Tema 1: Introducción

Antonio J. Sierra

Indice

Lenguajes de programación

- Modelo computacional: colección de valores y operaciones
- Tipos de modelos computacionales (Paradigma): imperativo, funcional, lógico …
- Computación: aplicación de una secuencia de operaciones a un valor para obtener otro valor
- Programa: especificación de una computación
- Lenguaje de programación: notación para escribir programas
- Sintaxis de un lenguaje de programación: estructura o forma de los programas
- Semántica de un lenguaje de programación: relaciones entre un programa y un modelo de computación
- Pragmática de un lenguaje de programación: grado de éxito con el que un programa cumple sus objetivos tanto en su fidelidad con el modelo de computación subyacente como su utilidad para los programadores

Modelo de Programación.

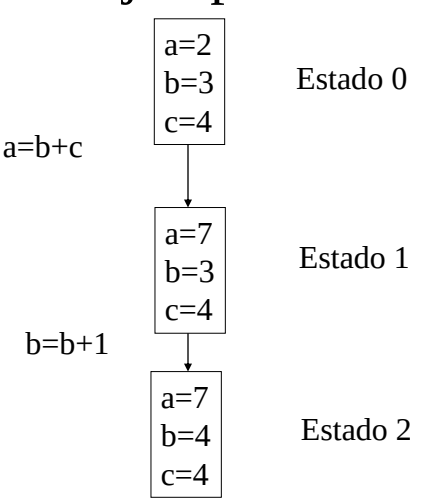
- Un modelo de programación provee (y determina) la visión y métodos de un programador en la construcción de un programa o subprograma.
- Los diferentes paradigmas son el resultado de los distintos estilos de programación y las diferentes formas de pensar en la solución de problemas (con la solución de múltiples “problemas” se construye una aplicación).

Modelo Imperativo

- Describe la programación como una secuencia instrucciones o comandos que cambian el estado de un programa.
- El código máquina en general está basado en el paradigma imperativo.
- Su contrario es el paradigma declarativo.
- En este paradigma se incluye el paradigma procedimental (procedural) entre otros.

Ejemplo

Programa
a=b+c
b=b+1



Programación Estructurada

- La programación se divide en bloques (procedimientos y funciones) que pueden o no comunicarse entre sí.
- Además la programación se controla con secuencia, selección e iteración.
- Permite reutilizar código programado y otorga una mejor compresión de la programación.
- Es contrario al paradigma no estructurado, de poco uso, que no tiene ninguna estructura, es simplemente un "bloque", como por ejemplo, los archivos batch (.bat)

Modelo Declarativo

- No se basa en el cómo se hace algo (cómo se logra un objetivo paso a paso), sino que describe (declara) cómo es algo.
  - En otras palabras, se enfoca en describir las propiedades de la solución buscada, dejando indeterminado el algoritmo (conjunto de instrucciones) usado para encontrar esa solución.
  - Es más complicado de implementar que el paradigma imperativo, tiene desventajas en la eficiencia, pero ventajas en la solución de determinados problemas.

Programación Declarativa

- Usa bloques de construcción como las funciones, la recursión o la equipación de patrones, para especificar más la solución que su cálculo de bajo nivel.
- Tipos:
  - Lenguajes funcionales
  - Lenguajes lógicos

Programación Funcional

- Usan funciones libres de efectos secundarios como bloques primitivos de construcción de programas.
  - Estas funciones pueden aplicarse, construirse y pasarse como argumento a otras funciones.
- Concibe a la computación como la evaluación de funciones matemáticas y evita declarar y cambiar datos.
- En otras palabras, hace hincapié en la aplicación de las funciones y composición entre ellas, más que en los cambios de estados y la ejecución secuencial de comandos (como lo hace el paradigma procedimental).
- Permite resolver ciertos problemas de forma elegante y los lenguajes puramente funcionales evitan los efectos secundarios comunes en otro tipo de programaciones.
  - Haskell, Miranda, Scala, Lisp, Scheme,Ocaml, SAP, Standard ML, Erlang,R, F#

Paradigma lógico

- Se basa en la definición de reglas lógicas para luego, a través de un motor de inferencias lógicas, responder preguntas planteadas al sistema y así resolver los problemas. Ej.: prolog.

Programación Orientado a Objetos

- Basado en la idea de encapsular estado y operaciones en objetos.
- En general, la programación se resuelve comunicando dichos objetos a través de mensajes (programación orientada a mensajes).
- Se puede incluir -aunque no formalmente- dentro de este paradigma, el paradigma basado en objetos, que además posee herencia y subtipos entre objetos. Ej.: Simula, Smalltalk, C++, Java, Visual Basic .NET, etc.
- Su principal ventaja es la reutilización de códigos y su facilidad para pensar soluciones a determinados problemas.

