Clases y Objetos en JAVA

JAIME ALBERTO GUZMAN LUNA
Universidad Nacional de Colombia
Medellín

Contenido

- Clases y Objetos:
  - Atributos
  - Métodos

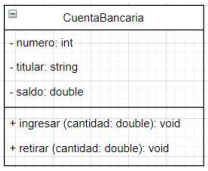
Clases y Objetos

Visión general

La Clase (1)

- La clase es una abstracción del mundo real
- Permite el encapsulamiento de datos y código
  - atributos + métodos →**Clase**
- Ejemplo de una clase

Cuenta Bancaria



Clase en UML

Clase en JAVA

```
class CuentaBancaria {
   long numero;
   String titular;
   double saldo;
   void ingresar (double cantidad) {
    saldo += cantidad;
   }
   void retirar (double cantidad) {
     if (cantidad > saldo)
     System.out.println ("Saldo insuficiente");
     else saldo -= cantidad;
   }
}
```

Nombre

Atributos

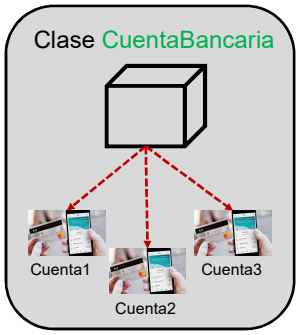
Métodos

Aspectos clave de las clases

- Es una definición que nos permite crear y manipular objetos de esa clase.
- Las clases no pueden ejecutarse por si mismas.
  - Crear una clase con la operación principal (main) para iniciar la ejecución del programa.

```
public static void main(String[] args) {
}
```

Los Objetos



Objeto: es una instancia de una clase.

- Una clase define un tipo de dato que se puede utilizar para declarar atributos
  - CuentaBancaria cuenta1, cuenta2;
- Declarar un objeto es declarar una referencia a un objeto
  - Los objetos en java se crean con el operador new

cuenta1 = **new** CuentaBancaria ();

Cuenta Bancaria	
número	-- sin definir --
titular	-- sin definir --
saldo	-- sin definir --

  - Crear un objeto significa reservar espacio en memoria para sus atributos
  - En java ***new*** reserva memoria para un objeto y devuelve una referencia al objeto
- **Los objetos siempre utilizan memoria dinámica**

Pregunta.....

- ¿Cuántos objetos se crean en el siguiente código?

```
Datos primero = new Datos( "Elena" );      // (1)
String nombre = new String( "Juan" );      // (2)
Datos segundo;                             // (3)
primero = segundo;                         // (4)
segundo = new Datos( nombre );             // (5)
primero = new Datos( "Pedro" ) ;           // (6)
```

- Respuesta
  - Se crean cuatro objetos (por que existen 4 **news**, líneas 1, 2, 5 y 6).

Los atributos

Tipos de Atributos y su acceso

- Tipo primitivo (int, long, double, etc.)

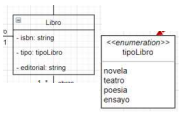
```
class CuentaBancaria {
    long numero;
    String titular;
    double saldo;
    ..... // definir método ingresar y retirar
}
```

- Tipo **Referencias**:
  - Una clase del programa (es decir pueden asociar a "objetos")

```
class CuentaBancaria {
    long numero;
    Cliente titular;
    double saldo;
    ..... // definir método ingresar y retirar
}
```

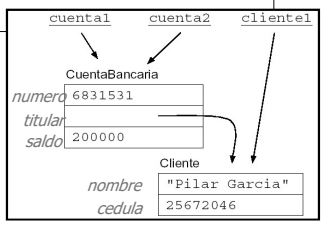
```
class Cliente {
    String nombre;
    long cedula;
}
```

  - Enumerados: son clases que representan un conjunto finito de valores



Acceso a atributos

```
public class Banco {
   public static void main(String[] args) {
      CuentaBancaria cuenta1, cuenta2;
      Cliente cliente1;
      cuenta1 = new CuentaBancaria ();
      cuenta2 = cuenta1;
      cliente1 = new Cliente ();
      cliente1.nombre ="jaime Guzman";
      cliente1.cedula = 25672046;
      cuenta1.numero = 6331531;
      cuenta1.titular = cliente1;
      cuenta1.ingresar(100000);
      cuenta2.ingresar(100000);
      cuenta2.titular.nombre = "Pilar Garcia";
      }
}
```



HELPFUL TIPS

Tipiado Estático: Todos los datos se inicializan automáticamente a valores por defecto (por ejemplo, datos numéricos a 0, los datos de referencia a null, String es un null y booleano a false).

