COE808 Lab2

Prelab preparation

Before coming to the lab you should:

1. Read the lab. The most recent version can be found at the URL:
   www.ee.ryerson.ca/~courses/coe808
2. Try to prepare any questions you may have about the lab.

To get started you must perform the following operations.

1. Create your lab2 sub-directory with the command:
   mkdir lab2
2. Change to your lab2 directory with the command:
   cd lab2

In this lab you must use Netbeans as IDE

Netbeans tutorial: http://netbeans.org/kb/docs/java/quickstart.html

Exercise 1 (40 marks)

Goals for this exercise:

1. Explore some of the properties of the pre-defined String class.
2. Read lines of keyboard input with a Scanner object.

Strings: Strings are represented as objects of the String class, which is defined in the java.lang package. Strings can be created in several ways, of which we consider two:

```
String name = "Jean Valjean";
```

or by using a constructor in the String class that takes a string as a parameter and creates a unique String object with the same value. For example:

```
String  protagonist = new String(name);
```

Now, both variables *name* and *protagonist* are references to String objects which have the same value, "Jean Valjean".

In this exercise, we wish to explore the common, very useful String instance methods described in the table below. The String object to which the method is being applied is referred to as *this String* in the descriptions:

METHOD	DESCRIPTION	EXAMPLE
Boolean equals(String string2)	returns true or false depending upon whether this String and the String parameter have the same value	String x = "123", y= "12" + "3"; x.equals(y) returns the value true.
int length( )	returns the length of this String object	String greeting = "Bonjour"; greeting.length( ) returns the value 7.
char charAt(int position)	returns the character (type char) in this String at index position. Position is a non-negative integer less than the length of this String.	String greeting = "Bonjour"; greeting.charAt(5) returns the value 'u'
int indexOf(String string2)	returns the index of the first occurrence of the string string2 in this String object. Returns -1 if string2 not found.	String greeting = "Bonjour", word = "jou"; greeting.indexOf(word) returns 3.
String substring(int start)	returns the substring of this String object starting from start through to the end of the String object.	String name = "Sheryl Crow"; name.substring(7) returns the string "Crow"
String substring(int start, int end)	returns the substring of this String object starting from start through, *but not including*, index end of this String object.	String name = "Sheryl Crow"; name.substring(3,8) returns the string "ryl C"
String toUpperCase()	return this String converted to upper case.	"Hi Jane!".toUpperCase() returns the string "HI JANE!"
String toLowerCase()	return this String converted to lower case.	"Hi Jane!".toLowerCase() returns the string "hi jane!"

Part I

1. In Netbeans, save your program as "**Project1**" on your lab2 directory. Name your new class "StringDemo**"** and save it to your directory.

```
import java.util.Scanner;
/**
 * Here we will explore some of the methods of the String class.
 * We will distinguish == and equals.
 * We will also use the Scanner class to read a whole line of input.
 */
public class StringDemo
{
   public static void main (String[] args){
      String str1 = "It was the best of times.";
```

```
        String str2 = str1;
        String str3 = "It was the best of times.";
        String str4 = "Of times it was the best.";
        System.out.println("str1=" + str1);
        System.out.println("str2=" + str2);
        System.out.println("str3=" + str3);
        System.out.println("str4=" + str4);
        System.out.println();
        System.out.println("The length of str1 is " + str1.length());
        System.out.println("Index of 'b' in str1 is: " + str1.indexOf('b'));
        System.out.println("Index of 'x' in str1 is: " + str1.indexOf('x'));
        System.out.print("The substring of str1 from ");
        System.out.println("position 7 to position 14 is: " + str1.substring(7,15));
        System.out.println("str1 in upper case is: " + str1.toUpperCase());
        if (str1==str2)
           System.out.println("str1 and str2 refer to the same object.");
        if (str1!=str3)
           System.out.println("str1 and str3 do NOT refer to the same object.");
        if (str1.equals(str3))
           System.out.println("str1 and str3 contain the same characters.");
        if (!str1.equals(str4))
           System.out.println("str1 and str4 do NOT contain the same characters.");

        Scanner in = new Scanner(System.in);
        System.out.println("Enter a line:");
        String user1 = in.nextLine();
        System.out.println("Enter a second line:");
        String user2 = in.nextLine();
        if (user1.equals(user2))
           System.out.println("Your strings are the same.");
        // An object returned can be immediately used to call another method
        // or be used as a parameter:
        else if (user1.toUpperCase().equals(user2.toUpperCase()))
           System.out.println(" If you ignore case, your strings are the same.");
        else
           System.out.println("Your strings are different.");
        System.out.println("Done");
    }// end of main method
} // end of class
```

Compile and run the main program, type the two lines of input requested into the terminal window (remembering to press the Enter key both times), and analyze the output. Try to understand the effect of each method call.

