Tema 0. Conceptos básicos Programación Orientada a Objetos

Estructura de Datos y Algoritmos (EDA)

Slides created by ADS professors

Objetivos

- Al final de la clase, los estudiantes deben ser capaces de:

1) Comprender los conceptos de clase y objeto
2) Comprender las características del soporte de Java a la Programación Orientada a Objetos (POO)

Contenidos

1. ¿Qué es la POO?
2. Clases y objetos
3. Atributos
4. Métodos
5. Paso de parámetros en Java
6. Destrucción de objetos

Contenidos

1. ¿Qué es la POO?
2. Clases y objetos
3. Atributos
4. Métodos
5. Paso de parámetros en Java
6. Destrucción de objetos

¿Qué es la POO?

- Un paradigma de programación que define un programa como un conjunto de objetos que realizan acciones

- Un objeto tiene:
  - propiedades: datos almacenados en atributos
  - operaciones: métodos

- Una clase define una plantilla para un tipo de objetos

Ventajas de la POO

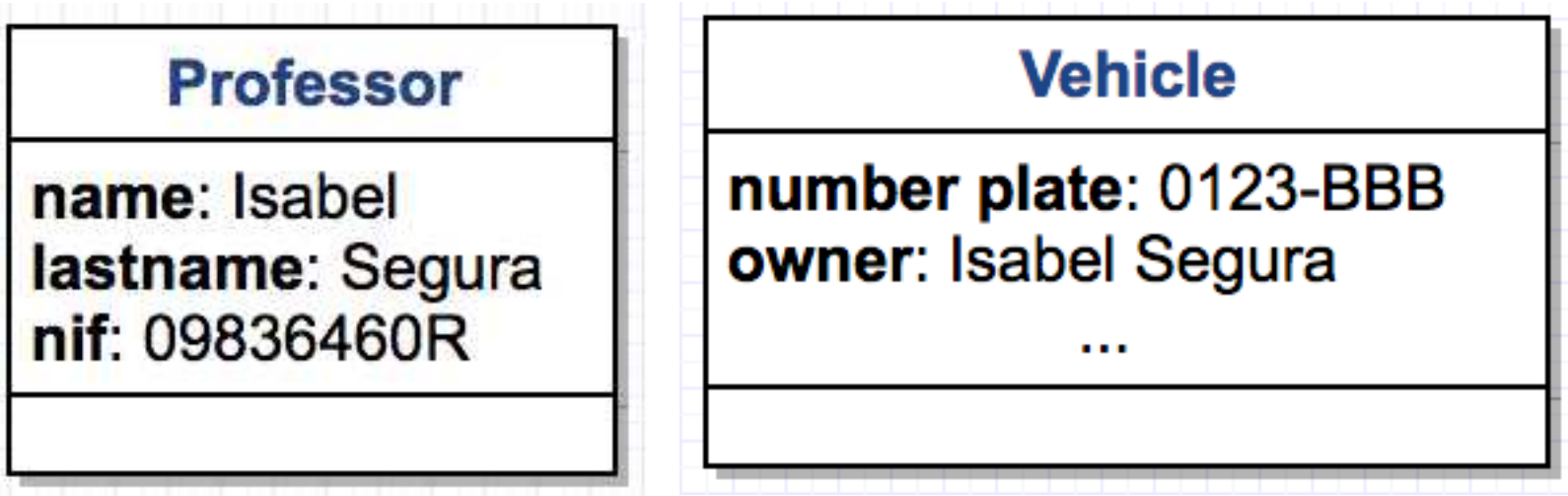
- Los datos y los comportamientos se encapsulan en una sola unidad (clase)
- Fácil de mantener
- Permite un desarrollo más rápido
- Promueve la reutilización

Contenidos

1. ¿Qué es la POO?
2. Clases y objetos
3. Atributos
4. Métodos
5. Paso de parámetros en Java
6. Destrucción de objetos

