Programación 2

Introducción al Lenguaje Java

Clases en Java

Profesores:

Ismael Figueroa - ifigueroap@gmail.com

Eduardo Godoy - eduardo.gl@gmail.com

¿Qué es un Objeto?

Cualquier cosa que vemos a nuestro alrededor. Ej: auto.

Un objeto, en general, está compuesto por:

- *Características* tales como: marca, modelo, color, etc.
- *Comportamiento*, por ejemplo: encender, acelerar, frenar, retroceder, etc.

¿Cómo se caracterizan los objetos?

Características

Los objetos tienen un *estado interno* que se representa mediante variables conocidas como *atributos*.

Comportamiento

Además, el comportamiento de los objetos se programa mediante *métodos*, que son funciones o procedimientos que tienen acceso directo al estado interno del objeto.

Clases e Instancias

- **Clases:**

  Modelo para múltiples objetos con características y comportamientos similares. Las clases comprenden todas las características de una serie particular de objetos. En **POO** se definen clases como un modelo abstracto de un objeto. Lo más parecido en C es un struct.

- **Instancias de una clase:**

  Representación concreta de un objeto. Al definir una clase se pueden crear muchas instancias de la misma y cada instancia puede tener diferentes características mientras se comporte y reconozca como objeto de la clase. En programación estructurada sería una *variable*.

Características de un Objeto

- **Atributos:**

  Características que diferencian a un objeto de otro y determinan la "*forma*" de ese objeto.

  Los atributos se definen como **variables**. De hecho podrían, ser vistas como **variables globales del objeto**. Cada instancia de una clase puede tener diferentes valores para sus variables, por lo que a cada variable se le denomina **variable de instancia**. La clase define el tipo de atributo y cada instancia guarda su propio valor para ese atributo.

Características de un Objeto

- **Métodos:**

  Comúnmente denominadas **funciones** o **procedimientos** definidas dentro de una clase, y que operan en sus instancias. Los objetos se **comunican** entre sí mediante el uso **llamadas a de métodos**. Se dice que una clase puede **llamar** al método de otra clase.

  Se pueden definir métodos de instancia (que operan en una instancia de la clase) y los métodos de clase que operan sobre la clase.

Declarando una nueva clase

Cuenta.java

```
1  public class Cuenta {
2
3    private String nombre;
4
5    public void
6    establecerNombre(String nombre) {
7      this.nombre = nombre;
8    }
9
10   public String obtenerNombre() {
11     return nombre;
12     // se refiere a this.nombre
13   }
14 }
```

Declaración de clase

La declaración de clase es:

Cuenta.java

```
1  public class Cuenta
```

1. Por ahora usaremos siempre `public` como nivel de acceso
2. El nombre de la clase siempre debe comenzar con mayúscula
3. **El nombre define un nuevo tipo de dato**
4. El nombre de la clase debe coincidir con el nombre del archivo, por lo tanto debemos trabajar en el archivo `Cuenta.java`
5. El cuerpo de la clase es lo que está entre llaves, y sigue inmediatamente al nombre

Estilo *CamelCase*

Por convención, en Java todos los nombres de clases, variables, métodos, etc., se escriben usando un esquema que se conoce como *CamelCase*:

1. Un nombre de clase parte con mayúscula, y se ocupa minúsculas hasta el comienzo de la siguiente palabra. Por ejemplo `CuentaAhorro`, `CuentaCorriente` son nombres de otras clases
2. Los nombres de variables y métodos parten con mínusculas, y al cambiar palabras se usa una maýuscula en la nueva palabra. Por ejemplo: `obtenerNombre`, `rutTitular`, son nombres de método/variable que siguen la convención

Declaración de variables de instancia

La declaración de variable de instancia es:

Cuenta.java

```
1  private String nombre;
```

Sobre estas variables podemos decir:

1. Existen antes, durante, y después de la ejecución de los métodos
2. Se definen en el cuerpo de la clase, pero fuera del cuerpo de los métodos
3. Los métodos pueden—y en general es la idea que lo hagan—manipular los valores de las variables de instancia
4. Cada *instancia* tiene su propio juego de variables de instancia. Los valores específicos se conocen como *el estado del objeto*

Acceso `private` y `public` para variables de instancia

Las variables de instancia también se pueden configurar con un modificar de acceso:

