Trabajo Práctico – Lenguaje Java

Objetivo: Comprender el funcionamiento de clases, tipos primitivos, arreglos y estructuras de control en java.

Tips:

- Recuerde que para validar su implementación debe crear test de unidad.
- Lea la documentación de las clases del sistema que va a manipular como así también los métodos que implementan. De esta forma va a poder saber cómo utilizar sus instancias o métodos de clase. Si no se visualiza el código fuente en su ambiente de programación recuerde los videos tutoriales que están publicados en la página de la materia.
- Recuerde que además de las clases presenciales puede hacer uso de la lista de la materia para evacuar cualquier duda o comentario que tenga sobre los trabajos prácticos. La lista para realizar consultas tpi-est-obj2@listas.unq.edu.ar

Antes de empezar

- Instalar la JDK 1.8 (o superior)
  - Descargarla desde el sitio http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - Ejecutar el archivo y seguir los pasos de la instalación. Prestar atención al directorio de instalación.
- Instalar Eclipse
  - Descargar eclipse desde el sitio https://www.eclipse.org/downloads/
  - Descomprimir el archivo. Ejecutar eclipse.exe.
- Agregar los archivos de código fuente de la API Java.
  - Seguir los pasos del video https://www.youtube.com/watch?v=k1yt6A6pM6Q
  - El archivo src.zip debería estar en el directorio de instalación de la JDK.
- Para usuarios Linux: Eclipse es análogo a Windows, para la JDK ejecutar el comando
  $ *sudo apt-get install openjdk-8-jdk*
  - Debe descargarse el src.zip desde http://download.java.net/openjdk/jdk8/ e incorporarlo al eclipse.

Clases en Java

- En Java, una clase se compone de:
  - Declaración
  - Cuerpo
- Forma general de declaración
  - Si una clase no declara explícitamente su superclase, entonces se asume que extiende a la clase Object.

```
package nombrePaquete;
// importaciones
[modificadores] class NombreClase [extends NombreSuperClase]
[implements NombresInterfaces] {
     // cuerpo
}
```

- **NombreDeClase**:
  - El nombre debe comenzar con mayúsculas, por convención.
  - Java utiliza caracteres Unicode: Persona, Pequeño y AutoDobleTracción son válidos.
  - El alcance de un identificador de clase es todo el paquete en donde se declara la clase, por lo tanto no puede haber dos clases con el mismo nombre dentro de un mismo paquete.
  - Si la clase es pública, el nombre de la clase debe concordar con el nombre del archivo: Persona -> Persona.java.
- El cuerpo de una clase esta delimitado por los signos { y }.
- En el cuerpo se declaran:
  - Atributos
  - Constructores
  - Métodos
  - Otras clases
- El orden de los elementos declarados no está pre-establecido. Por convención se mantiene el orden propuesto.

Paquetes

- Un paquete organiza clases e interfaces detrás de un espacio de nombres. Por lo tanto, está formado por clases e interfaces.
- **nombreDePaquete**:
  - Todo nombre de paquete, por convención, debe comenzar con una letra minúscula.
  - Java utiliza caracteres Unicode: model, cliente.esquema, araña, vistaCliente2 son nombres de paquetes válidos.
  - El alcance de un identificador de paquete es todo el paquete en donde se declara, por lo tanto no puede haber dos paquetes con el mismo nombre dentro de un mismo paquete.

Tipos Primitivos

- int 32-bit complemento a dos.
- boolean true o false.
- char 16-bit caracteres Unicode.
- byte 8-bit complemento a dos.
- short 16-bit complemento a dos.
- long 64-bit complemento a dos.
- float 32-bit IEEE 754.
- double 64-bit IEEE 754.

