    }

}
```

Interfaces

Implement a class for the following interface:

```
public interface Bicycle {
    void changeCadence(int newValue);

    void changeGear(int newValue);

    void speedUp(int increment);
```

```
    void applyBrakes(int decrement);

}
```

Answers

Concept questions

1. declare them `private` and write getter and setter methods
2. class variables are declared `static`
3. `final static`
4. Using the same name for two methods or more methods, with different signatures (the parameters they take are different)
5. A constructor

JUnit

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class TestMyClass {

    @Test
    void testEquals() {
        MyClass objectOne = new MyClass();
        objectOne.setX("something");
        assertEquals("something", objectOne.getX());
    }

    @Test
    void testNull() {
        MyClass objectOne = new MyClass();
        assertNull(objectOne.getX());
    }

    @Test
    void testNotNull() {
        MyClass objectOne = new MyClass();
```

```
        objectOne.setX("something");
        assertNotNull(objectOne.getX());
    }

    @Test
    void testTrue() {
        MyClass objectOne = new MyClass();
        objectOne.setX("something");
        assertTrue(objectOne.isLonger(2));
    }

}
```

Interfaces

```
public class ACMEBicycle implements Bicycle {
    int cadence = 0;
    int speed = 0;
    int gear = 1;

   // The compiler will now require that methods
   // changeCadence, changeGear, speedUp, and applyBrakes
   // all be implemented. Compilation will fail if those
   // methods are missing from this class.

    public void changeCadence(int newValue) {
         cadence = newValue;
    }

    public void changeGear(int newValue) {
         gear = newValue;
    }

    public void speedUp(int increment) {
         speed = speed + increment;
    }

    public void applyBrakes(int decrement) {
         speed = speed - decrement;
    }
```

```
    public void printStates() {
         System.out.println("cadence:" +
             cadence + " speed:" +
             speed + " gear:" + gear);
    }

}
```