Introducción histórica(I)

- CPL→BCPL → B → C → C++ →Java
- CPL (Combined Programming Language): 1960, basado en ALGOL 60.
- BCPL (Basic Combined Programming Language): Martin Richards, 1966.
- B: Ken Thompson y Dennis Ritchie, reemplazado por C (1969).

Introducción histórica(II)

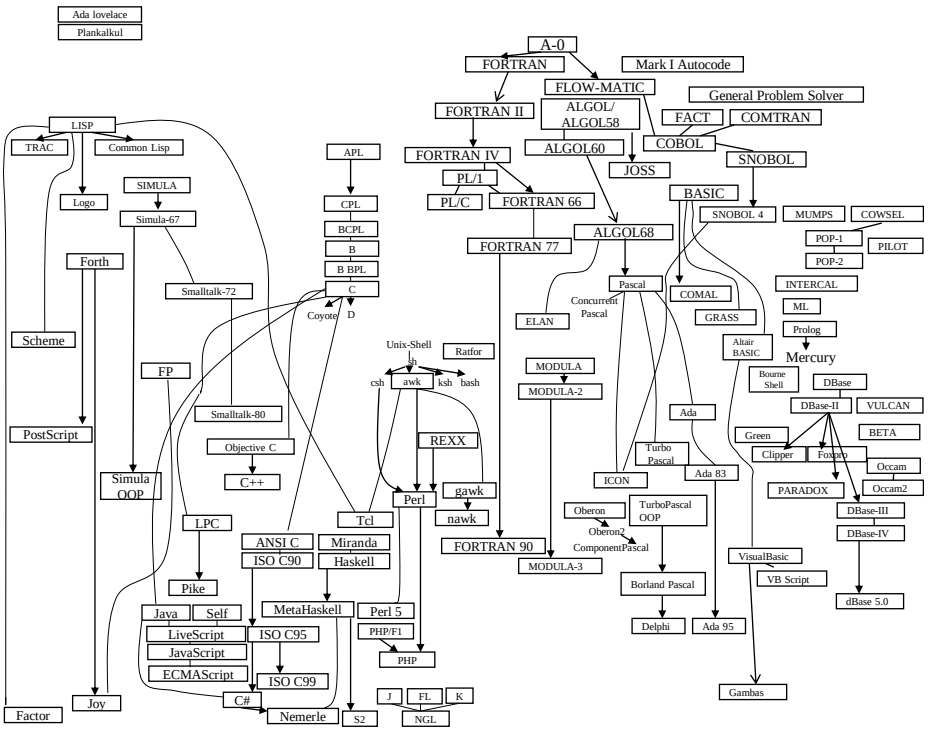
- C: Bell Telephone Laboratories (1972) por Dennis Ritchie para usarlo con Unix
  - Propósito general,
  - Estructurado por bloques,
  - Imperativo,
  - Procedimientos

Introducción histórica(II)

- C++:
  - Bjarne Stroustrup (1979) Bell Labs.
  - Como mejora a C: “C con clases”. C++ in 1983.
  - Las mejoras comenzaron con la adición de clases, funciones virtuales, sobrecarga de operadores, herencia múltiple, plantillas, y manejo de excepciones.
  - C++ fue ratificado como estándar en 1998 como ISO/IEC 14882:1998, la actual versión es de 2003, ISO/IEC 14882:2003.

Java

- Sun Microsystems (1995)
- Sintaxis deriva de C y C++.
- Orientado a Objetos
- Compilado a bytecode. Ejecutado sobre cualquier Java virtual machine (JVM). Sobre cualquier arquitectura.
- Desde 1995 Sun desarrolla e implementa compiladores, máquinas virtuales y librería de clases.

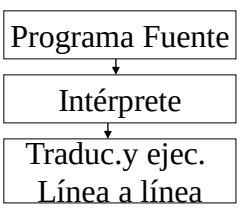


Características de Java

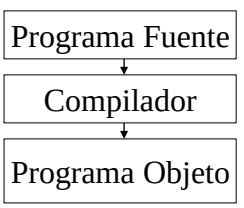
- Simple y seguro
- Portable
- OOP
- Robusto
- Multihilo
- Neutral
- Interpretado
- Rendimiento
- Distribuido
- dinámico