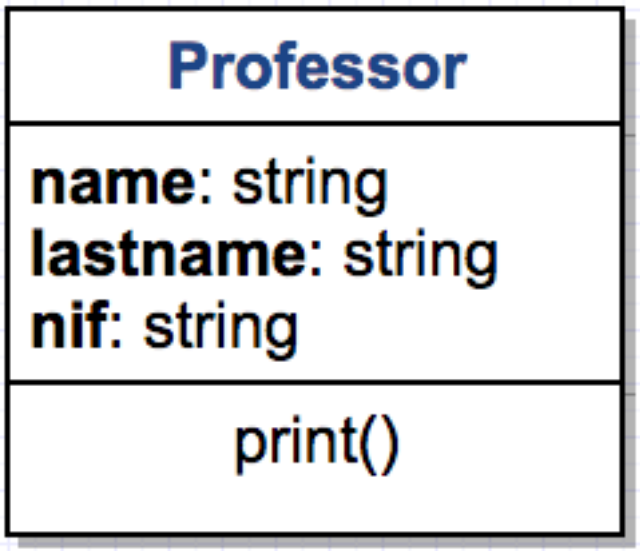
Clases y objetos

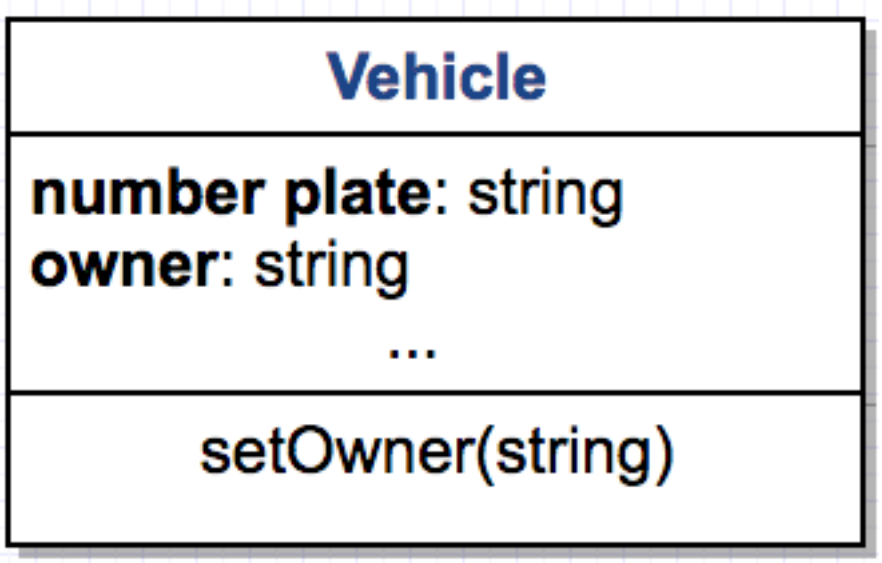
- Objeto: representa una entidad del dominio de la aplicación



Clases y objetos

- Clase: plantilla para objetos
- Clase = estructura de datos + operaciones





https://www.gliffy.com/uses/uml-software/

Clases en Java

public class <name_class> {

<attributes>

<methods>

}

```
public class Point {

}
```

Modificadores

- public: accesible desde cualquier otra clase
- private: no accesible fuera de esta clase
- protected: accesible por subclases
- none: solo accesible en el mismo paquete
- final: no se puede modificar
- static: compartido por todos los objetos de la clase

Recomendaciones sobre el alcance de los modificadores

- **public** para clase
- **private** para todos los atributos y para definir sus métodos getters y setters (podemos controlar cómo se modifican los atributos)
- **public** para métodos (excepto para métodos internos)

Contenidos

1. ¿Qué es la POO?
2. Clases y objetos
3. Atributos
4. Métodos
5. Paso de parámetros en Java
6. Destrucción de objetos

Atributos

- Representar el estado de un objeto.

[modifiers] <type> <name_attribute>;

- Puede ser un tipo básico (char, int, float, boolean, etc), una colección (array) o incluso un objeto

Static vs. Non-static atributos

- **Static (or class) atributos**: valores comunes para todos los objetos (constants, counters, etc)
  - Se pueden llamar directamente (sin crear ningún objeto): nombre de la clase, punto (.), seguido del nombre del método
- **Non-static atributos**: son locales a una instancia del objeto (cada objeto tiene sus propios valores)
  - Se debe crear una instancia de objeto para acceder a ellos

```
public class Point {
    public final static float MAX=100;
    public float x;
    public float y;
}

public class TestPoint {
    public static void main(String args[]) {
        System.out.println(Point.MAX);
        Point p=new Point();
        p.x=3;
        p.y=4;
    }
}
```

