Platzi

Programación Orientada a Objetos

Anahí Salgado

@anncode

Developer

Teacher

Syllabus Planner

Education Team

Platzi

Java

Android

Firebase

¿Por qué aprender Programación Orientada a Objetos?

Programar más rápido

- Programar más rápido
- Dejar de ser Programador Jr.
  *Reclutadores
-

Encapsulamiento

Abstracción

Herencia

Polimorfismo

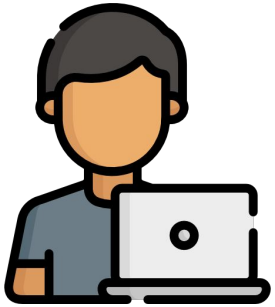
- Programar más rápido
- Dejar de ser Programador Jr.
  *Reclutadores
- Dejar de Copiar y Pegar Código

Ser un Programador Sr.
y conseguir un mejor salario

¿Qué haremos?

Analizar

Plasmar



Programar

Analizar

Problemas

- Observación
- Entendimiento
- Lectura

Plasmar

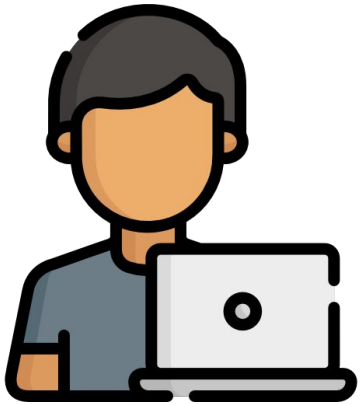
Análisis de problemas

- Diagramas

Programar

Diagramas

- Lenguajes de Programación

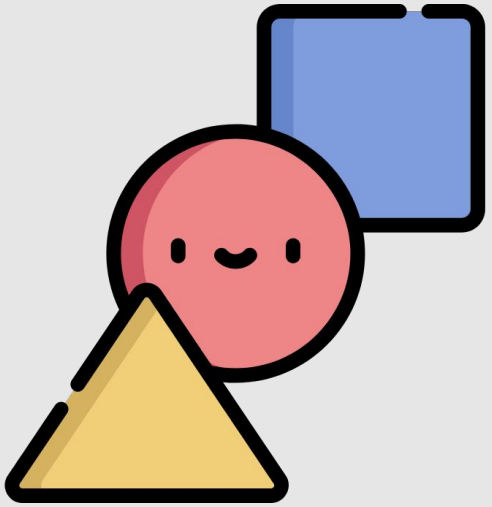


¿Qué resuelve?

Programación Orientada a Objetos

Programación Estructurada

Programación Estructurada



Programación Orientada a Objetos

Código muy largo

Código muy largo

Si algo falla, todo se rompe

Programación Estructurada

Código muy largo

Si algo falla, todo se rompe

Difícil de mantener

Programación Estructurada

Código Espagueti

if () {} else {}

```
C     A weird program for calculating Pi written in Fortran.
C     From: Fink, D.G., Computers and the Human Mind, Anchor Books, 1966.

      PROGRAM PI
      DIMENSION TERM(100)
      N=1
    3 TERM(N)=((-1)**(N+1))*(4./(2.*N-1.))
      N=N+1
      IF (N-101) 3,6,6
    6 N=1
    7 SUM98 = SUM98+TERM(N)
      WRITE(*,28) N, TERM(N)
      N=N+1
      IF (N-99) 7, 11, 11
   11 SUM99=SUM98+TERM(N)
      SUM100=SUM99+TERM(N+1)
      IF (SUM98-3.141592) 14,23,23
   14 IF (SUM99-3.141592) 23,23,15
   15 IF (SUM100-3.141592) 16,23,23
   16 AV89=(SUM98+SUM99)/2.
      AV90=(SUM99+SUM100)/2.
      COMANS=(AV89+AV90)/2.
      IF (COMANS-3.1415920) 21,19,19
   19 IF (COMANS-3.1415930) 20,21,21
   20 WRITE(*,26)
      GO TO 22
   21 WRITE(*,27) COMANS
   22 STOP
   23 WRITE(*,25)
      GO TO 22
   25 FORMAT('ERROR IN MAGNITUDE OF SUM')
   26 FORMAT('PROBLEM SOLVED')
   27 FORMAT('PROBLEM UNSOLVED', F14.6)
   28 FORMAT(I3, F14.6)
      END
```

```
function iIds(startAt, showSessionRoot, iNewNmVal, endActionsVal, iStringVal, seqProp, htmlEncodeRegEx) {
    if (SbUtil.dateDisplayType === 'relative') {
        iRange();
    } else {
        iSelActionType();
    }
    iStringVal = notifyWindowTab;
    startAt = addSessionConfigs.sbRange();
    showSessionRoot = addSessionConfigs.elHiddenVal();
    var headerDataPrevious = function(tabArray, iNm) {
        iPredicateVal.SBDB.deferCurrentSessionNotifyVal(function(evalOutMatchedTabUrlsVal) {
            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                iPredicateVal.SBDB.normalizeTabList(function(appMsg) {
                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                        iPredicateVal.SBDB.detailTxt(function(evalOrientationVal) {
                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                iPredicateVal.SBDB.neutralizeWindowFocus(function(iTokenAddedCallback) {
                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                        iPredicateVal.SBDB.evalSessionConfig2(function(sessionNm) {
                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                iPredicateVal.SBDB.iWindow2TabIdx(function(iURLsStringVal) {
                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                        iPredicateVal.SBDB.idx7Val(undefined, iStringVal, function(getWindowIndex) {
                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                addTabList(getWindowIndex.rows, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? show
                                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                        evalSAllowLogging(tabArray, iStringVal, showSessionRoot && showSessionRoot.length > 0 ?
                                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                BrowserAPI.getAllWindowsAndTabs(function(iSession1Val) {
                                                                                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                        SbUtil.currentSessionSrc(iSession1Val, undefined, function(initCurrentSe
                                                                                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                                                                                                addSessionConfigs.render(matchText(iSession1Val, iStringVal, eva
                                                                                                    id: -13,
                                                                                                    unfilteredWindowCount: initCurrentSessionCache,
                                                                                                    filteredWindowCount: iCtrl,
                                                                                                    unfilteredTabCount: parseTabConfig,
                                                                                                    filteredTabCount: evalRegisterValue5Val
                                                                                                }] : [], cacheSessionWindow, evalRateActionQualifier, undefined,
                                                                                                    if (seqProp) {
                                                                                                        seqProp();
                                                                                                    }
                                                                                                });
                                                                                            }
                                                                                        });
                                                                                    }
                                                                                });
                                                                            }
                                                                        });
                                                                    }
                                                                });
                                                            }
                                                        });
                                                    }
                                                });
                                            }
                                        }, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt ? [startAt] : []);
                                    }
                                });
                            }
                        });
                    }
                });
            }
        });
    };
});
```

