Programación Orientada a Objetos en JAVA

Jorge Pérez

Introducción a la Computación

Orientación a Objetos

- OO es un paradigma de modelación y programación
- Idea principal: modelar los problemas lo mas cercanos al contexto en el que ocurren y
- Hacer un programa siguiendo esta modelación.
- Conceptos fundamentales (iniciales):
  - Clases
  - Objetos

Clases y Objetos

Lo más simple es verlo con ejemplos:

- Clase: Persona
- Objetos: Juan, Alberto, Julia
- Clase: Perro
- Objetos: Boby, Lazy, Benji
- Clase: Pais
- Objetos: Chile, Australia, Afganistan

Clases y Objetos

Lo más simple es verlo con ejemplos:

- Clase: Curso
- Objetos: ING1310, ING2210
- Clase: Alumno
- Objetos: Pedro, Oscar, Carolina, Diego
- Clase: Universidad
- Objetos: UAndes, PUC, UTalca

Clases y Objetos

Lo más simple es verlo con ejemplos:

- Clase: Número Complejo
- Objetos: $(3 + 5i)$, $(-5 - 7i)$, 0, i
- Clase: Polinomio
- Objetos: $2x + 7$, $3x^2 - 5$, 0, x^{100}

Clases y Objetos

- Clase: Computador
- Objetos: ....
- Clase: Automóvil
- Objetos: ....
- Clase: Figura Geométrica
- Objetos: .....
- Clase: Animal
- Objetos: ....

Clases y propiedades (atributos)

Un objeto que pertenece a una clase se llama instancia de esa clase.
Si un objeto A es una instancia de la clase C diremos que A *es de tipo* C.
Todos los objetos (o instancias) de una misma clase compaten ciertos atributos.

- Clase: Persona
- Atributos: nombre, sexo, fecha de nacimiento, estatura ....

- Clase: Perro
- Atributos: raza, color de pelo, sexo, ...

- Clase: Pais
- Atributos: nombre, área, número de habitantes, ...

Objetos y atributos

Distintos objetos (de una misma clase) pueden tener distintos valores en estos atributos:

- Clase: Persona
- Objeto: Ivan Zamorano (el ex-futbolista)
- Atributos:
  - nombre: Ivan Zamorano
  - sexo: masculino
  - estatura: 1,80 metros
  - ....
- Clase: Perro
- Objeto: Cachupin (el perro de mi vecino)
- Atributos:
  - raza: no definida
  - color de pelo: negro con café
  - sexo: macho
  - ....

Objetos y atributos

- Dos objetos de una misma clase pueden tener exactamente los mismos valores para todos sus atributos pero ser objetos distintos
- Por ejemplo pueden existir dos personas distintas que se llamen Ivan Zamorano, de sexo masculino, que midan 1,80 mts, ...
- Los valores de los atributos NO definen la identidad de un objeto
- (recuerden lo que pasaba con los strings en JAVA....)
- Los valores de los atributos definen el estado del objeto

Ejercicios: Atributos, Objetos?

- Clase: Curso
- Clase: Alumno
- Clase: Universidad
- Clase: Computador
- Clase: Automóvil
- Clase: Figura Geométrica
- Clase: Animal

Objetos, atributos, clases...

- Un objeto puede tener otros objetos como (valores de) atributos
- Por ejemplo, un objeto de tipo Alumno podria tener como atributos un conjunto de objetos de tipo Curso
  - Tengo varios objetos de tipo Curso: ING1310, ING1130, ING1110, ING1150, ING1140
  - Un objeto A de tipo Alumno podria tener a ING1310, ING1130, ING1150 como cursos
  - Un objeto B de tipo Alumno podria tener a ING1150, ING1140 como cursos
- En este caso decimos que la clase Alumno tiene un conjunto de Cursos como parte de sus atributos.
- Ejercicio: piense en ejemplos de objetos que tienen otros objetos como atributos.

