Intro a Java

Trabajo Práctico N°2 - Introducción a Java

Objetivos

En este trabajo se espera que el alumno aprenda a implementar una clase en Java, a aplicar el concepto de herencia, a redefinir métodos y a a hacer tests.

Pre-requisitos

Para la realización de este trabajo, el alumno debe haber:

1. leído el material Java para objetosos
2. instalado el entorno de desarrollo Java llamado Eclipse. En el sitio de la cátedra hay videos sobre cómo instalar el ambiente.
3. leído el objeto de aprendizaje con título ArrayList Cookbook

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:

1. Todo nombre de paquete, por convención, debe **comenzar con una letra minúscula**.
2. Java utiliza **caracteres Unicode**: model, cliente.esquema, araña, vistaCliente2 son nombres de paquetes válidos.
3. 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: > < >= <= == !=

- o 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);
```

1. Contador de pares, impares y múltiplos

Cree un nuevo proyecto de Java en Eclipse. Cree una clase llamada `Counter` dentro del paquete `unq` que, a partir de un `ArrayList` que mantiene como variable privada, sea capaz de contar:

1. La cantidad de pares
2. La cantidad de impares
3. La cantidad de múltiplos de un cierto número.

Luego, cree una carpeta dentro del proyecto llamada "test" (click derecho sobre el proyecto>new>source folder. Debe estar a la misma altura de "src"). En ella, cree un nuevo jUnitTestCase llamado `CounterTestCase`. Complete el test de unidadpara verificar el funcionamiento de la clase `Counter`:

```
package unq;
```

```
import static org.junit.Assert.*;

import org.junit.Before;

import org.junit.Test;



public class CounterTestCase {

    private Counter counter;



    /**

     * Crea un escenario de test básico, que consiste en un contador

     * con 10 enteros

     *

     * @throws Exception

     */

    @Before

    public void setUp() throws Exception {



        //Se crea el contador

        counter = new Counter();
```

```
    //Se agregan los numeros. Un solo par y nueve impares
    counter.addNumber(1);
    counter.addNumber(3);
    counter.addNumber(5);
    counter.addNumber(7);
    counter.addNumber(9);
    counter.addNumber(1);
    counter.addNumber(1);
    counter.addNumber(1);
    counter.addNumber(1);
    counter.addNumber(4);
}

/**
 * Verifica la cantidad de pares
 */
@Test
```

```
    public void testEvenNumbers() {



        // Getting the even occurrences

              int amount = counter.getEvenOcurrences();



        // I check the amount is the expected one

              assertEquals(amount, 9);

    }




}
```

Componentes del lenguaje: Strings

- La 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?");`

2. Examinar las expresiones

Dado el siguiente código:

```
String a = "abc";
```

```
String s = a;


String t;
```

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

- `s.length();`
- `t.length();`
- `1 + a;`
- `a.toUpperCase();`
- `"Libertad".indexOf("r");`
- `"Universidad".lastIndexOf('i');`
- `"Quilmes".substring(2,4);`
- `(a.length() + a).startsWith("a");`
- `s == a;`
- `a.substring(1,3).equals("bc")`

3. Tipos primitivos

Teniendo en cuenta la documentación oficial de Java:

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.

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: /** */

4. Multioperador

Programe la clase `Multioperador`, que permite aplicar las operaciones de suma, resta y multiplicación

sobre `ArrayLists` de enteros. Es decir, poder sumar todos los números que contiene, poder restar todos los números que contiene y poder multiplicar a todos los números que contiene entré si.

5. Jerarquía de paquetes

Cree una clase que se encuentre en el 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 crea conveniente.

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

6. Point

Diseñe e implemente la clase `Point` (punto). La misma debe tener el siguiente comportamiento:

1. Debe ser posible crearse indicando como referencia los valores `x` e `y`
2. También debe ser posible crear un punto directamente sin enviarles parámetros, en este caso el punto debe crearse en las coordenadas (0,0).
3. Debe ser posible mover un punto a otra posición.
4. Sumarse con otro punto y como resultado obtener un nuevo punto con las valores de `x` e `y` sumados.

7. 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:

1. Crearse en forma apropiada y asegurando su consistencia.
2. Obtener el área
3. Obtener el perímetro.
4. Determinar si son horizontales o verticales.

Además, diseñe la clase `Cuadrado` (re)utilizando lo anterior.

Realice los test de unidad correspondientes y luego implemente en java.

8. Encapsulamiento

Implemente la clase `Persona` en Java. Una persona tiene un `nombre` y una `fecha de nacimiento`, por lo que debe ser posible pedirle su `nombre, fecha de nacimiento y edad`. En base a esto:

1. Responder: Si un objeto cualquiera que le pide la edad a una `Persona`: ¿Conoce cómo ésta calcula u obtiene tal valor? ¿Cómo se llama el mecanismo de abstracción que permite esto?
2. Agregue a la clase `Persona` definida anteriormente el método
   `menorQue(Persona persona)`
   que recibe como parámetro a otra persona y retorna `true` en caso de que el receptor sea menor en edad que el parámetro, o `false` en caso contrario.
3. Agregue a la clase `Persona` un método (de clase) de creación, respetando la siguiente firma:
   `Persona(String nombre, Date fechaNacimiento)`
   que recibe como parámetros el `nombre` y la `fecha de nacimiento` de la persona a crear, crea una nueva instancia de `Persona` y la retorna inicializada con los valores recibidos como parámetro.

9. Equipo de Trabajo

1. Defina la clase `Persona` y modélela en Java. Una persona tiene su *nombre, apellido* y *edad*.
2. Defina la clase `EquipoDeTrabajo` y modélela en Java. Un equipo tiene un *nombre* y un conjunto de *integrantes* (que son instancias de `Persona`).
3. Un `EquipoDeTrabajo` debe saber responder su nombre y el promedio de edad de sus integrantes.
4. Instancie un `EquipoDeTrabajo`, instancie 5 `Persona`s y agreguelas al mismo.
5. Pida al `EquipoDeTrabajo` el promedio de edad de sus integrantes e imprima el resultado devuelto.