


2022/02

Lenguajes y Paradigmas de Programación

Hasta ahora que hemos visto

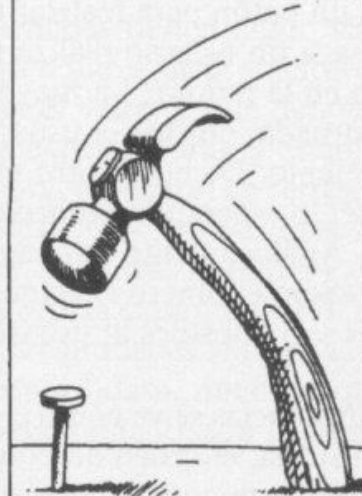
- ✓- Historia de programación
- ✓- Paradigma procedural
- ✓- App Tarea C
-  Paradigma orientado al objeto
- App Tarea Java
 - Paradigma funcional
 - App Tarea Clojure / Python

Atributos



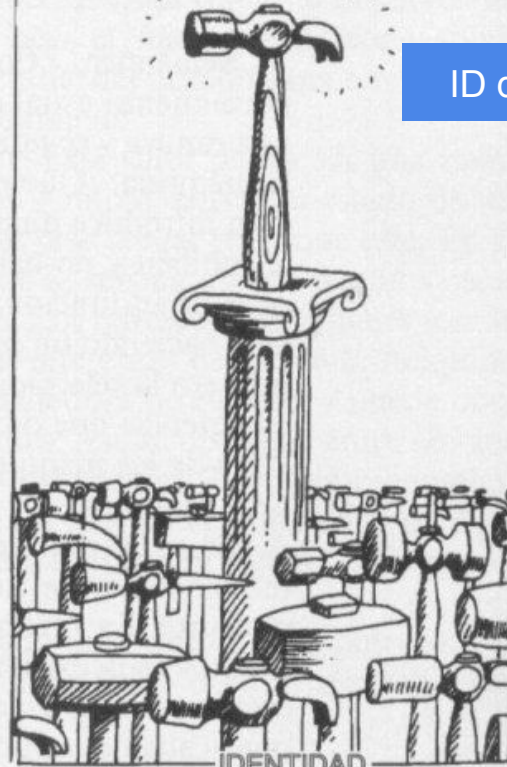
ESTADO

Métodos



COMPORTAMIENTO

ID de Objeto

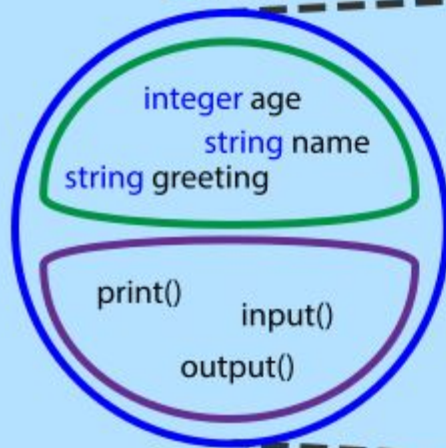


IDENTIDAD

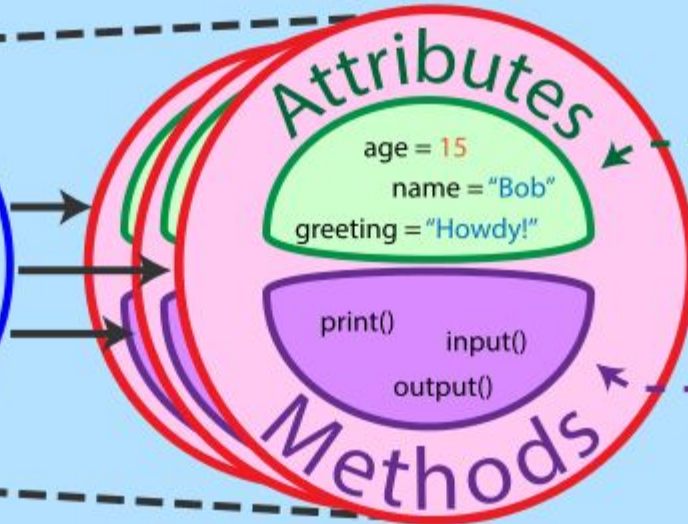
Un objeto tiene estado, exhibe algún comportamiento bien definido, tiene una identidad única.