Manejo de variables y atributos (1)

EJERCICIO 1

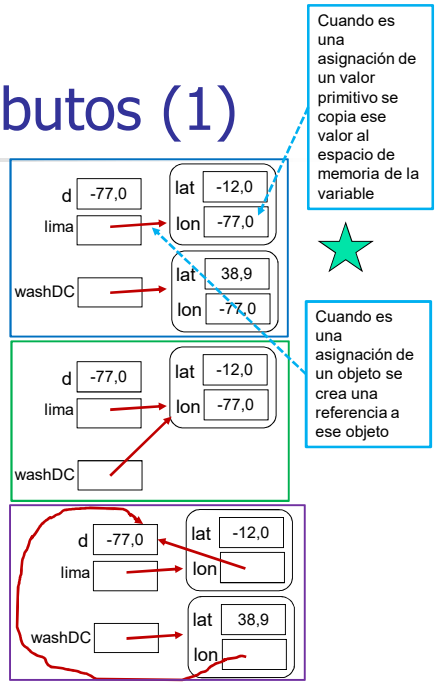
- Dada la definición para la clase SimpleLocation

```
class SimpleLocation {
double lat;
double lon;
}
```

- Seleccione el correcto modelo de memoria después de correr el siguiente código main de la clase

```
public class Ejercicio1 {

   public static void main(String[] args) {
      double d=-77.0;
      SimpleLocation lima=new SimpleLocation ();
      lima.lat=-12.0;
      lima.lon=d;
      SimpleLocation washDC=new SimpleLocation ();
      washDC.lat=38.9;
      washDC.lon=lima.lon;
   }
}
```



Manejo de variables y atributos (2)

EJERCICIO 2

- Dada la definición para la clase SimpleLocation

```
class SimpleLocation{
    double lat;
    double lon;
}
```

- Seleccione la respuesta correcta después de ejecutar el siguiente código main de la clase

```
public class Ejercicio2 {
    public static void main(String[] args ){
        SimpleLocation loc1=new SimpleLocation();
        loc1.lat=39.9;
        loc1.lon=116.4;
        SimpleLocation loc2=new SimpleLocation();
        loc2.lat=55.8;
        loc2.lon=37.6;
        loc1=loc2;
        loc1.lat=-8.3;
        System.out.println(loc2.lat+","+loc2.lon);
    }
}
```

1. 55,8, 37,6
2. -8,3, 37,6 ★
3. -8,3, 116,4
4. 39,9, 116,4

Se asigna a loc1 el objeto al que apunta loc2. loc1 y loc2 apuntan al mismo objeto

Se cambia en el objeto al que apuntan loc1 y loc2, el atributo **lat** desde el apuntador loc1.

Atributos de instancia y de clase (1)

```
class CuentaCorriente {
   private long numero; //Atrib. de instancia
   public static double interes=2.0;//Atrib. de clase
   ...
}
```

```
CuentaCorriente c = new CuentaCorriente ();
System.out.println(c.interes);
System.out.println(CuentaCorriente.interes);
```

2.0
2.0

Acceso mediante la clase

Acceso mediante objetos

Atributos de instancia: están vinculados a una instancia específica (objeto) de la clase; no se comparten entre objetos de la misma clase.

Atributos de clase: están vinculados a una clase; se comparten entre objetos de la misma clase. También llamados **atributos estáticos**.

Los atributos de clase son atributos compartidos por todos los objetos de la clase.

```
CuentaCorriente cuenta1 = new CuentaCorriente ();
CuentaCorriente cuenta2 = new CuentaCorriente ();
cuenta1.interes = 0.7;
cuenta2.interes = 1.0;
System.out.println (cuenta1.interes);
```

→ 1.0

Sólo hay un ejemplar del atributo, de forma que todos los objetos acceden a la misma zona de almacenamiento

La memoria de una variable de clase se reserva al cargar la clase en el intérprete

Constantes

- Cuando queremos definir constantes dentro de clases, utilizamos la combinación **final static.**
- El valor no podrá ser modificado

```
public class Circulo {

  private double radio;
  public final static double PI = 3.1415;
  void f () {
     PI = 3.0;
  }
}
```