- Clases “wrappers” para cada tipo primitivo:
  - int --> Integer
  - boolean --> Boolean
  - char --> Character

Sintaxis – Operadores

- **Asignación:** =
  - i=i+3;
  - Otros como C: i += 3; i++;
- **Aritméticos:** + - * / %
  - i+4*3;
  - Otros como C: i++;
- **Lógicos:** & && | || !
  - (i != null) && (3*i > 4)
  - (i>3) & (3*i > 4)
- **Relacionales y Condicionales:** > < >= <= == !=
  - 2 >= 3

Sintaxis – Estructuras de Control

- **If**

```
if (x==3) {
     System.out.println("Tres");
}
```

- **If/else**

```
if (x==1) {
     System.out.println("Uno");
}else {
     System.out.println("Distinto de uno");
}
```

- **While**

```
while (count != -1){
     count++;
     System.out.println("Faltan " + count);
}
```

- **For**

```
for (int i=0; i<9 ; i++ ) {
     System.out.println("El valor de i es " + i);
}
```

- **Foreach**

```
int[] arregloDeEnteros = new int[]{1,2,3};
int suma = 0;
for (int entero:arregloDeEnteros ) {
     suma=+entero;
}
System.out.println(("El valor de la suma es es " + suma);
```

Ejercicio 1 - Contador de pares, impares y múltiplos

Realice un programa que, a partir de un arreglo, cuente:

a. La cantidad de pares.
b. La cantidad de impares.
c. La cantidad de múltiplos de un cierto número.

Realice tests de unidad que verifiquen el funcionamiento.

Componentes del Lenguaje – Strings

- Clase String:
  - No es un tipo primitivo.
  - Los String son instancias de la clase java.lang.String.
  - El compilador trata a los String como si fuesen tipos primitivos del lenguaje.
  - La clase tiene varios métodos para trabajar con ellos.
  - Como crear uno:
  - String saludo = “Hola”;
  - String otroSaludo = new String(“¿Cómo andás?”);

Ejercicio 2 – Examinar las expresiones

Dado el siguiente código:

```
String a = "abc";
String s = a;
String t;
```

Indique que valores retornan o si dan error por qué se produce, las siguientes expresiones:

a. s.length();
b. t.length();
c. 1 + a;
d. a.toUpperCase();
e. "Libertad".indexOf("r");
f. "Universidad".lastIndexOf('i');
g. "Quilmes".substring(2,4);
h. (a.length() + a).startsWith("a");
i. s == a;
j. a.substring(1,3).equals("bc")

Ejercicio 3 – Tipos primitivos

1. ¿Qué son los tipos de datos primitivos?

2. ¿Cuál es la diferencia entre un int y un Integer?

3. ¿Si se define una variable de instancia de tipo int cual sería su valor predeterminado? ¿Y si se define una de tipo Integer? Haga la prueba en Eclipse.

4. Responder la pregunta del punto anterior (3), pero ahora en lugar de definir una variable de instancia se define una variable de método.

Nota: puede obtener información sobre tipos primitivos en:
http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Componentes del lenguaje – Arreglos

- Los arrays de Java (vectores, matrices, hiper-matrices de más de dos dimensiones) se tratan como objetos de una clase predefinida.

- Ejemplo:
  - int[] arregloDeEnteros - Declara un arreglo de enteros pero no inicializa ni reserva memoria.
  - Pueden declararse arreglos de más de una dimensión:
    - int[][] matrizDeEnteros;
  - Se reserva memoria para un arreglo declarado usando new:
    - arregloDeEnteros = new int[5];
    - matrizDeEnteros = new int[5][4];
  - Son “zero-based”.
  - Acceso:
    - Int[] val = matrizDeEnteros[2];
    - matrizDeEnteros[5][3] = 4; //ERROR! El arreglo tiene definido hasta la posición 4.
    - val = matrizDeEnteros[3][0];

Componentes del lenguaje – Comentarios

- Existen 3 tipos:
  - Por línea: **//**
  - Bloque de código: **/* */**
  - JavaDoc: **/** */**

Ejercicio 4 – Multioperador

Programe la clase Multioperador, que permite aplicar las operaciones de suma, resta y multiplicación sobre arreglos de enteros.

Ejercicio 5 – Contador de Palabras

Programe la clase contador de palabras, que dado un String, retorna la cantidad de palabras que contiene (cuenta la cantidad de espacios en blanco). Pista: a un string puede pedirle un arreglo de caracteres equivalente mediante toCharArray().

Ejercicio * - Para hacer entre todos.

