ITI 1121. Introduction to Computer Science II

Laboratory 2

Winter 2014

Part I
Prerequisite material

Objectives

- Manipulating arrays and references
- Understanding when to use "==" or "equals"

1 FindAndReplace

Complete the implementation of the (static) class method **String[] findAndReplace(String[] in, String[] what, String[] with)** of the class **Utils** (Utils.java). The method returns a copy of the array **in** where each word occurring in the array **what** has been replaced by the word occurring at the corresponding position in the array **with**. The array designated by **in** must remain unchanged.

Later in the semester, we will learn about exceptions. Exceptions provide tools for handling error situations. In the meantime, the method **findAndReplace** returns **null** whenever the preconditions of the methods are violated. In particular, the formal parameters cannot be **null**. For all three arrays, none of the elements can be **null**. The query and replacement arrays must be of the same length.

Part II
Unit testing: JUnit

Objectives

- Introduction to unit testing

Although "testing shows the presence, not the absence of bugs"[1] , testing is never the less an important activity of software development. JUnit is a Java framework to assist you with this important task.

Your teaching assistant will now present a brief introduction and tour of JUnit. She/he will lead a discussion around each test case found in **TestFindAndReplace.java**: what is the test case testing, what are the strengths and weaknesses of the test set, what cases are missing, etc.

- TestAll.java
- TestFindAndReplace.java

Ah, you are back! Make all the necessary changes to your method **findAndReplace** so that it passes all the tests.

Part III
More objects

Objectives

- Further understanding of object-oriented programming
- Problem solving using Java

2 Rational

Implement a class to represent rational numbers. Each rational number consists of a **numerator** and a **denominator**, both of type **int**. Since each rational number has its own **numerator** and **denominator**, these must be instance variables. Furthermore, good object-oriented programming suggests that the visibility of the variables should be private.

3 Constructors

The class **Rational** has two constructors. One of them has two formal parameters, which provide the initial values for both instance variables. The other constructor has a single parameter, which provides the initial value for the numerator; the denominator is assumed to be 1.

4 getters

Implement access methods that return the numerator and denominator of this rational, but no setter methods. An object that has no setter methods, and no other methods for transforming the **state** of the object, is said to be **immutable**. **Immutable** is a great property. Do you see why? Discuss this with your neighbors and TA.

5 plus

Implement the instance method **plus**. The method has a single formal parameter, of type **Rational**. The method returns a **new Rational** number that is the sum of this number and that of the parameter.

6 plus

Implement a class method **plus**. The method has two formal parameters, both of type **Rational**. The method returns a **new Rational** number that is the sum of the two numbers.

7 gcd

Implement a private class method for calculating the greatest common divisor of two integers, which are the formal parameters of the method.

8 reduce

Implement a private instance method called **reduce** that transforms this number into its reduced form.

9 reduce

Make all the necessary changes so that a rational number is always stored in reduced form.

10 equals

Implement the instance method **public boolean equals(Rational o)** that returns **true** if this fraction and the one designated by **o** represent the same fraction (content equality).

11 toString

Add a method **public String toString()** that returns a **String** representation of this fraction with the numerator followed by the symbol "/", followed by the denominator. If the denominator is 1 then the method returns a **String** consisting of the numerator only.

12 compareTo

Implements the instance method **int compareTo(Rational o)**. It compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Part IV
Documentation: JavaDoc

Objectives

- Introduction to JavaDoc

Another important aspect of software development is the documentation. JavaDoc is a format for your Java comments, and a set of tools for producing Web pages automatically. In ITI1121, we are asking you to document your code (variables, methods, etc.) using JavaDoc. Your teaching assistant will now present a brief introduction of JavaDoc.

You must add JavaDoc comments for the class Rational.

1. Add JavaDoc comments for all the methods. Each comment should include a brief description of what the method does and descriptions of the parameters and the return value using JavaDoc format.
2. Add a brief description of the class Rational, using the JavaDoc syntax, make sure to include the name of the author of the class (you).

You can now produce HTML files automatically, either using your favorite IDE (DrJava, Eclipse, Netbeans, etc.) or by running the **javadoc** command in a shell, the parameters are -d doc, followed by the name(s) of the Java file(s) to be processed

```
> javadoc -d doc Rational.java
```

When several files need to be processed simultaneously, use `*` in place of the names of the files.

```
> javadoc -d doc *
```

Part V
Quiz (1 mark)

- Referring to the class **Counter** below, the keyword **this** can be omitted without affecting the compilation or execution of the program.
  True or **False**

```
public class Counter {
    private int value = 0;
    public Counter( int initialValue ) {
        this.value = initialValue;
    }
}
```

Submit your answer on Blackboard Learn:

- https://uottawa.blackboard.com/

Last Modified: May 15, 2015