ERROR

The final field Circulo.PI cannot be assigned

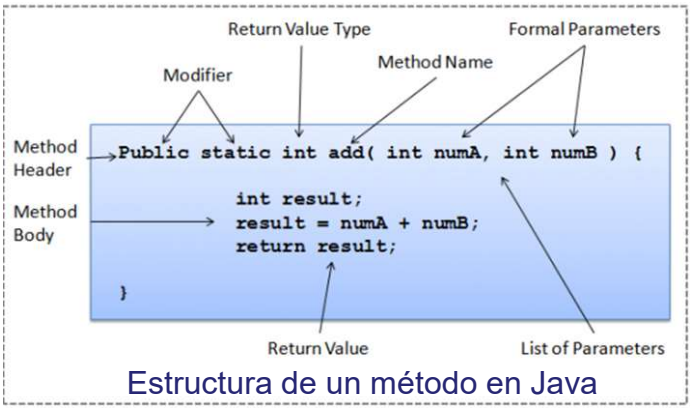
- Si una variable de Clase lleva el calificador **final**, quiere decir que no se permite que se vuelva a crear una nueva instancia con el calificador new, pero si se permite cambiar el contenido de la instancia.

```
final Punto pto = new Punto();
pto.setPtX(3); //Correcto
//Error con pto = new Punto();
```

Los métodos

Los métodos

- Los métodos son funciones definidas dentro de una clase



Estructura de un método en Java

- Todo método tiene un valor de retorno
  - Si no devuelve nada se indica con *void*
- Para cada método se establece el nivel de visibilidad
- Instrucciones que pueden ir:
  - Asignación
  - Estructuras condicionales y de iteración
  - Invocación a otros métodos
  - Creación de objetos

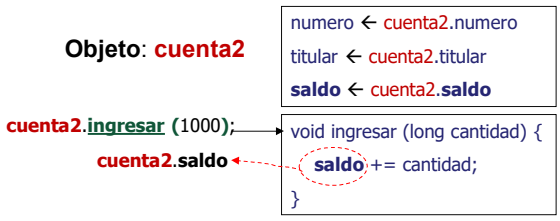
Los métodos y los atributos

- Los métodos pueden referenciar directamente a los atributos de la clase

```
class CuentaBancaria {
   long numero;
   Cliente titular;
   double saldo;
   void ingresar (double cantidad) {
   saldo += cantidad;
   }
   void retirar (double cantidad) {
     if (cantidad > saldo)
     System.out.println ("Saldo insuficiente");
     else saldo -= cantidad;
   }
}
```

Atributo

- Al ejecutar un método desde un objeto de clase **A**, las variables del método toman el valor que tienen sus atributos en el objeto



Llamadas a métodos desde un método

- Los métodos pueden invocar directamente otros métodos de la misma clase
- Al ejecutar un método invocado desde un **objeto** de clase A, las llamadas a otros métodos de la clase A se ejecutan sobre el mismo objeto a menos que se invoquen sobre otro objeto

- **cuenta3** = new CuentaBancaria ();
- cuenta2.transferencia (**cuenta3**, 1000);

```
class CuentaBancaria {
   ....
   void transferencia (CuentaBancaria destino, long cantidad) {
        If (cantidad <= saldo) {
            retirar (cantidad);
            destino.ingresar (cantidad);
        }
   }
}
```

cuenta2.retirar (cantidad)

cuenta3.ingresar (cantidad)

Paso de argumentos

```
class CuentaBancaria {
   long numero;
   String titular;
   long saldo;
}
```

- Por valor:

```
public class ClaseMain {

    public static void main(String[] args) {
        int n = 5;
        System.out.println("antes: " + n);
        f(n);
        System.out.println("despues: " + n);
    }
    static void f (int i) {
        System.out.println("durante: " + ++i);
    }

}
```

RESPUESTA

```
antes: 5
durante: 6
despues: 5
```

- Por referencia:

```
public class ClaseMain {

    public static void main(String[] args) {
      CuentaBancaria cuenta = new CuentaBancaria();
      cuenta.saldo = 100000;
      System.out.println("saldo antes: " + cuenta.saldo);
      arruinar (cuenta);
      System.out.println("saldo despues: " + cuenta.saldo);
    }
    static public void arruinar (CuentaBancaria cnt) {
      cnt.saldo = 0;
      cnt = null;
    }
}
```

RESPUESTA

```
saldo antes: 100000.0
saldo despues: 0.0
```

Métodos de instancia y de clase (1)

- Un **método de instancia** es el que se invoca siempre sobre una instancia (objeto) de una clase (métodos comunes).
- Los **métodos de clase**, son métodos que se ejecutan desde la propia clase.
  - Para declarar un método de clase, basta añadir el modificador **static** en su declaración.