Agregue la siguiente clase (new->Junit TestCase) a un proyecto eclipse. Implemente una clase que sea validada por el test.

```
package promedio;

import static org.junit.Assert.*; import org.junit.Before; import
org.junit.Test;

public class CalculadorPromedioTestCase {

     private CalculadorPromedio testedCalculadorPromedio;

     @Before

     public void setup(){

          this.testedCalculadorPromedio=
```

```
            new CalculadorPromedio(new int[]{4,6,8,10,7,9,9,2,3,4});
      }
      @Test
      public void testGetPromedio() {

      assertTrue(this.testedCalculadorPromedio.getPromedio().equals(6.2f));
      }
      @Test
      public void testGetPromedioSinAplazos() {
            assertTrue(this.testedCalculadorPromedio.getPromedioSinAplazos()
                        .equals(7.125f));
      }
}
```

Ejercicio 6 - Pangrama

Un pangrama es una frase que contiene todas las letras del abecedario. Son utilizados en pruebas de impresión para ver en una frase corta como quedan las diferentes letras de una tipografía particular. Programe un pangramador que posee definido las siguientes funcionalidades:

- Indicar si una frase es un pangrama o no. Por ejemplo “Queda gazpacho, fibra, látex, jamón, kiwi y viñas” es un pangrama.
- Indicar, para una frase, cuantas letras del abecedario faltaron.
- Indicar cuantas palabras contiene una frase.

Ejercicio 7 – Jerarquía de paquetes

Cree una clase que se encuentre en paquete model, otra clase que se encuentre en el paquete model.gui y otra que se encuentre en el paquete model.stack . Compílelas utilizando la forma que cree conveniente.

a. ¿Cómo están organizadas en el sistema de archivos?
b. ¿Encuentra alguna relación entre el nombre del paquete y la ubicación de los archivos fuentes de las clases (.java) y los archivos compilados (.class)?

Ejercicio 8 - Point

Diseñe e implemente la clase Point (Punto). La clase debe tener el siguiente comportamiento.

a. Debe ser posible crearse indicando como referencia los valores x e y, también debe ser posible crear un punto directamente sin enviarles parámetros, en estos casos el punto debe crearse en las coordenadas (0,0).
b. Debe ser posible mover un punto a otra posición.
c. Sumarse con otro punto y como resultado obtener un nuevo punto con las valores de x e y sumados.

Ejercicio 9 – Rectángulo

Utilizando el punto implementado anteriormente, defina el comportamiento de un rectángulo definido en un espacio de dos dimensiones, es decir, poseer una ubicación en un espacio de coordenadas x e y. Los rectángulos deben tener el siguiente comportamiento:

a. Crearse en forma apropiada.
b. Obtener el área
c. Obtener el perímetro.
d. Determinar si son horizontales o verticales.

Además, diseñe la clase Cuadrado utilizando lo anterior.

Realice los test de unidad para definir el comportamiento y luego implemente en java.

Ejercicio 10 – De Smalltalk a Java – Personas y Equipos.

Implemente en Java los siguientes puntos (similares a los ya implementados en Smalltalk en el TP1).

a) Defina la clase Persona y modélela en Java. Una persona tiene su nombre, apellido y edad.
b) Defina la clase Equipo y modélela en Java. Un equipo tiene un nombre y un conjunto de integrantes (que son instancias de Persona).
c) Un Equipo debe saber responder su nombre y el promedio de edad de sus integrantes.
d) Instancie un Equipo, instancie 5 Personas y agregue éstas a aquél. Pida al equipo el promedio de edad de sus integrantes e imprima el resultado devuelto.

Ejercicio 11 – De Smalltalk a Java – Personas y Mascotas.

Revea el ejercicio 4 del TP1, en el que se pedía realizar lo siguiente en Smalltalk:

“Instancie dos Personas, dos Mascotas e inserte los cuatro objetos en una colección. Itere sobre la colección e imprima un listado con los nombres de todos los objetos de la misma”.

Responda las siguientes preguntas:

a.-¿Cree que es posible implementar esto de la misma forma en Java?
b.-¿Qué diferencias y dificultades encontraría?