Taller de Lenguajes II
Práctica nº 3 – A

Temas : Tipos de datos, Constructores, Sobreescritura. Clase String

1. **IMC.** Escriba un programa que calcule el "Indice de Masa Corporal" (IMC) de una persona. Este índice pretende determinar el intervalo de peso saludable que debería tener una persona. Se suele establecer un intervalo de 18.0 a 25.0 como saludable en personas adultas.
    a. Escriba una clase IMC sin implementar un constructor explícito, y agregue 2 variables de instancia que representan peso y altura respectivamente.
    b. Escriba los getters y setters para sus variables de instancia
    c. Escriba un método llamado "calculo" que realiza la siguiente cuenta y devuelve el valor de IMC.:

        imc = peso/(altura^2)

    d. Escriba una clase TestIMC donde pruebe el funcionamiento de su código.
        i. En su método main, recibirá como argumentos un peso y una altura (valores con decimales). Use una instancia de la clase IMC para realizar el cálculo. En base al resultado obtenido muestra: "XX.YY - Ud. parece saludable!" si el valor resultante se encuentra comprendido en el intervalo, caso contrario indica si se está por debajo ó por encima del IMC.
    e. Modifique la clase TestIMC para que permita ingresar por teclado los 2 valores que necesita para el cálculo.
    f. Agregue un constructor a la clase IMC que reciba el peso y la altura. ¿Su código compila correctamente? **JUSTIFIQUE**
    g. ¿Es posible escribir un método llamado "calculo" que realiza lo mismo que el punto c pero recibiendo 2 argumentos de tipo long?
    h. ¿Cómo se llama el mecanismo que permite a una clase tener dos métodos con el mismo nombre pero que difieren en su firma?

2. **Cadena de Constructores.** Analice el siguiente código y responda

```
// archivo CadenaDeConstructores.java
class CadenaDeConstructores {
    public static void main(String[] args) {
        Hijo h = new Hijo();
    }
}

// archivo Hijo.java
class Hijo extends Padre {
    Hijo() {
        System.out.println("Constructor Hijo()");
    }
}

// archivo Padre.java
```

```
class Padre extends Abuelo{
    Padre(int x) {
        System.out.println("Constructor Padre(" + x + ") ");
    }
}

// archivo Abuelo.java
class Abuelo{
    Abuelo() {
        System.out.println("Constructor Abuelo()");
    }
}
```

a. Verifique que compila. En caso de aparecer errores corríjalos de modo que compile exitosamente.
b. ¿Qué imprime la ejecución de la clase CadenaDeConstructores?
c. ¿Dónde se encuentran estas llamadas sucesivas que forman la cadena de constructores?

3. **Uso de literales String, clases String, StringBuffer y StringBuilder.** En caso de ser necesario lea la siguiente información relacionada al manejo de literals String y String:

- http://java67.blogspot.com.ar/2014/08/difference-between-string-literal-and-new-String-object-Java.html

a. Cree un paquete llamado **unlp.info.comparacionstring**

Escriba el siguiente código y ejecútelo.

```
package unlp.info.comparacionstring;
public class StringDemo {
   public static void main(String[] args) {
          String str1="Leones y Tigres y Osos!";
          String str2="Leones y Tigres y Osos!";
          String str3=str2;
          String str4=new String("Leones y Tigres y Osos!");
          String str5=" Y yo!";
          String str6="Leones y Tigres y Osos! Y yo!";
          String str7= str1 + str5;
          System.out.println(str1==str2);
          System.out.println(str1==str3);
          System.out.println(str1==str4);
          System.out.println(str2==str3);
          System.out.println(str2==str4);
          System.out.println(str3==str4);
          System.out.println(str6==str7);
          System.out.println(str1.equals(str4));
          System.out.println(str6.equals(str7));
   }
}
```

Sentencia	true/false	¿Por qué? - JUSTIFIQUE
str1 == str2		
str1 == str3		
str1 == str4		

str2 == str3		
str2 == str4		
str3 == str4		
str6 == str7		
str1.equals(str4)		
str6.equals(str7)		

b. ¿Qué hace el método **equals** de la clase String? (puede observar la implementación adjuntando los archivos fuente src.zip del JDK1.6+)

c. Suponga que cuenta con una clase Persona que modela a las personas del mundo real.
   i. ¿Considera que sería interesante hacer un "override" (sobreescritura) del método equals? Si la respuesta es afirmativa, indique el criterio de comparación, caso contrario **JUSTIFIQUE**.
   ii. En caso de no sobreescribir el método equals, ¿cuál es el criterio por default en Java para comparar dos (2) personas?

d. Descargue del sitio de la cátedra el archivo TestString.java.
   i. Cree un proyecto en eclipse e importe TestString.java
   ii. Ejecute la clase TestString
      1. Indique los resultados obtenidos
      2. **JUSTIFIQUE** los resultados obtenidos. Para justificarlos puede revisar la teoría, verificar como están implementadas esas operaciones ó acceder a alguna de las siguientes URLs:

- http://stackoverflow.com/questions/2971315/string-stringbuffer-and-stringbuilder
- http://www.java-tips.org/java-se-tips-100019/24-java-lang/2009-difference-between-string-stringbuffer-and-stringbuilder.html

Recuerde que tiene disponible el código fuente de las clases de Java en la instalación (scr.zip).