Método de clase

```
public class CuentaCorriente {
    private long numero;
    private static double interes;
    private static long ncuentas;

    static long generarNumero () {
       return ncuentas++;
    }
}
```

- Pueden ser invocados desde la clase o desde los objetos

```
CuentaCorriente cuenta = new CuentaCorriente ();
cuenta.numero = cuenta.generarNumero ();
cuenta.numero = CuentaCorriente.generarNumero ();
```

Desde el objeto

Desde la clase

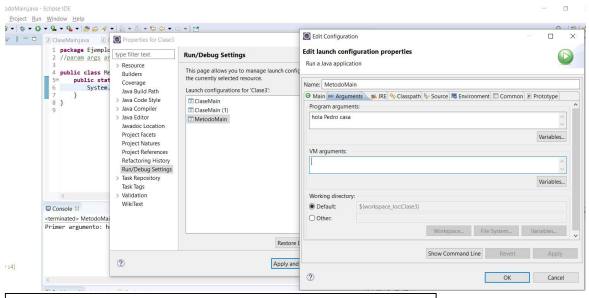
- Restricciones:
  - Pueden acceder a atributos y métodos estáticos de la misma clase
  - **No** pueden acceder a atributos ni métodos de instancia de la clase
  - Los métodos de instancia sí pueden acceder a variables y métodos estáticos

Método Main

```
public static void main(String[] args) {
}
```

- En aplicaciones java se requiere por lo menos tener un método u operación **main** dentro de alguna clase.
- Debe ser publico para que el programa lo pueda acceder.
- Debe ser estático ya que hace parte de la clase y no de los objetos.
  - Supongamos que no fuese estático, de esta manera necesitaríamos un objeto para ejecutar esta operación.
    ¿Cómo creamos el objeto para luego ejecutar esta operación? -> no tiene sentido, no se puede.

Para eclipse los argumentos se configuran en en menú Project; Run Debug Settings; Edith launch configuration properties; Arguments.




```
public class ClaseMain {
  public static void main(String[] args) {
    System.out.println("arg0: " + args[0]);
    System.out.println("arg1: " + args[1]);
  }
}
```

```
arg0: hola
arg1: Pedro
```

La variable this

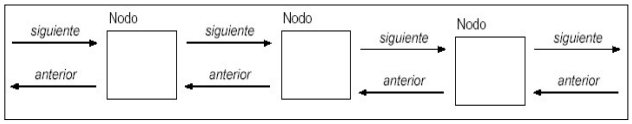
- Es una variable de referencia en java que hace referencia al objeto actual
- Usos:
  - Caso 1: Se puede usar para referir la variable de instancia (objeto) de la clase actual

```
.....
private int edad;
public setEdad(int edad){
   this.edad = edad;
}
.....
```

- Se puede usar para devolver la instancia (objeto) de clase actual sobre el que se invoca el método
  - Ejemplo **this** para modelar relaciones inversas

```
class Nodo {
      Nodo anterior;
      Nodo siguiente;
      void conectar (Nodo z) {
            siguiente = z;
            z.anterior = this;
      }
}
```

Objeto dueño del mètodo



Pregunta de repaso

- Dado el siguiente código: Identifique que imprime el método main

```
public class Alcance {
   int x = 1;
   int y = 2;

   int metodo1(int x){
      x = 5;
      return x*x*x;
   }
   void metodo2(){
      --x;
   }
   void metodo3(){
      int x = 5;
      this.x = x * this.x;
   }
```

```
   void metodo4(){
      int x = 6;
      metodo3();
      System.out.println("x = " + x );
      System.out.println("x = " + this.x );
      metodo2();
      x = metodo1(this.x);
      System.out.println("x = " + x );
      System.out.println("x = " + this.x );
      this.x = metodo1(x);
      System.out.println("x = " + x );
      System.out.println("x = " + this.x );
   }
   public static void main(String [] args){
      Alcance alc1 = new Alcance ();
      alc1.metodo4();
   }
}
```

Solución:

```
x = 6
x = 5
x = 125
x = 4
x = 125
x = 125
```

Resumen

- Los conceptos aprendidos en el contexto de JAVA fueron…
  - Que es una clase y objeto
  - Características de los atributos primitivos y de referencia
  - Como se manejan en memoria los atributos
  - Que son los Atributos de instancia y de clase
  - Como se definen los métodos
  - Que son métodos de instancia y de clase
  - Que es el método *Main*
  - Que es la palabra clave *this*