Object Oriented Programming

Class

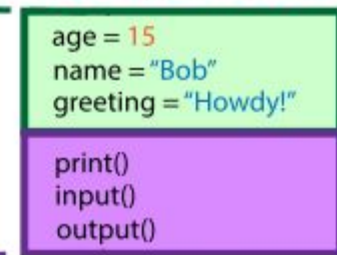


Objects



Procedural Programming Equivalent

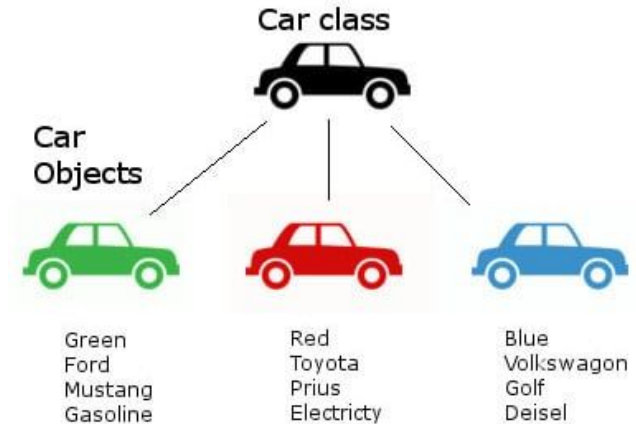
Variables



Procedures

POO - Pilares

- Encapsulamiento
- Abstracción
- Herencia
- Polimorfismo



Actividad

- Para entender mejor POO lo que vamos a hacer es modelar la sala de clases
- Definan las clases de todos los objetos que hay en la sala, incluyendo sus variables y funciones que cambien su estado

Discusión de implementación

Conceptos

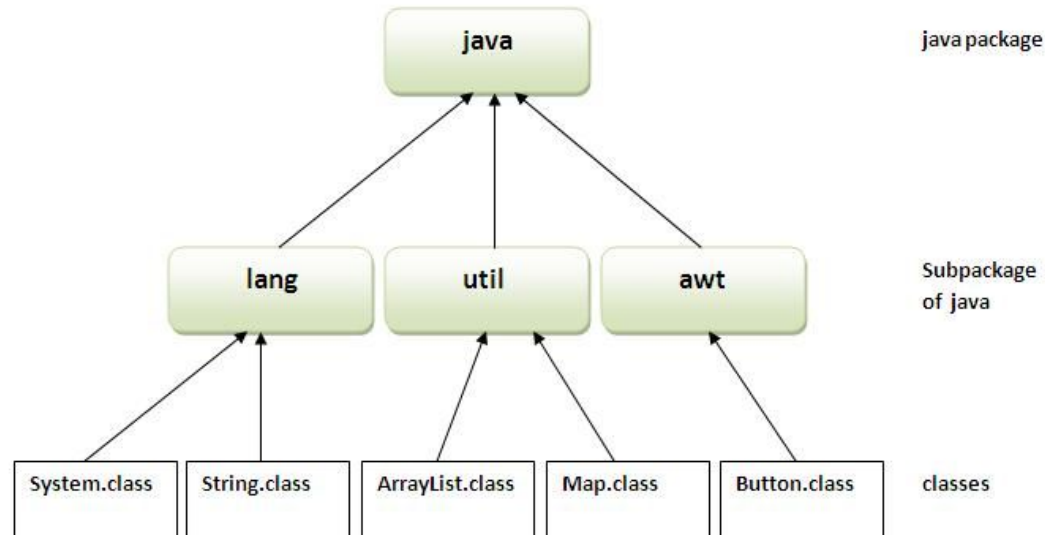
Paquetes:

Los paquetes son el mecanismo que usa Java para **facilitar la modularidad del código**. Un paquete puede contener una o más definiciones de interfaces y clases, distribuyéndose habitualmente como un archivo.

Módulos y Paquetes

Los módulos y paquetes son agrupaciones de clases usualmente organizados por organización/empresa (usando el nombre o el dominio al revés) y/o por funcionalidad

En Java



¿Cómo se usa?

1. Declarar el paquete:

Al inicio de la clase que se quiere meter en el paquete:

Nombre del paquete

```
package cl.uai.tics200;
```

Keyword package

En Java

Lo que hago es escribir al principio de cada archivo algo del estilo

```
package cl.uai.tics200;
```

Keyword package

Nombre del paquete

Y **colocando** ese archivo en el directorio cl/uai/tics200 de mi código fuente

¿Cómo se usa?

Usar las clases del paquete: 2 formas

1) `import cl.uai.tics200.Persona;` // o * para todas las clases

Keyword import

Nombre del paquete

Nombre de la clase

...

`Persona p = new Persona();`

...

1) `cl.uai.tics200.Persona p = new
cl.uai.tics200.Persona();`

Los import son como el usar los .h en C, y se colocan al principio del archivo

En Python

En Python el equivalente son los módulos, que al igual que Java son directorios

PERO

- No deben ser declarados
- Para que sean módulos deben tener un archivo llamado `__init__.py` en él

```
sound/                                Top-level package
  __init__.py                          Initialize the sound package
  formats/                             Subpackage for file format conversions
    __init__.py
    wavread.py
    wavwrite.py
    aiffread.py
    aiffwrite.py
    auread.py
    auwrite.py
    ...
  effects/                             Subpackage for sound effects
    __init__.py
    echo.py
    surround.py
    reverse.py
    ...
  filters/                             Subpackage for filters
    __init__.py
    equalizer.py
    vocoder.py
    karaoke.py
    ...
```

¿Cómo se usa?