Operaciones, métodos sobre objetos

- Se pueden realizar *operaciones* o *métodos* sobre objetos (instancias) de una clase, para modificar su estado.
- Por ejemplo:
  - Cambiarle el nombre a una persona
  - Asignar un nuevo curso a un alumno
  - Eliminar a un alumno un curso que tenia asignado
- Estos métodos son intrinsecos a la Clase, o sea, pueden ser aplicadas a cualquier objeto de la clase.
- Note que estos métodos deben recibir parámteros (el nuevo nombre, el nuevo curso, el curso a eliminar)

Operaciones, métodos sobre objetos

- Existen otros métodos que nos permiten consultar por el estado de un objeto
- o por cierta información que puede ser deribada de los atributos...
- Por ejemplo:
  - Cuál es el nombre de una persona
  - Cuál es la edad de una persona
  - Tiene un alumno algún curso asignado
  - Cuántos cursos tiene asignado un alumno
  - Cuántos créditos en total tiene un alumno
- Estas operaciones son intrinsecas a la Clase, o sea, pueden ser aplicadas a cualquier objeto de la clase.
- Note que estos métodos entregan resultados.

Clases, atributos, operaciones

- En general para definir (modelar) una clase se debe especificar
  - los atributos que tendrán los objetos de esa clase
  - las operaciones que se pueden realizar sobre los objetos de la clase (consultas, modificaciones de estado)
- Tanto los atributos como las operaciones sobre los objetos que definamos, dependerán de la realidad que queremos modelar.

Ejemplo: Super 8

Queremos modelar a un vendedor de "Super 8"

- Clase: Vendedor de Super 8
- Atributos:
  - cantidad de Super 8
  - precio de cada Super 8
  - monto actual
- Métodos:
  - le quedan Super 8
  - cuantos Super 8 le quedan
  - cuanto cuesta un Super 8
  - comprarle un Super 8
  - comprar varios Super 8

¿Qué métodos reciben parámetros? ¿Cómo afecta a una instancia de la clase "Vendedor de Super 8" la aplicación de estos métodos? Haga ejemplos para distintas instancias

Ejercicio

Modele al "Vendedor de Super 8" pero suponiendo que debe lidiar con el vuelto para cada compra
¿Qué atributos debe tener?
¿Qué parámetros deben recibir los métodos para comprar Super 8?
¿Cómo afecta al estado de un objeto la aplicación de estos métodos?

Construyendo una instancia

- Hemos visto que el *estado* de una instncia está dado por los valores de sus atributos.
- Que las operaciones sobre los objetos nos permiten consultar por o modificar el estado de una instancia.
- Pero cuál es el estado inicial?
- En el ejemplo del vendedor de Super 8, si queremos modelar la venta diaria podriamos pensar que:
  - la cantidad inicial de Super 8, y el precio de cada Super 8 lo decide el vendedor (la instancia) al salir de su casa en la mañana.
  - el monto de dinero inicial es 0.
  - luego durante el día se interactúa con la instancia del vendedor y su estado evoluciona.
- La idea de asociar un estado inicial a una instancia se conoce técnicamente como *construir una instancia*.
- Un constructor es un método que asigna un estado inicial a una instancia de una clase (debe recibir los parámteros necesarios)

Clases en JAVA

```
class NombreClase
{
   Atributos de la clase
   ...

   Metodo Constructor de la clase
   Metodos de la clase
   ...
}
```

- En JAVA todas las clases comienzan su definicion con la palabra reservada `class` seguido del nombre de la clase
- Le siguen los atributos de la clase (que tendran valores para las distintas instancias)
- El constructor (o constructores)
- Los métodos de la clase.

Atributos de Clases en JAVA

- Los atributos de una clase en JAVA se definene igual que declaraciones de variables.
- Por ejemplo para la case Pais podriamos tener los atributos:

```
class Pais
{
  string nombre;
  double area;
  int cantidadHabitantes;
  ....
}
```

- Para el Vendedor de Super 8:

```
class VendedorSuper8
{
  int precioSuper8;
  int cantidadSuper8;
  int monto;
  ....
}
```

Métodos de Clases en JAVA

- Los métodos de las clases tienen una sintaxis similar a los métodos que hemos estado utilizando, salvo que no usaran la palabra `static`.
- En general un método dentro una clase se verá como

```
public valor_retorno nombre_metodo(...parametros...)
{
   codigo del metodo
   .....
}
```

- Dentro de los métodos se puede alterar el estado modificando los valores de los atributos.
- Tambien se puede consultar por el estado accediendo a los valores de los atrbutos.
- MUY IMPORTANTE: el nombre del método constructor debe ser el mismo nombre del la clase y NO se debe especificar valor de retorno para el constructor.