Simplificar la Programación

Programación Orientada a Objetos

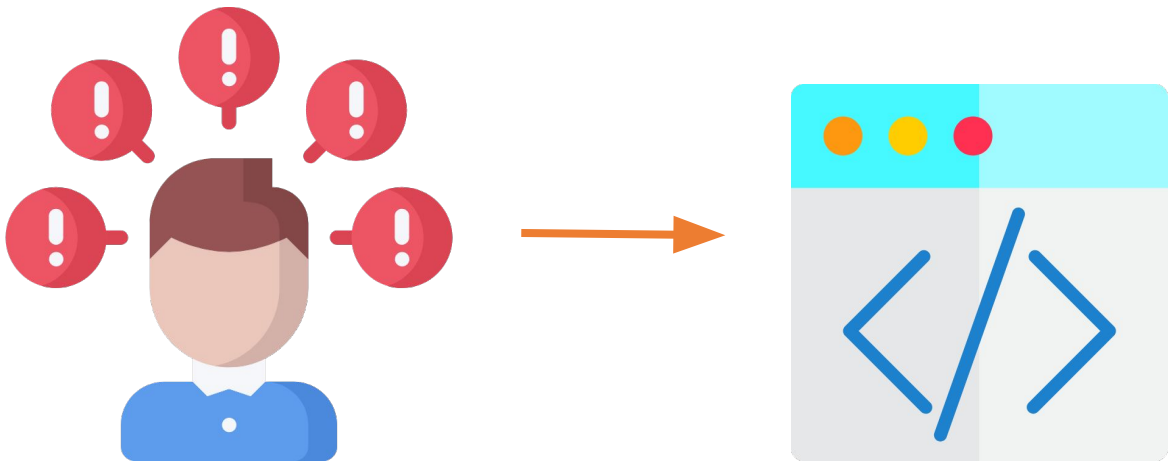
Programación Orientada a Objetos

POO

Orientación a Objetos

Orientación a Objetos

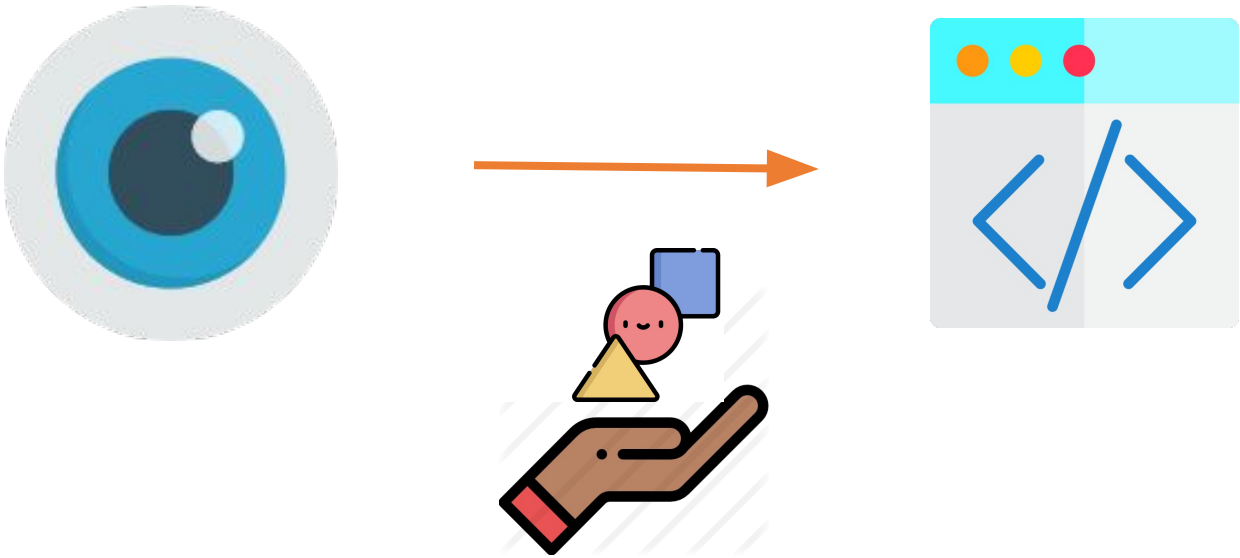
Surge a partir de los problemas que tenemos y necesitamos plasmar en código



Orientación a Objetos

Observar los problemas en

forma de objetos



Paradigma

Programación Orientada a Objetos

Paradigma

+ Teoría que suministra la base y modelo para resolver problemas

Paradigma de Programación Orientada a Objetos

+ Se compone de estos 4 elementos:

Clases

Propiedades

Métodos

Objetos

Paradigma de Programación Orientada a Objetos

+ Se compone de estos 4 elementos

+ Tiene estos pilares:

Encapsulamiento

Abstracción

Herencia

Polimorfismo

Lenguajes Orientados a Objetos


Java


php


C#


C++

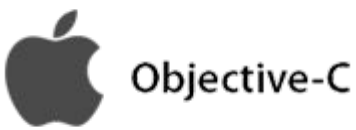

Perl


JavaScript


Ruby


DELPHI


Kotlin


Objective-C


Ada
The Programming Language

Lenguajes de Programación

- Java
- PHP
- Python
- JavaScript
- C#
- Ruby
- Kotlin

Lenguajes de Programación

- Java
- PHP
- Python
- JavaScript
- C#
- Ruby
- Kotlin

Java

- Orientado a Objetos naturalmente
- Android
- Server Side

`.java`

PHP

- Lenguaje interpretado
- Pensado para la Web

`.php`

Python

- Diseñado para ser fácil de usar
- Mútiples usos: Web, Server Side, Análisis de Datos, Machine Learning, etc.

`.py`

JavaScript

- Lenguaje interpretado
- Orientado a Objetos pero basado en prototipos
- Pensado para la Web