Declaración de static attribute

Declaración de non-static attribute

Acceso al static attribute

Acceso al non-static attribute

Contenidos

1. ¿Qué es la POO?
2. Clases y objetos
3. Atributos
4. Métodos
5. Paso de parámetros en Java
6. Destrucción de objetos

Métodos

- Definir el comportamiento de un objeto
- Operar sobre los atributos
- Pueden aceptar parámetros como entrada
- Pueden devolver un valor u objeto como salida

Constructores

- Crean un objeto y proporciona valores iniciales para sus atributos
- Mismo nombre que la clase
- Pueden tomar argumentos
- No devuelven nada (ni siquiera vacío)
- Se ejecutan con el operador **new** (se asigna memoria para el objeto)
- Una clase puede tener varios constructores

Constructores

```
public class Point {
    public float x;
    public float y;
}
```

Si no se define ningún constructor, Java proporciona un constructor implícito que crea un objeto e inicializa sus atributos con valores por defecto

```
public static void main(String args[]) {
    Point p=new Point();
    System.out.println("Point="+ p.x +","+ p.y);
}
```

Produce el siguiente resultado: Point= 0.0 , 0.0

Constructores

```
public Point(float a, float b) {
        x=a;
        y=b;
}
```

Si defines un constructor, el constructor implícito no existe

```
4=      public static void main(String args[]) {
5           Point p=new Point();
6           System.out.println("Point="+ p.x +","+ p.y);
7       }
```

The constructor Point() is undefined

Error: El constructor Point () no está definido

Creación de un objeto

```
Point p;
```

p es una variable de referencia para un objeto de clase Point. Esta instrucción declara la variable p, pero el objeto aún no se ha creado

```
p=new Point(3,5);
```

El operador **new** asigna memoria para el objeto
El constructor crea el objeto
La variable p contiene la dirección de memoria donde se almacena el objeto
p también nos permite llamar a los atributos y métodos del objeto

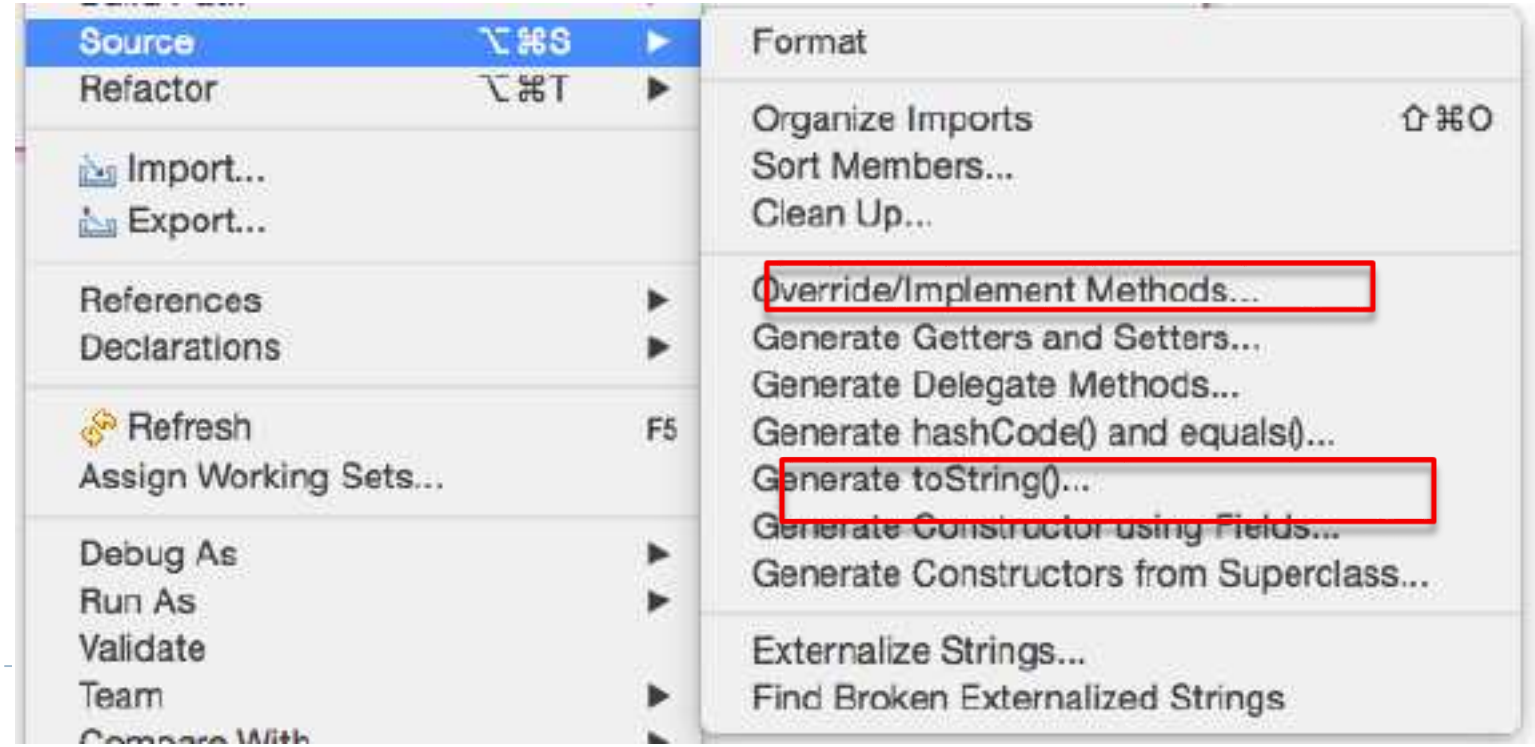
Creación de un objeto

```
Point p =new Point(3,5);
```