Interprete & Compilador

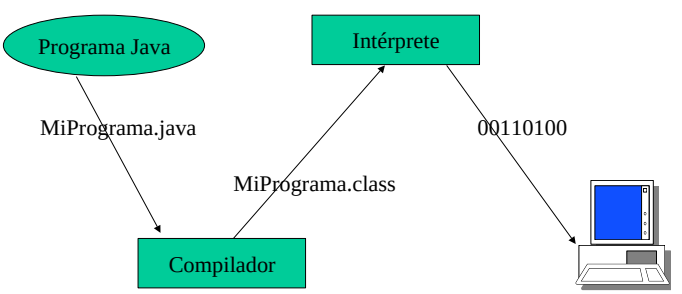
- Un **Intérprete** es un traductor que toma el programa fuente y lo traduce y ejecuta lína a línea.
- Basic, Java, Smalltalk.

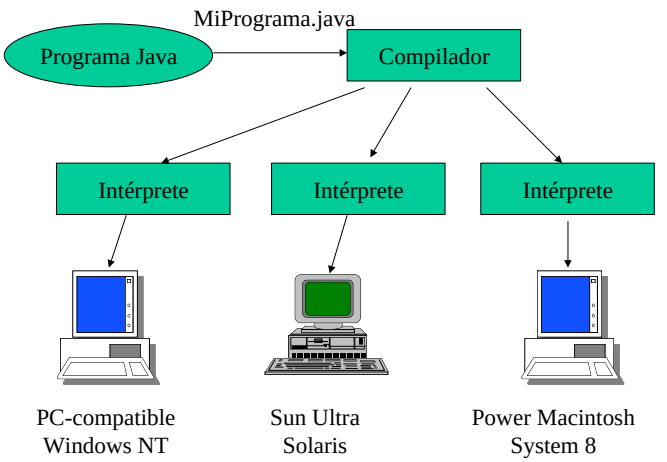


- Un Compilador de un programa que traduce los programas escritos en lenguaje de alto nivel a lenguaje máquina.
- C, C++, Pascal, Fortran, Cobol.



La máquina virtual de Java: Bytecode.

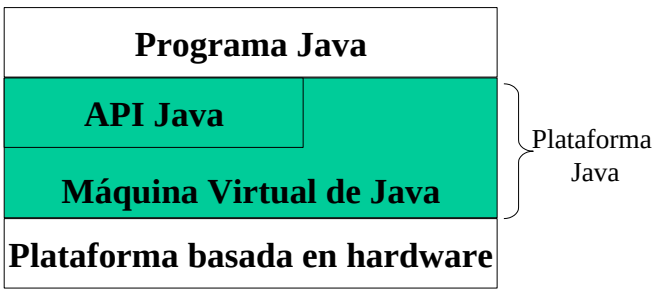


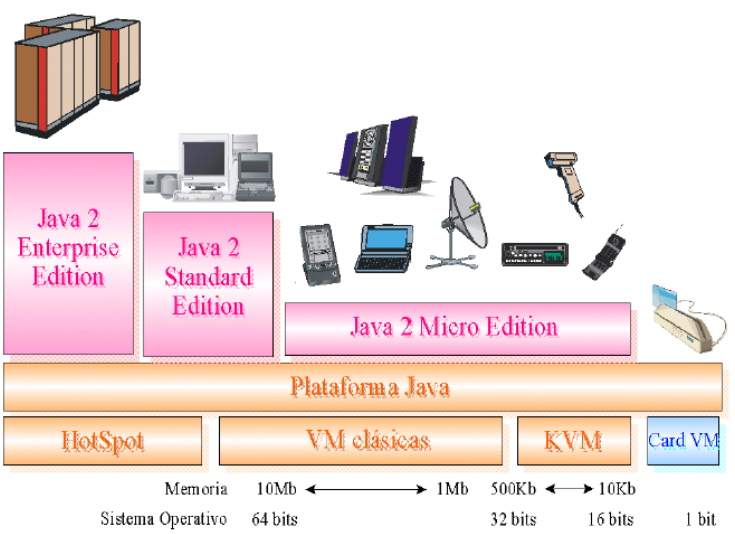


Application Programming Interface (API)

- Una Application Programming Interface (API) es un conjunto de funciones, procedimientos o clases que un sistema operativo, librería o servicio proporciona para soportar peticiones realizadas por un programa de ordenador.
- Son **dependientes** de lenguaje, ya que están disponibles solo en un lenguaje de programación particular. Utilizan la sintaxis y elementos de los lenguajes de programación para hacer que sea adecuada para usarla en un contexto particular.
- Son **independientes** del lenguaje, ya que están escritas en una forma en que pueden ser llamadas desde diferentes lenguajes de programación. Esta característica se conoce como API al estilo servicio, ya que no limita a un proceso particular o sistema y está disponible como una llamada a procedimiento remoto.