`.js`

Entorno de Desarrollo

Entorno de Desarrollo

- Visual Studio Code

Diagramas de Modelado

OMT

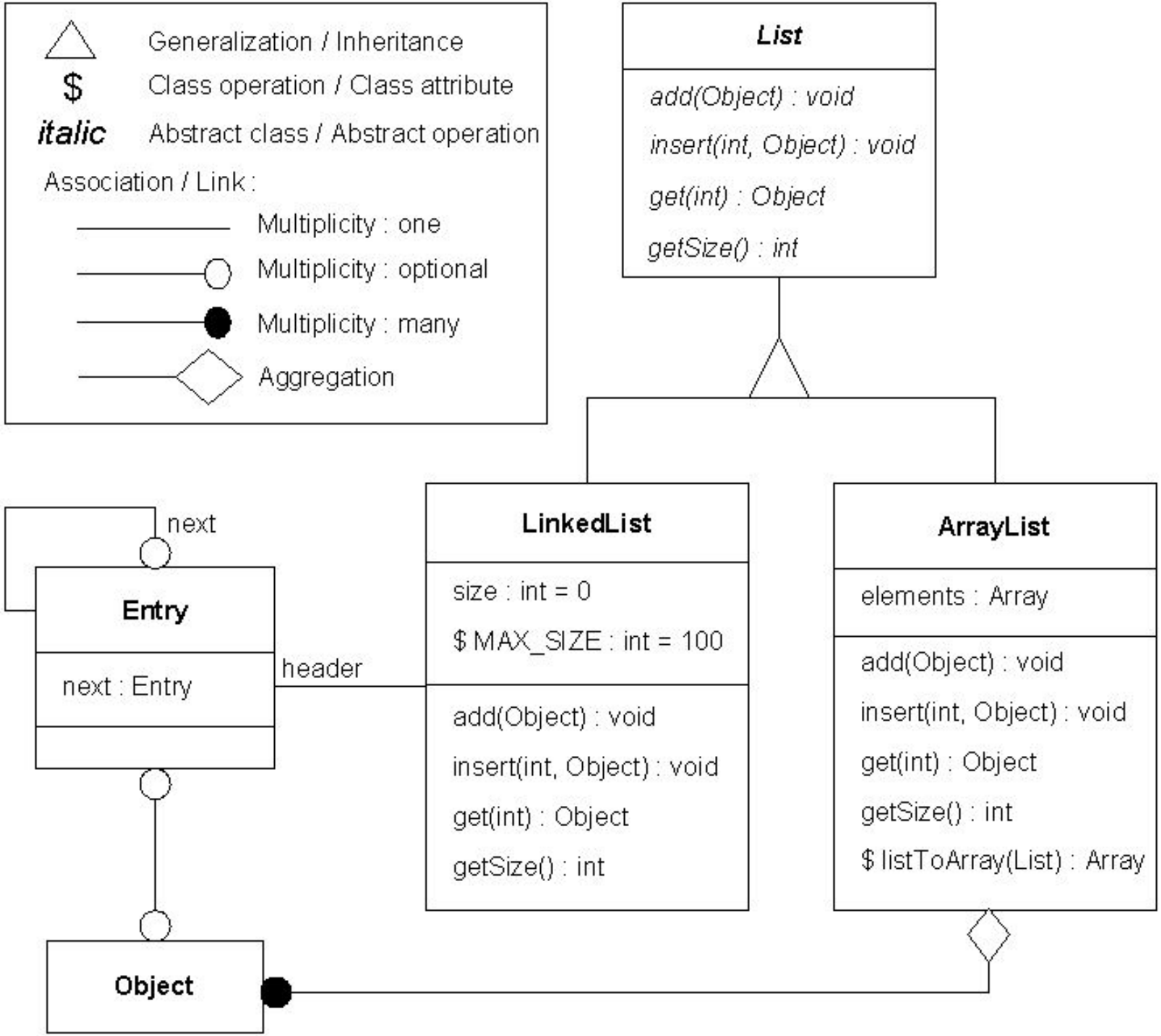
UML

OMT

Object Modeling Techniques

OMT
Object Modeling Techniques

+ Metodología para el análisis orientado a objetos


Generalization / Inheritance
$
Class operation / Class attribute
italic
Abstract class / Abstract operation
Association / Link :
Multiplicity : one
Multiplicity : optional
Multiplicity : many
Aggregation
List
add(Object) : void
insert(int, Object) : void
get(int) : Object
getSize() : int
LinkedList
size : int = 0
$ MAX_SIZE : int = 100
add(Object) : void
insert(int, Object) : void
get(int) : Object
getSize() : int
ArrayList
elements : Array
add(Object) : void
insert(int, Object) : void
get(int) : Object
getSize() : int
$ listToArray(List) : Array
next
Entry
next : Entry
header
Object