Part II Design, compile and run Java programs to accomplish each of the following tasks. The program can just include a main method like the program in part I. In fact you can modify the program in part I if you like. Each part can be in a separate method or you can put it all in the main program. Be sure to include *prompts for each line of input requested*:

(A) Read a sentence (string) from a line of input, and print whether it represents a *declarative* sentence (i.e. ending in a period), *interrogatory* sentence (ending in a

question mark), or an *exclamation* (ending in exclamation point) or is *not a sentence* (anything else).

(B) Read a name from a line of input (first and last separated by a space) and print last name first followed by a comma followed by the first name. For example, if the input is "Marcel Proust", the output is "Proust, Marcel".

(C) Read a string and print the characters in *reverse order*. For example, if the input string is "Mine is a sad tale, said the mouse.", the output string would be ".esuom eht dias ,elat das a si eniM".

Exercise 2 (30 marks)

In Netbean, save your program as "**Project2**" on your lab2 directory. Create new classes. Save these classes to your directory and study it to see how it works.

Consider the following hierarchy of classes, with various details to be filled in:

```
// Person.java: abstract base class
public abstract class Person {
  public abstract double getWages(); // abstract, so omit body
 }
```

```
// Employee.java: the class Employee
public class Employee extends Person {
     protected String name; // in the form "Last, First"
     protected String address;
     // constructor
       public Employee(String n, String a) { // first name, then address
         // Fill in for Part a.
       }
     // convert the point into a String representation
        public String toString() {
            return "[" + name + ", " + address + "]";
       }
       public double getWages() { // must be either faculty or staff for wages
         return 0.0;
       }
     }
```

```
// Faculty.java: the class Faculty
public class Faculty extends Employee {  // inherits from Employee
```

```
  protected double salary;

  // Constructor
  public Faculty(double s, String n, String a) { // salary, name, address
    // Fill in for Part b.
  }

  // convert the Faculty class to a String
  public String toString() {
    // Fill in for Part c.
  }

  public double getWages() {
    return salary;
  }
}
```

```
// Staff.java: the class Staff
public class Staff
  // Fill in for Parts d., e., f.
}
```

```
// Persons.java: test the Person-Employee-Faculty-Staff  hierarchy
public class Persons {
  public static void main (String[] args) {
    Person[] persons = new Person[3];

    persons[0] = // Fill in for Part g.
    persons[1] = // Fill in for Part g.
    persons[2] = // Fill in for Part g.

    printPersons(persons);
    System.out.println();
  }
  public static void printPersons(Person[] s) {
    for (int i = 0; i < s.length; i++)
      System.out.println(s[i]);
  }
}
```

a. Fill in the constructor for **Employee** above.
b. Fill in the constructor for **Faculty** above, using the **Employee** constructor with **super**.

c. Fill in the **toString** method for **Faculty** above, using the **Employee toString** with **super**.
d. Write a **Staff** class that inherits from **Employee** (that is, extends **Employee**). The **Staff** class should have data fields **double hourlyRate** and **int hoursWorked**. getWages method of the Staff class will return hourlyRate*hoursWorked.
e. Write a constructor for **Staff** that takes parameters the two new ones above, followed by the name and address.
f. Write a **toString** method for **Staff** above, using the Employee **toString** with **super**.
g. In the **Persons** class above, create new objects of each of the three kinds:
    - **Employee** (with name "Nobody, Nonce", and address "120 Worry Lane")
    - **Faculty** (with salary 2020, name "Blow, Joe", and address "222 Center St.")
    - **Staff** (with hourly rate 7.5, hours worked 40, name "Bonkers, Bruce", and address "1 Prime Ave").

Exercise 3 (30 marks)

Consider the grammar

<P> → <S>$ (P is the start symbol)
<S> → b<M>b
<M> → (<L>
<M> → a
<L> → <M> a)

Design a recursive descent parser for this grammar. In Netbeans, save your program as "**Project3**" on your lab2 directory. The parser class must be named as “GrammarTest”. The example sentence to be parsed is
b ((a a) a) b $

Submitting your lab

(1) Zip your lab2 assignment within your coe808 directory
zip –r lab2 lab2

(2) Attach your lab1.zip file and send e-mail to GA, Mr. Naimul Mefraz Khan, subject of your e-mail must be lab1 followed by your name and student id.
e-mail: n77khan@ryerson.ca