- `private`: la variable solamente puede utilizarse en los métodos de la clase que lo declara. En este caso objetos `Cuenta`
- `public`: la variable puede usarse en los métodos de su clase, pero también está disponible para otros objetos

Declaración de métodos

Cuenta.java

```
1  public void establecerNombre(String nombre) {
2    this.nombre = nombre;
3  }
```

1. El *encabezado o firma* del método consiste en su acceso, tipo de retorno, parámetros, y excepciones lanzadas
2. Los parámetros funcionan como variables locales dentro del cuerpo del método
3. El parámetro `nombre` esconde a la variable de instancia que tiene el mismo nombre
4. Por eso se usa el acceso `this.nombre` para referenciar explícitamente al atributo

Usando nuestra clase en un programa

Main.java

```
1  public class Main {
2    public static void main(String[] args) {
3      // El operador new crea instancias de una clase dada
4      Cuenta miCuenta = new Cuenta()
5      System.out.println("Nombre cuenta: " +
6                                 miCuenta.obtenerNombre());
7
8      miCuenta.establecerNombre("Juanito");
9      System.out.println("Nombre cuenta: " +
10                                miCuenta.obtenerNombre());
11   }
12 }
```

Creando una instancia de Cuenta

Main.java

```
1  Cuenta miCuenta;
```

Esta declaración define la variable `miCuenta` como una variable de tipo `Cuenta`. Todas las clases definen un nuevo tipo de dato, que es el tipo asociado a sus instancias.

Creando una instancia de Cuenta

Main.java

```
1  Cuenta miCuenta = new Cuenta();
```

La variable puede inicializarse inmediatamente.
El operador `new` *se usa para construir una nueva instancia*.

El constructor por defecto

Main.java

```
1  new Cuenta();
```

Cuando hacemos `new Cuenta()` estamos invocando un método especial que se llama *constructor*, y que sirve para crear instancias.

Toda clase tiene un *constructor por defecto*, que es el nombre de la clase y sin parámetros. Por eso se invoca como `new Cuenta ()`

Los constructores solamente pueden ser invocados al usar `new`

Constructores Explícitos

Podemos construir tantos constructores como queramos:

Cuenta.java

```
1  public class Cuenta {
2    private String nombre;
3
4    public Cuenta(String nombre) {
5      this.nombre = nombre;
6    }
7
8    /* ... idem anterior ... */
9  }
```

Usando constructores

Main2.java

```
1  public class Main2 {
2    public static void main(String[] args) {
3      // El operador new crea instancias de una clase dada
4      Cuenta miCuenta1 = new Cuenta()
5      System.out.println("Nombre cuenta 1: " +
6                         miCuenta1.obtenerNombre());
7
8      Cuenta miCuenta2 = new Cuenta("Juanito");
9      System.out.println("Nombre cuenta 2: " +
10                        miCuenta2.obtenerNombre());
11   }
12 }
```

Error de compilacion!

Dado que definimos un constructor, Java ya no crea uno por defecto!

Más Constructores Explícitos

Podemos construir tantos constructores como queramos:

Cuenta.java

```
1  public class Cuenta {
2    private String nombre;
3
4    /* define explicitamente
5    el constructor por defecto */
6    public Cuenta() { /* nada */ }
7
8    public Cuenta(String nombre) {
9      this.nombre = nombre;
10   }
11
12   /* ... idem anterior ... */
13 }
```

Agregando saldo a las cuentas

```
public class Cuenta {
  private String nombre;
  private double saldo;
  public Cuenta(String nombre, double saldo) {
    this.nombre = nombre;
    if (saldo > 0.0) {
      this.saldo = saldo;
    }
  }

  public void depositar(double montoDeposito) {
    if (montoDeposito > 0.0) { saldo += montoDeposito; }
  }

  public double obtenerSaldo() {
    return saldo;
  }
  /* ... idem anterior ... */
}
```

Encapsulamiento

- Técnica para asociar en un mismo lugar *el estado interno* y *el comportamiento* de los objetos
- En Java el encapsulamiento se realiza mediante la definición de clases

Ocultamiento de Información

- Es la capacidad de ocultar los detalles internos del comportamiento de una clase y exponer sólo los detalles que sean necesarios para el resto del sistema
- En Java esto se hace a través de los atributos de control de acceso: `private` y `public`

Preguntas

Preguntas?