UML

Unified Modeling Language

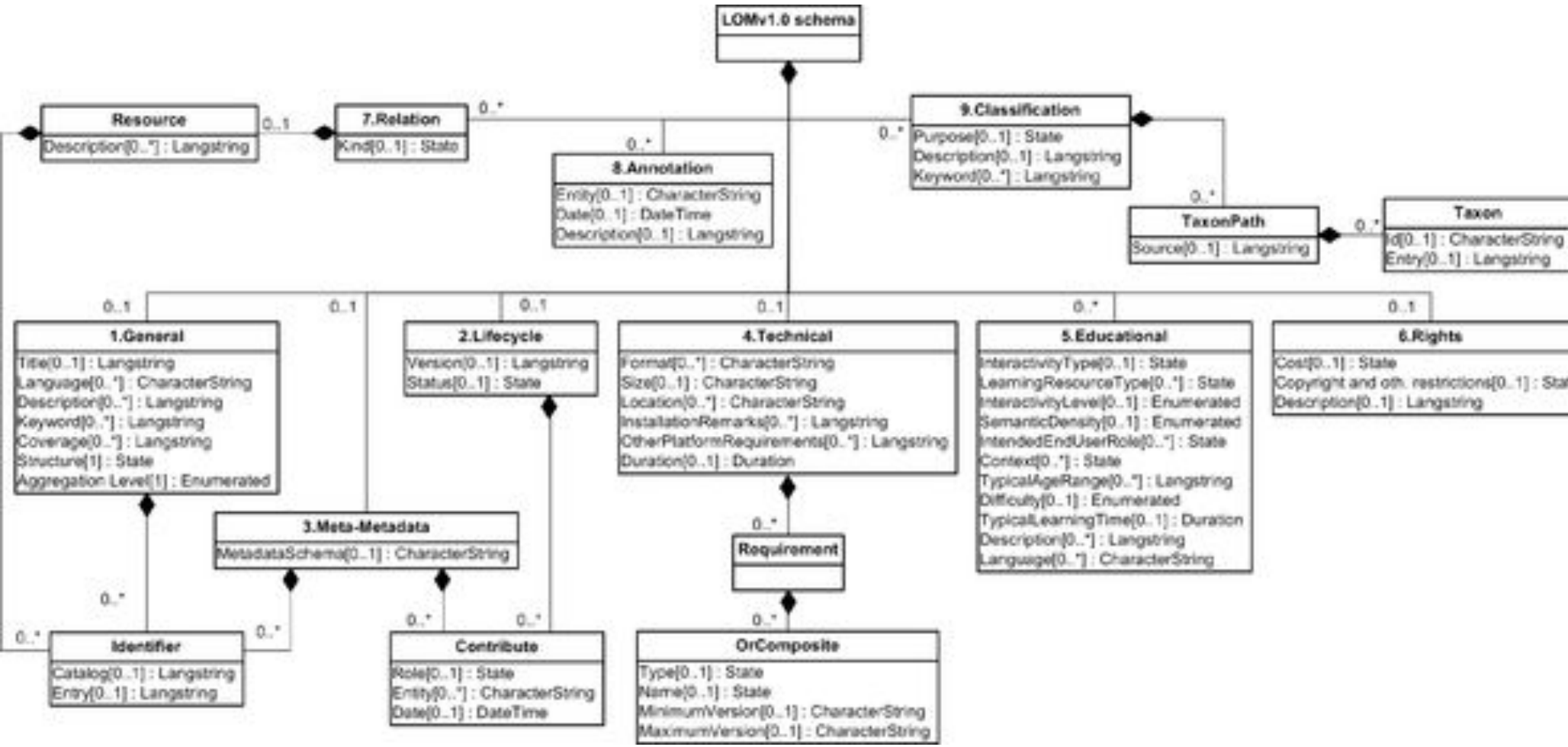
UML

Unified Modeling Language

Lenguaje de Modelado Unificado

UML
Unified Modeling Language

+ Clases
+ Casos de Uso
+ Objetos
+ Actividades
+ Iteración
+ Estados
+ Implementación


LOMv1.0 schema
Resource
Description[0..*] : Langstring
0..1
7.Relation
Kind[0..1] : State
0..*
8.Annotation
Entity[0..1] : CharacterString
Date[0..1] : DateTime
Description[0..1] : Langstring
0..*
9.Classification
Purpose[0..1] : State
Description[0..1] : Langstring
Keyword[0..*] : Langstring
0..*
TaxonPath
Source[0..1] : Langstring
0..*
Taxon
Id[0..1] : CharacterString
Entry[0..1] : Langstring
0..1
1.General
Title[0..1] : Langstring
Language[0..*] : CharacterString
Description[0..*] : Langstring
Keyword[0..*] : Langstring
Coverage[0..*] : Langstring
Structure[1] : State
Aggregation Level[1] : Enumerated
0..1
2.Lifecycle
Version[0..1] : Langstring
Status[0..1] : State
0..1
4.Technical
Format[0..*] : CharacterString
Size[0..1] : CharacterString
Location[0..*] : CharacterString
InstallationRemarks[0..*] : Langstring
OtherPlatformRequirements[0..*] : Langstring
Duration[0..1] : Duration
0..*
5.Educational
InteractivityType[0..1] : State
LearningResourceType[0..*] : State
InteractivityLevel[0..1] : Enumerated
SemanticDensity[0..1] : Enumerated
IntendedEndUserRole[0..*] : State
Context[0..*] : State
TypicalAgeRange[0..*] : Langstring
Difficulty[0..1] : Enumerated
TypicalLearningTime[0..1] : Duration
Description[0..*] : Langstring
Language[0..*] : CharacterString
0..1
6.Rights
Cost[0..1] : State
Copyright and oth. restrictions[0..1] : Sta
Description[0..1] : Langstring
0..1
3.Meta-Metadata
MetadataSchema[0..1] : CharacterString
0..*
Identifier
Catalog[0..1] : Langstring
Entry[0..1] : Langstring
0..*
Contribute
Role[0..1] : State
Entity[0..*] : CharacterString
Date[0..1] : DateTime
0..*
Requirement
0..*
OrComposite
Type[0..1] : State
Name[0..1] : State
MinimumVersion[0..1] : CharacterString
MaximumVersion[0..1] : CharacterString

OMT

1991

UML

1997

UML

1997

Objetos

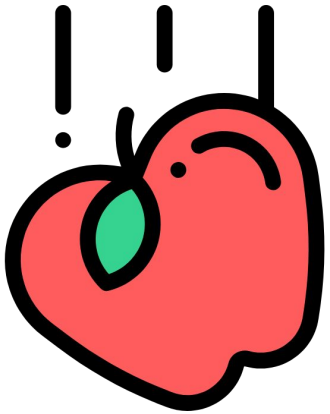
Objetos

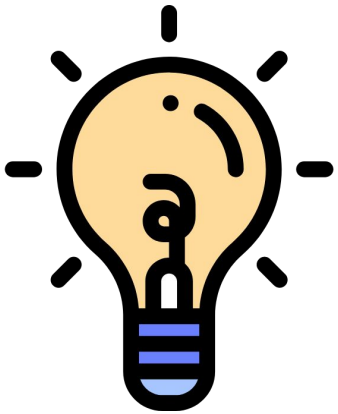
- Cuando tengamos un problema lo primero que debemos hacer es **identificar Objetos**

Objetos son aquellos que tienen propiedades y comportamientos

Objetos

- Pueden ser Físicos o Conceptuales





User

Session

Propiedades
también pueden
llamarse **atributos**
serán sustantivos

Propiedades
también pueden
llamarse **atributos**
serán sustantivos

nombre, tamaño, forma, estado

Comportamientos

serán todas las operaciones del objeto, suelen ser verbos o sustantivo y verbo

Comportamientos

serán todas las operaciones del objeto, suelen ser verbos o sustantivo y verbo

login(), logout(), makeReport()