También puedes crearlo en la misma línea !!!

Escribe menos con Eclipse …

- La opción **Source** permite generar automáticamente un constructor así como los métodos getters y setters (para manipular los atributos)



```
public class Point {
    public float x;
    public float y;
    public Point(float a, float b) {
        x = a;
        y = b;
    }
    public float getX() {return x;}
    public float getY() {return y;}

    public void setX(float a) {
        x = a;
    }
    public void setY(float b) {
        y = b;
    }
}
```

Cómo acceder a los atributos y métodos del objeto

```
Point p=new Point(3,5);
p.setX(p.getX()+2);
p.setY(p.getY()-5);
System.out.println(p.getX()+","+p.getY());
```

Usando la variable de referencia, p, seguido de punto (.) Y el nombre del método o atributo

¿Qué pueden devolver los métodos?

```
public float getX() {
    return x;
}
public Point clone() {
    return new Point(x,y);
}
public void show() {
    System.out.println("("+x+","+y+")");
}
```

a tipo básico (chair, int, float, double, boolean, etc)

Un objeto

void

Nota: también pueden devolver arrays de objetos o de tipos básicos

Static vs. Non-static métodos

- **Static (o class) métodos**: son métodos generales que devuelve valores calculados a partir de los valores de los parámetros
  - **Se puede sólo acceder** a static atributos/métodos
- **Non-static métodos**: manipula los atributos de un objeto
  - **Se puede acceder a** non-static y static atributos/métodos

this

- Un objeto puede referirse a sí mismo con la palabra clave **this**

```
public Point(float a, float b) {
    this.x = a;
    this.y = b;
}
```

this

- Si un atributo y un parámetro comparten el mismo nombre => use **this** para distinguirlos.

```
public class Point {
    public float x;
    public float y;

    public Point(float x, float y) {
        this.x = x;
        this.y = y;
    }
```

Contenidos

1. ¿Qué es la POO?
2. Clases y objetos
3. Atributos
4. Métodos
5. Paso de parámetros en Java
6. Destrucción de objetos

Paso de parámetros en Java

- **Una variable de tipo básico** siempre se pasa por valor (si el método cambia este valor, el efecto no es visible fuera del método)
- Las **Referencias a variables** de arrays y objetos siempre pasan por valor. Estas referencias son su dirección de memoria.
- Sin embargo, los valores de los arrays o de los atributos de un objeto puede ser modificado por el método, y el cambio será visible fuera del método

Ámbito

- Parámetros y variables locales: un único método donde se definen
- Non-static atributos: variables globales para todos los non-static métodos
- Static atributos: variables globales para static y non-static métodos

Contenidos

1. ¿Qué es la POO?
2. Clases y objetos
3. Atributos
4. Métodos
5. Paso de parámetros en Java
6. Destrucción de objetos

Destrucción de objetos

- Java proporciona un **garbage collector** para encontrar automáticamente objetos y arrays que ya no son necesarios