Usar las clases del paquete: 2 formas

1) `import cl.uai.tics200`

Keyword import

Nombre del paquete

`p = cl.uai.tics200.Persona()`

...

Nombre del módulo

1) `from cl.uai.tics200 import Persona`
`p = Persona()`

Ventajas

1. Categorizar las clases de modo que puedan ser mantenidas fácilmente.
2. Protección de acceso.
3. Elimina la colisión de nombres.

Protección de acceso

- Public
- Private
- Protected
- Package-level

Niveles de Acceso

Visibilidad	public	protected	/* default */	private
De la misma clase	Sí	Sí	Sí	Sí
De cualquier clase en el mismo paquete	Sí	Sí	Sí	No
De una clase hija en el mismo paquete	Sí	Sí	Sí	No
De cualquier clase hija afuera del paquete	Sí	Sí	No	No
De cualquier clase	Sí	No	No	No

Protección de acceso

- Public se refiere a que se puede acceder desde cualquier parte los métodos o atributos definidos
 - En Java se usa el keyword public para definirlo
 - Es el nivel de protección por omisión de Python

Java

```
public class Demo {  
    public String myString;  
}
```

Cualquiera puede acceder a la clase Demo y a la variable myString

```
public class Main {  
    public static void main(String[]  
    argv) {  
        Demo demo = new Demo();  
        demo.myString = "hola";  
    }  
}
```

Protección de acceso

- Protected se refiere a que solamente la clase actual o alguna clase hija puede acceder a métodos o atributos definidos
 - En Java se usa el keyword `protected` para definirlo
 - En Python no existe esto... pero por convención se usa un underscore (`_`) para decir que por favor no toquen esto a menos que seas una clase hija

Java

```
public class Demo {  
    public String myString;  
    protected String  
anotherString;  
}
```

```
public class Main {  
    public static void main(String[]  
argv) {  
        Demo demo = new Demo();  
        demo.myString = "hola";  
        demo.anotherString = "chao";  
    }  
}
```

Movamos la clase Demo a otro paquete

Java

```
package demo;

public class Demo {
    public String myString;
    protected String
anotherString;
}
```

```
import demo.Demo;

public class Main {
    public static void main(String[]
argv) {
        Demo demo = new Demo();
        demo.myString = "hola";
        demo.anotherString = "chao";
    }
}
```

Main.java

saved

```
1 import demo.Demo;
2 public class Main {
3     public static void main(String[]
4         argv) {
5         Demo
6         demo.String anotherString
7         demo.anotherString = "chao";
8     }
9 }
```

[Java] The field Demo.anotherString is not visible

demo.String anotherString

demo.anotherString = "chao";

<https://TraumaticScarceCo>

OpenJDK Runtime Envi

4)

[]

**Volvamos la clase Demo al
paquete raíz :)**

Python

```
class Cup:
    def __init__(self):
        self.color = None
        self._content = None # protected variable

    def fill(self, beverage):
        self._content = beverage

    def empty(self):
        self._content = None
```



Protección de acceso

- Private implica que nadie fuera de la clase debería ser capaz de acceder a este método o atributo
 - En Java se usa el keyword private para definirlo
 - En Python existe el concepto de name mangling para lograr esto

Java

```
public class Demo {  
    public String myString;  
    protected String  
anotherString;  
    private String lastString;  
}
```

```
public class Main {  
    public static void main(String[]  
argv) {  
        Demo demo = new Demo();  
        demo.myString = "hola";  
        demo.anotherString = "chao";  
        demo.lastString = "nop";  
    }  
}
```

```
Main.java  saved  ▼  
1  public class Main {  
2      public static void main(String[]  
3          argv) {  
4          Demo  
5          demo.  
6          demo.String lastString  
7          demo.lastString = "nop";  
8      }  
9  }
```

[Java] The field Demo.lastString is not visible

https://TraumaticScarceCoordina
OpenJDK Runtime Environm
4)
[]

¿Name mangling?

Si colocamos como nombre de método o atributo algo con dos underscores (__) al principio, entonces Python “oculta” esto, dejándolo como `__<nombre clase>__<nombre método o atributo>`


```
class Cup:
    def __init__(self, color):
        self._color = color    # protected variable
        self.__content = None  # private variable

    def fill(self, beverage):
        self.__content = beverage

    def empty(self):
        self.__content = None
```

Se podría acceder a este atributo usando:

```
c = Cup('black')
c._Cup__content = 'hola'
```