Perro

Perro

+ nombre
+ color
+ raza
+ altura

Perro

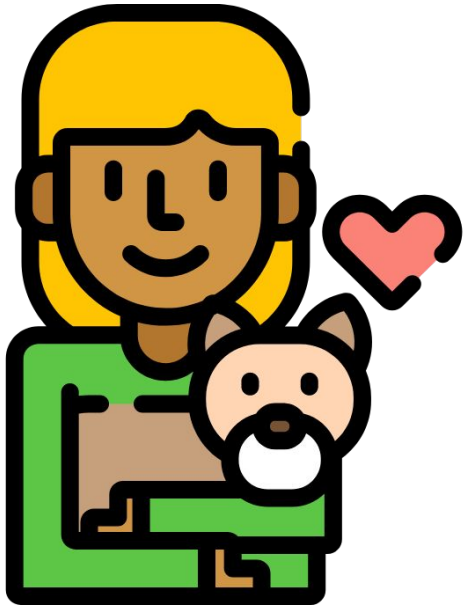
Comportamientos

+ ladrar
+ comer
+ dormir
+ correr

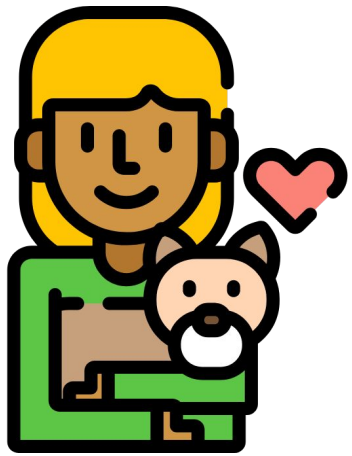
Propiedades

+ nombre
+ color
+ raza
+ altura

Adopciones



Adopciones



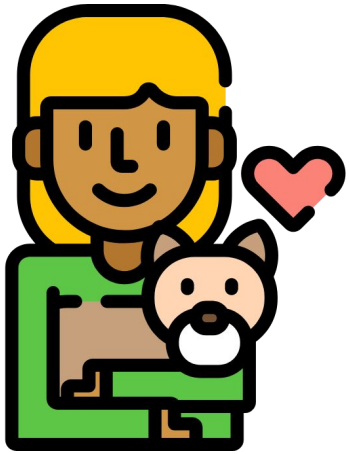
Comportamientos

+ **serAdoptado()**

+ **id**
+ nombre
+ color
+ raza
+ altura

Propiedades

Adopciones

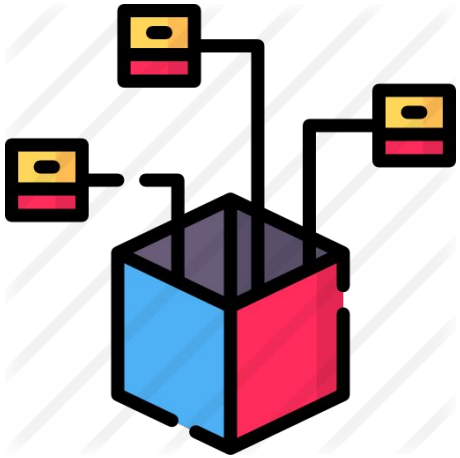


+ **id:** **001**
+ nombre: Franky
+ color: Café
+ raza: French Poodle
+ Altura: 40cm

Clase

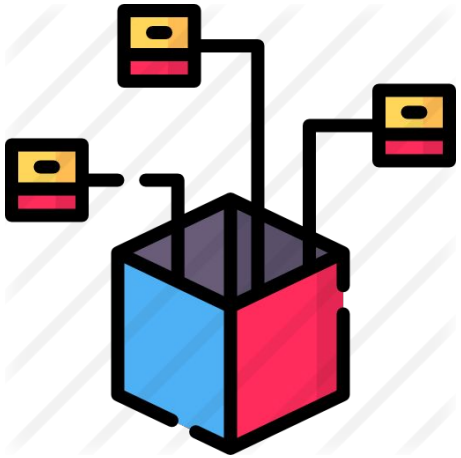
Clase

- Es el modelo sobre el cual se construirá nuestro objeto



Clase

- Las clases me permitirán generar más objetos



Analizar **Objetos** para crear **Clases**

Abstracción

Clases son los modelos sobre los cuales construiremos Objetos

Modularidad

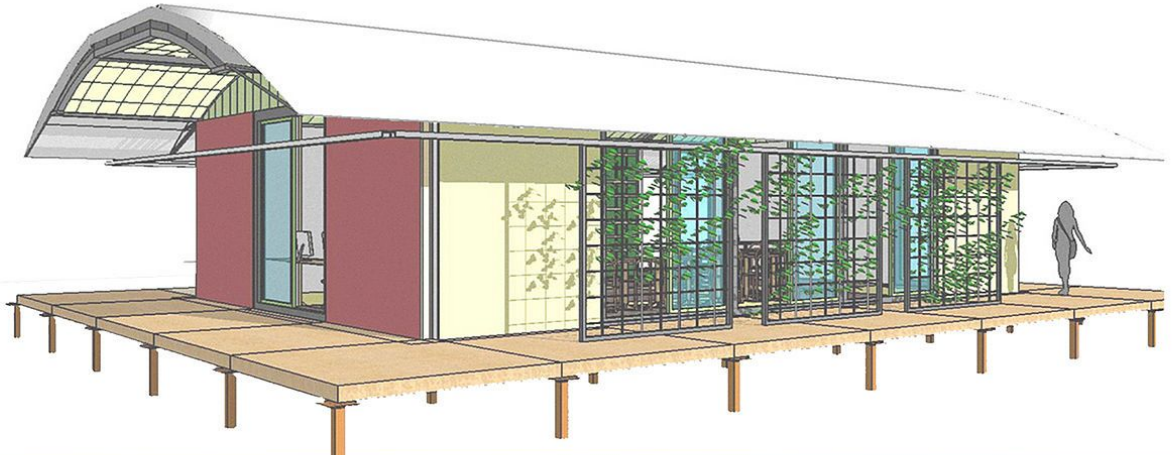
Diseño Modular

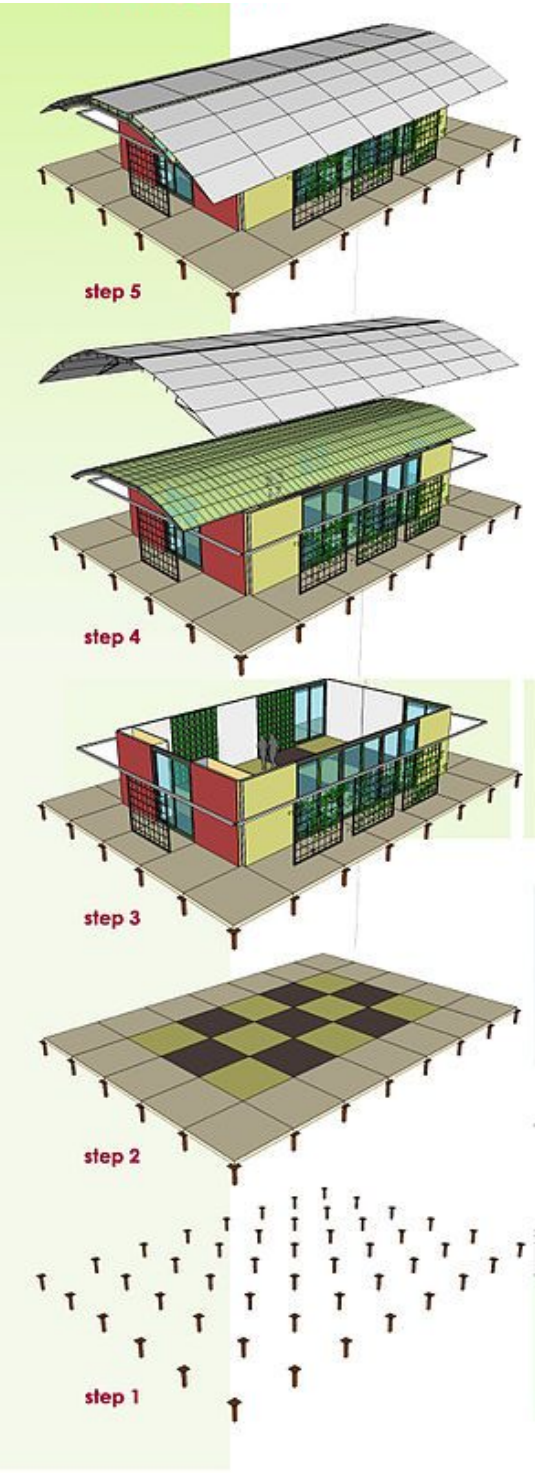
2 Seats + 4 Sides

4 seats + 5 Sides

6 seats + 8 sides

8 seats + 10 sides




step 5
step 4
step 3
step 2
step 1

Estructurada

Modular

Orientado a Objetos

Modular

Orientado a Objetos

+ Reutilizar
+ Evitar colapsos
+ Mantenible
+ Legibilidad
+ Resolución rápida de problemas