La máquina virtual de Java





La máquina virtual de Java

- http://java.sun.com

Palabras reservadas (I)

abstract	Especifica la clase o método que se va a implementar más tarde en una subclase.
boolean	Tipo de dato que sólo puede tomar valores **true** o **false**.
break	Sentencia de control para salirse de los bucles.
byte	Tipo de dato que soporta valores en 8 bits.
byvalue	Reservada para uso futuro.
case	Se utiliza en las sentencias **switch** para indicar bloques de texto.
cast	Reservada para uso futuro.
catch	Captura las excepciones generadas pro las sentencias **try**.
char	Tipo de dato que puede soportar caracteres Unicode sin signo en 16 bits.
class	Declara una clase nueva.
const	Reservada para uso futuro.
continue	Devuelve el control a la salida de un bucle.
default	Indica el bloque de código por defecto en una sentencia **switch**.
do	Inicia un bucle del tipo **do-while.**
double	Tipo de dato que soporta números en coma flotante, 64 bits.
else	Indica la opción alternativa en una sentencia **if**.
extends	Indica que una clase es derivada de otra o de una intefaz.
final	Indica que una variable soporta un valor constante o que un método no se sobrescribirá.

Palabras reservadas (II)

finally	Indica un bloque de código en una estructura **try-catch** que siempre se ejecutará.
float	Tipo de dato que soporta un número en coma flotante de 32 bits.
for	Utilizado para iniciar un bucle **for**.
future	Reservada para uso futuro.
generic	Reservada para uso futuro.
goto	Reservada para uso futuro.
if	Evalúa si una expresión es verdadera o falsa y la dirige adecuadamente.
implements	Especifica que una clase implementa una interfaz.
import	Referencia a otras clases.
inner	Reservada para uso futuro.
instanceof	Indica si un objeto es una instancia de una clase específica o implementa una interfaz específica.
int	Tipo de dato que puede soportar un entero con signo de 32 bits.
interface	Declara una interfaz.
long	Tipo de dato que soporta un entero de 64 bits.
native	Especifica que un método está implementado con código nativo (específico de la plataforma).
new	Crea objetos nuevos.
null	Indica que una referencia no se refiere a nada.
operator	Reservada para uso futuro.
outer	Reservada para uso futuro.

Palabras reservadas (y III)

package	Declara un paquete Java.
private	Especificador de acceso que indica que un método o variable sólo puede ser accesible desde la clase en la que se está declarando.
protected	Especificador de acceso que indica que un método o variable solo puede ser accesible desde la clase en la que está declarado (o una subclase de la clase en la que está declarada u otras clases del mismo paquete).
public	Especificador de acceso utilizado para clases, interfaces, métodos y variables que indican que un tema es accesible desde la aplicación (o desde donde la clase defina que es accesible).
rest	Reservada para uso futuro.
return	Envía control y posiblemente devuelve un valor desde el método que fue invocado.
short	Tipo de dato que puede soportar un entero de 16 bits.
static	Indica que una variable o método es un método de una clase (más que estar limitado a un objeto particular).
super	Se refiere a una clase base de la clase utilizada en un método o constructor de clase.
switch	Sentencia que ejecuta código basándose en un valor.
synchronized	Especifica secciones o métodos críticos de código multihilo.
this	Se refiere al objeto actual en un método o constructor.
throw	Crea una excepción.
throws	Indica qué excepciones puede proporcionar un método.
transiente	Especifica que una variable no es parte del estado persistente de un objeto.
try	Indica un bloque de código que es comprobado para las excepciones.
var	Reservado para uso futuro.
void	Especifica que un método no devuelve ningún valor.
volatile	Indica que una variable puede cambiar de forma asíncrona.
while	Inicia un bucle **while**.

Introducción a la programación orientada a objetos (OOP)

- Es un paradigma de programación que usa “objetos” y sus interacciones para diseñar aplicaciones y programas de ordenador.
- Esta técnica de programación incluye
  - Encapsulación
  - Modularidad
  - Polimorfismo, y
  - Herencia.