Métodos de Clases en JAVA

- Para el ejemplo del vendedor de Super 8 tenemos

```
class VendedorSuper8
{
  int precioSuper8;
  int cantidadSuper8;
  int monto;

  public VendedorSuper8(int p, int c)
  {
    precioSuper8 = p;
    cantidadSuper8 = c;
    monto = 0;
  }

  .... otros metodos ....
}
```

  incluyendo el constructor de las instancias.
- Dependiendo de los valores que sean entregados al constuctor se crearan distintas instancias.

Métodos de Clases en JAVA

- Para el ejemplo del vendedor de Super 8 podemos agregar un método que responde si le quedan o no Super8 (retorna un boolean)

```
class VendedorSuper8
{
  int precioSuper8;
  int cantidadSuper8;
  int monto;

  public VendedorSuper8(int p, int c)
  {
    precioSuper8 = p;
    cantidadSuper8 = c;
    monto = 0;
  }

  public boolean leQuedanSuper8()
  {
    if (cantidadSuper8 > 0)
    { return true; }
    else
    { return false; }
  }
  .... otros metodos ....
}
```

Métodos de Clases en JAVA

- Para el ejemplo del vendedor de Super 8 ahora agregamos el metodo para comprar un Super 8, note como se altera el estado.

```
class VendedorSuper8
{
  int precioSuper8;
  int cantidadSuper8;
  int monto;

  ......

  public void compraSuper8()
  {
    if(cantidadSuper8 > 0)
    {
      cantidadSuper8--;
      monto = monto + precioSper8;
    }
  }
  .... otros metodos ....
}
```

- complete la definicion agregando un método que permita obtener cual es el monto actual del vendedor.

Interactuando con instancias desde el método principal.

- Hemos visto como definir modelar una clase
- Veremos ahora como interactuar con objetos de estas clases
- La idea será usar inicialmente el método principal `main` para crear objetos y utilizar sus métodos y atributos.
- Veremos como se hace en JAVA para crear y utilizar objetos.

Creando una instancia

- Los objetos se utilizan usando variables de JAVA, al igual que otros tipos de datos.

```
Clase obj;
```

- Esto crea una variable `obj` que será una referencia a un objeto de la clase `Clase`, por ejemplo

```
VendedorSuper8 Juan;
```

- Para crear objetos se usa `new` y un llamado al constructor de la clase en particular

```
new Clase(.....);
```

- Esto crea un objeto usando el constructor de la clase `Clase`, por ejemplo

```
new VendedorSuper8(100, 5);
```

crea un objeto de la clase `VendedorSuper8` con 5 Super 8 y un precio de 100.

Creando una instancia y llamando a métodos

- Para crear un objeto y asigarselo a una variable hacemos

```
Clase obj = new Clase(...);
```

- Por ejemplo

```
VendedorSuper8 Juan = new VendedorSuper8(100, 5);
```

crea un objeto de la clase `VendedorSuper8` usando el constructor, y se lo asigna a la variable `Juan`.

Atributos y métodos

- Cuando tenemos creado un objeto podemos referirnos a sus atributos y llamar a sus métodos de la siguiente forma:

```
obj.atributo;
obj.metodo(.....);
```

- `obj.atributo` se puede usar como una variable cualquiera, para asignar un valor o dentro de una expresion
- `obj.metodo(...)` se usa igual que cualquier llamado a método, se puede utilizar su resultado para realizar tareas
- Por ejemplo podemos hacer

```
Juan.cantidadSuper8 = 10;
Juan.cantidadSuper8 = Juan.cantidadSuper8 * 2;
```

para manipular el atributo `cantidadSuper8` del objeto `Juan`.