Clase

Clase

- Modularidad
- Divide el programa en diferentes partes o módulos / clases
- Separar las clases en archivos

Clases

UML

Nombre Clase
Atributo 1 Atributo 2 Atributo 3 Atributo n
Operación 1 Operación 2 Operación 3 Operación n

Identidad

Estado

Comportamiento

Person
name
walk ()

Java

```
class Person { }
```

Python

```
class Person:
```

JavaScript

```
function Person() { }
```

PHP

```
class Person { }
```

Java

```
class Person {
    String name = “”;
    void walk() { }
}
```

Python

```
class Person:
    name = “”;
    def walk():
```

JavaScript

```
Person.prototype.walk = function (){

}
```

PHP

```
class Person {
    $name = “”;
    function walk() { }
}
```

“

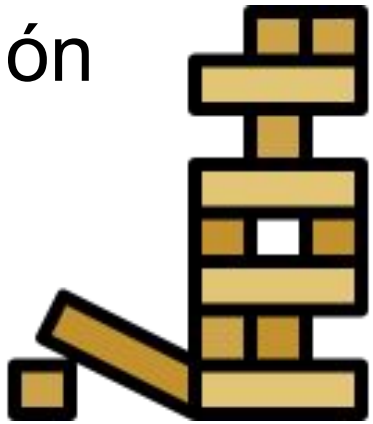
Don’t repeat yourself

”

DRY: *Don't repeat yourself*

- Promueve la reducción de duplicación en programación

- Toda pieza de información **nunca debería ser duplicada** debido a que la duplicación **incrementa la dificultad** en los cambios y evolución



Reutilización

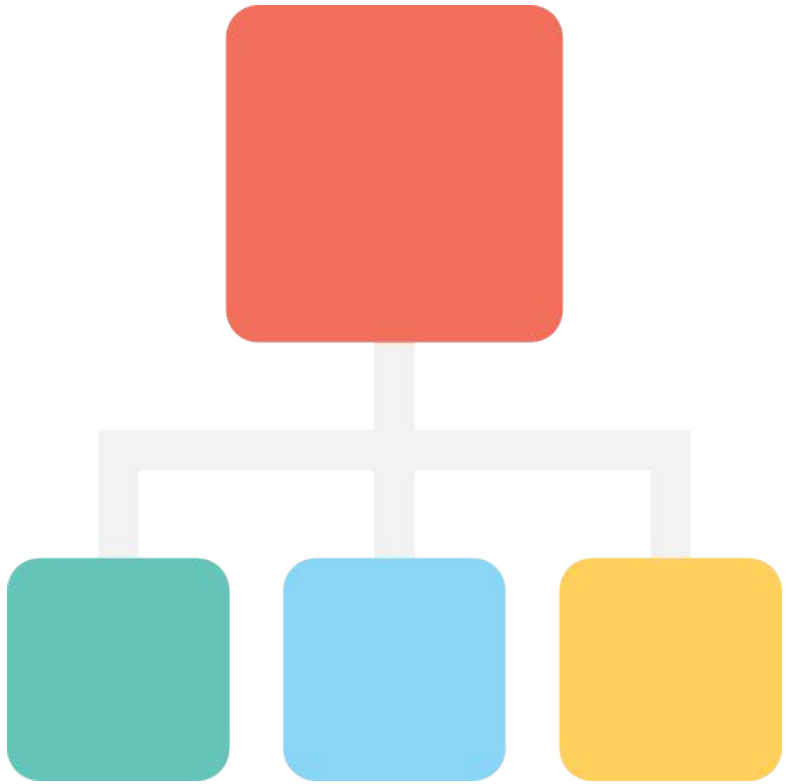
Herencia

Herencia

crearemos nuevas clases a partir de otras

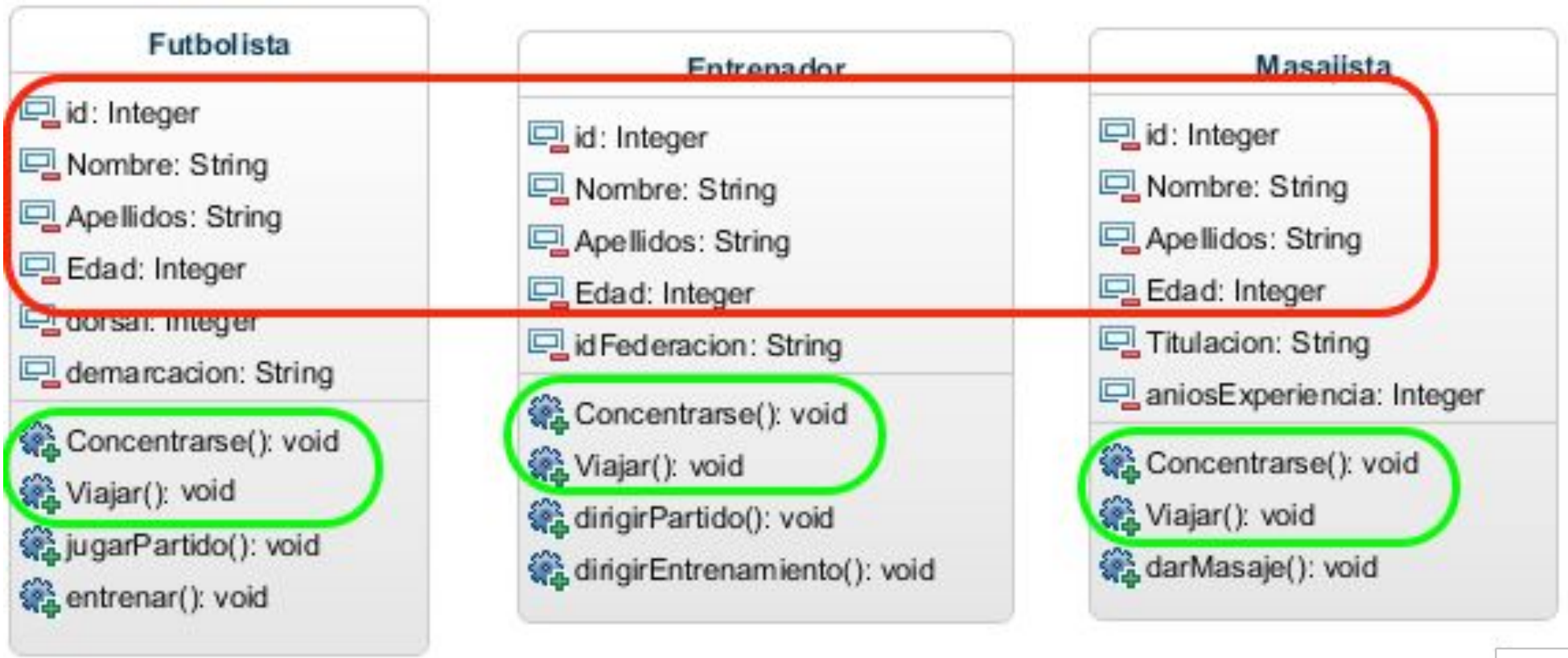
Herencia

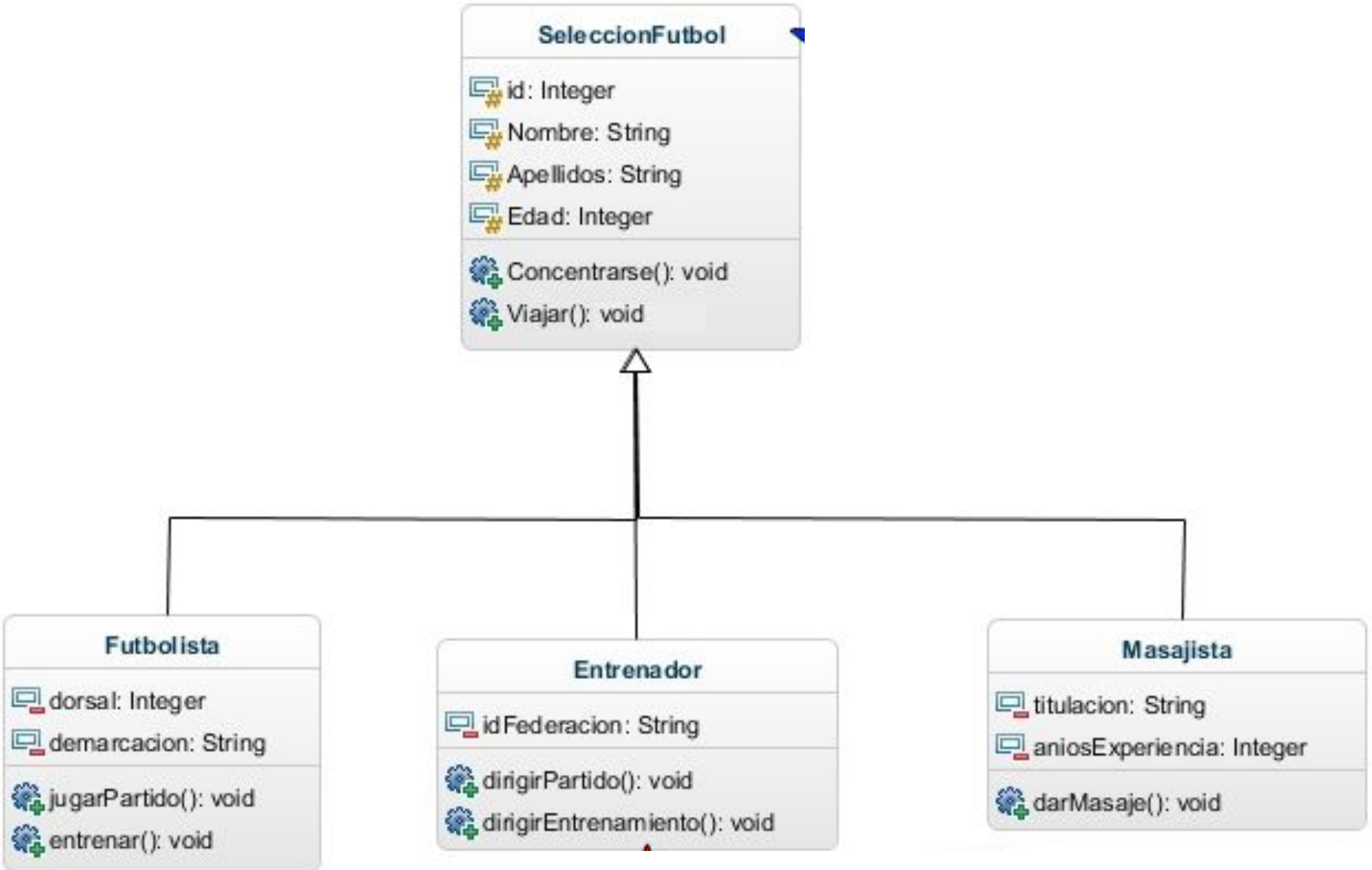
- Se establece una relación **padre e hijo**



Superclase

Subclase


Futbolista
id: Integer
Nombre: String
Apellidos: String
Edad: Integer
demarcacion: String
Concentrarse(): void
Viajar(): void
jugarPartido(): void
entrenar(): void
Entrenador
id: Integer
Nombre: String
Apellidos: String
Edad: Integer
idFederacion: String
Concentrarse(): void
Viajar(): void
dirigirPartido(): void
dirigirEntrenamiento(): void
Masajista
id: Integer
Nombre: String
Apellidos: String
Edad: Integer
Titulacion: String
aniosExperiencia: Integer
Concentrarse(): void
Viajar(): void
darMasaje(): void


SeleccionFutbol
id: Integer
Nombre: String
Apellidos: String
Edad: Integer
Concentrarse(): void
Viajar(): void
Futbolista
dorsal: Integer
demarcacion: String
jugarPartido(): void
entrenar(): void
Entrenador
idFederacion: String
dirigirPartido(): void
dirigirEntrenamiento(): void
Masajista
titulacion: String
aniosExperiencia: Integer
darMasaje(): void

Objeto

Java

```
Person person = new Person();
```

Python

```
persona = Person()
```

JavaScript

```
var person = new Person();
```

PHP

```
$person = new Person();
```

Java

```
Person person = new Person();
```

Python

```
person = Person()
```

JavaScript

```
var person = new Person();
```

PHP

```
$person = new Person();
```

Método constructor

- Dar un estado inicial al objeto
- Tiene el mismo nombre de la clase
- Son los parámetros mínimos que necesita el objeto para que pueda vivir