Atributos y métodos

```
VendedorSuper8 Juan = new VendedorSuper8(100, 5);

while( Juan.leQuedanSuper8() )
{
  System.out.println("Aun quedan Super 8, se puede comprar otro...");
  Juan.comprarSuper8();
}
```

Ejemplo: numeros complejos

Considere una clase para manejar números complejos

```
class Complejo
{

  double re;
  double im;

  public Complejo(double r, double i)
  {
    re = r;
    im = i;
  }

  public double Norma()
  {
    double norma = Math.sqrt( re * re + im * im );
    return norma;
  }

  public void Conjuga()
  {
    im = - im;
  }

}
```

Ejemplo: numeros complejos

En una clase podemos tener varios constructores que reciban distintos parámetros, por ejemplo en la clase complejo podriamos tener

```
class Complejo
{
  double re;
  double im;

  public Complejo(double r, double i)
  {
    re = r;
    im = i;
  }

  public Complejo(double r)
  {
    re = r;
    im = 0;
  }

  public Complejo()
  {
    re = 0;
    im = 0;
  }
  .....
}
```

El constructor usado dependera de como se llame.

Ejemplo: numeros complejos

```
Complejo c1 = new Complejo(3,4);
Complejo c2 = new Complejo(1, 3.5);
Complejo c3 = new Complejo(0.5);
Complejo c4 = new Complejo();

System.out.println(c1.Norma());

c1.Conjuga();

System.out.println(c1.Norma());

System.out.println(c2.re + " + " c2.im + "i");
System.out.println(c3.re + " + " c3.im + "i");
System.out.println(c4.re + " + " c4.im + "i");
```

Numeros complejos: ejemplos

Escriba métodos para la clase complejos

- `public boolean equals(Complejo c)` que retorna `true` si `c` es un complejo del mismo valor de la instancia que llama al método.
- `public Complejo Copy()` que retorna una copia (nuevo objeto) del complejo que hace la llamada.
- `public Complejo GeneraConjugado()` que retorne el número complejo conjugado de la instancia que se llama.
- `public void Suma(Complejo c)` que le suma el complejo `c` a la instancia.

Cuidado, en cada método se usa/necesita un complejo *diferente* del que hace la llamada al método.

Complejo: equals

```
class Complejo
{
  double re;
  double im;

  ...

  public boolean equals(Complejo c)
  {
    if( re == c.re && im = c.im )
    { return true; }
    else
    { return false; }
  }

  ...
}
```

Complejo: equals

```
public static main(String[] args)
{
  Complejo c1 = new Complejo(10, -7);
  Complejo c2 = new Complejo(10, -7);

  if ( c1 == c2 )
  {
    System.out.println("Son el mismo objeto"');
  }

  if ( c1.equals(c2) )
  {
    System.out.println("Los complejos tienen el mismo valor");
  }

  if ( c1.equals(c1) )
  {
    System.out.println("Los complejos tienen el mismo valor");
  }
}
```

Complejo: Copy, GeneraConjugado

```
class Complejo
{
  double re;
  double im;

  ...

  public Complejo Copy()
  {
    Complejo c = new Complejo(re, im);
    return c;
  }

  public Complejo GeneraConjugado()
  {
    Complejo c = new Complejo(re, im);
    c.Conjuga();
    return c;
  }

  ...
}
```

Complejo: Copy, GeneraConjugado

```
public static main(String[] args)
{
  Complejo c1 = new Complejo(10, -7);
  Complejo c2 = c1.Copy();
  Complejo c3 = c2.GeneraConjugado();

  if ( c1 == c2 )
  {
    System.out.println("Son el mismo objeto");
  }

  if ( c2.equals(c3) )
  {
    System.out.println("Los complejos tienen el mismo valor");
  }

  c2.Conjuga();

  if ( c2.equals(c3) )
  {
    System.out.println("Los complejos tienen el mismo valor");
  }
}
```