Java

```
public Person(String name){
   this.name = name;
}
```

Python

```
def __init__(self, name):
   self.name = name
```

JavaScript

```
function Person(name) {
   this.name = name
}
```

PHP

```
public function __construct($name){
   $this->name = name;
}
```

Java

```
Person person = new
Person("Ann");
```

Python

```
person = Person("Ann")
```

JavaScript

```
var person = new
Person("Ann");
```

PHP

```
$person = new Person("Ann");
```

Herencia

Java

```
class Student extends Person
```

Python

```
class Student(Person):
```

JavaScript

```
student.prototype = new Person();
```

PHP

```
class Student extends Person
```

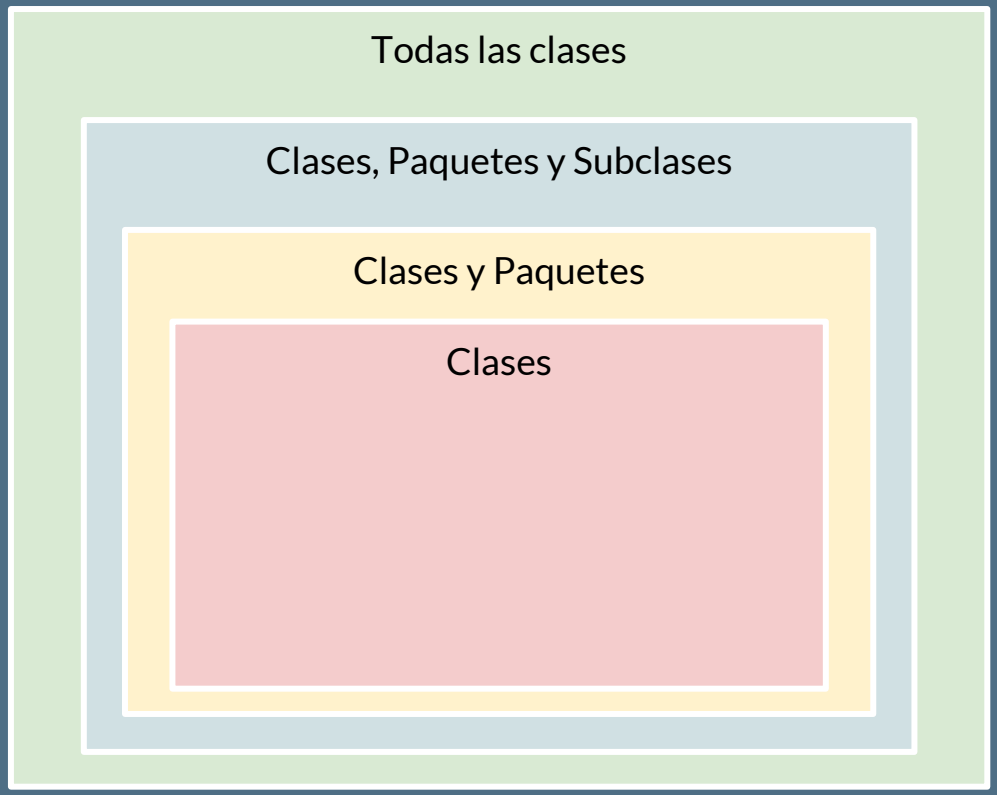
Encapsulamiento

Encapsulamiento

- Para que un dato permanezca inviolable, inalterable, se le asigna un modificador de acceso

public
protected
default
private

public
protected
default
private

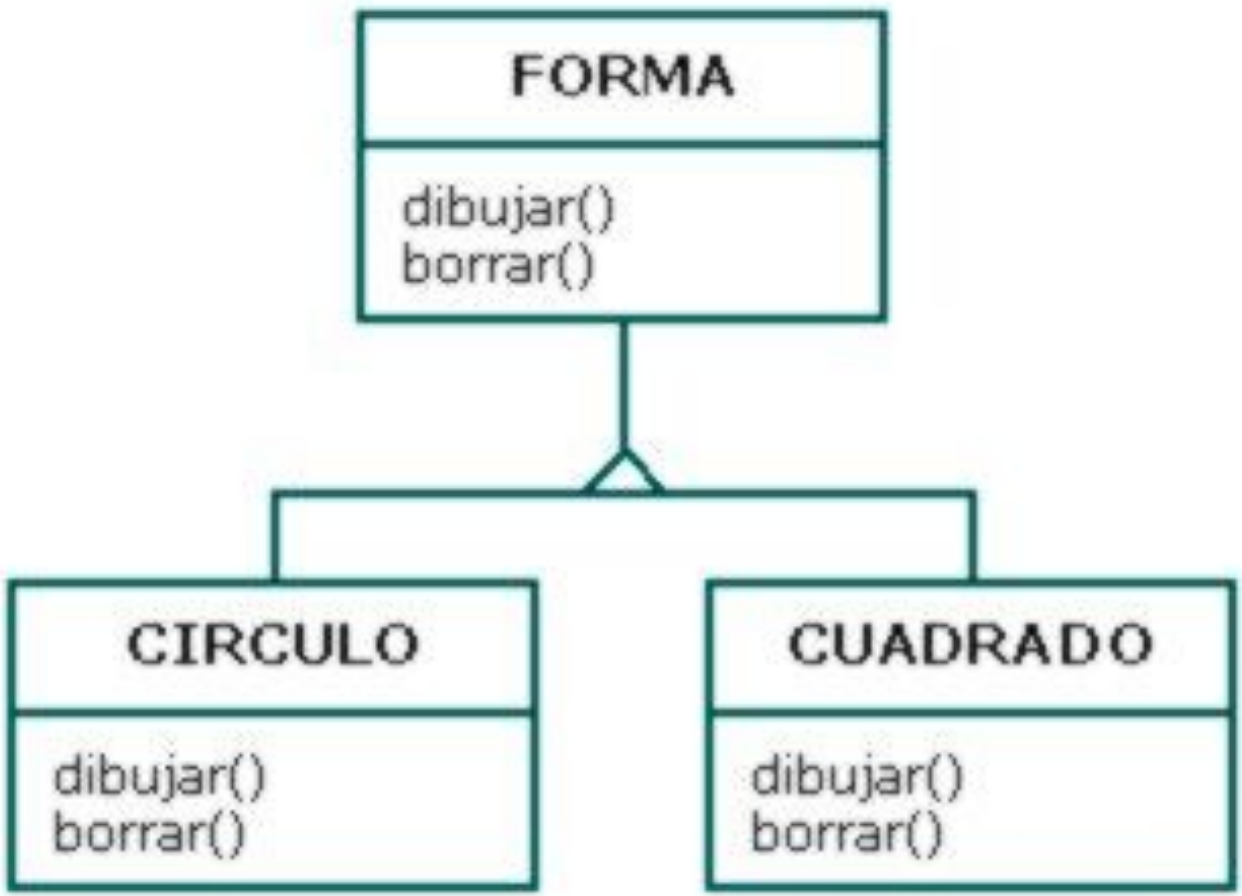


Getters y Setters

Polimorfismo

Polimorfismo

- Muchas formas
- Construir métodos con el mismo nombre pero con comportamiento diferente


FORMA
dibujar()
borrar()
CIRCULO
dibujar()
borrar()
CUADRADO
dibujar()
borrar()