Conceptos de la OOP

- Clase
- Objeto
- Instancia
- Método
- Paso de Mensaje
- Herencia
- Abstracción
- Encapsulado
- Polimorfismo

Clase (I)

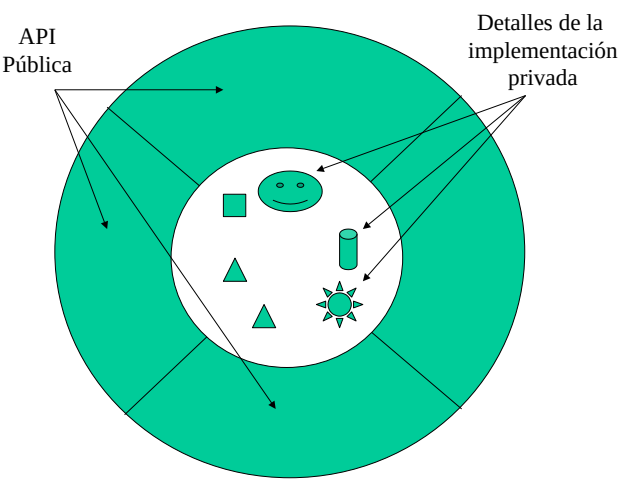
- Define la abstracción de las cosas (objetos), incluye sus **estados** o **características** (atributos, campos) y sus **propiedades** (las cosas que puede hacer, o métodos, operaciones).
- Se podría decir que una clase es un plano o molde que describe la naturaleza de algo.
- Ejemplo: la clase **`Perro`** podría considerar la raza, color (**características**), y la habilidades de ladrar y sentarse (**propiedades**).

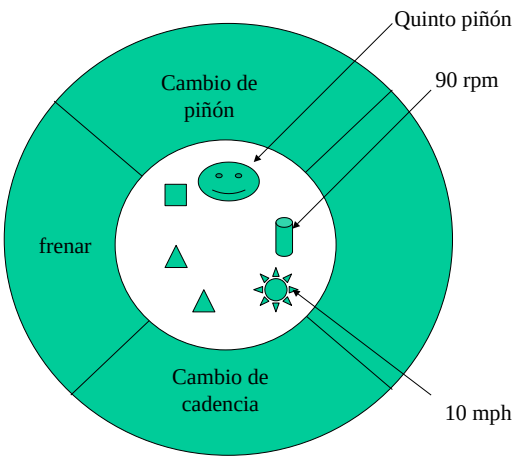
Clase (y II)

- Las clases proporcionan modularidad y estructura en OOP.
- Una clase debería normalmente ser reconocido por una persona del dominio del problema que no sea programador.
- El significado de la clase debería tener sentido en el contexto al que se le da significado.
- El código de una clase debería ser relativamente autocontenido (normalmente usando **encapsulación**).
- Las propiedades y métodos definidos en una clase se conocen como **miembros**

Objeto

- Es un ejemplar de una clase.
- La clase **`Perro`** define todos los posibles perros mediante un listado de las características y propiedades se puede tener el objeto **`Lassie`**, que es un perro particular, con versiones particulares de las características.
- Un **`Perro`** tiene un pelo. **`Lassie`** tiene el pelo de color marrón y blanco.





Instancia

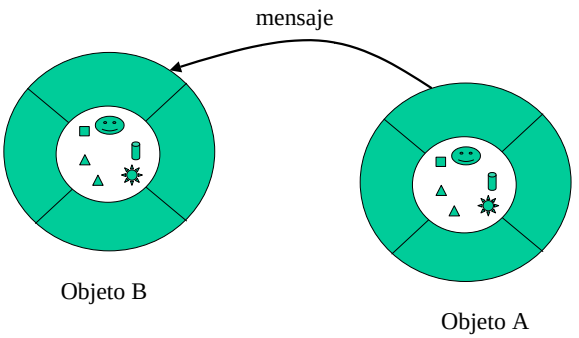
- Una instancia es el objeto creado a partir de una clase en tiempo de ejecución.
- El objeto **`Lassie`** es una instancia de la clase **`Perro`**.
- El conjunto de valores de los atributos del objeto particular se conocen como **estados**.
- El objeto consta de **estados** y **propiedades** que están definidas en la clase de objetos.

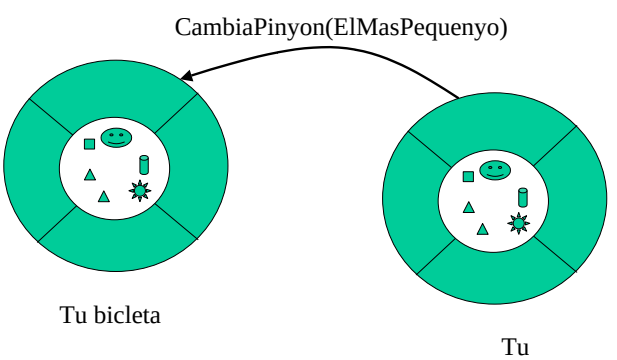
Método

- Son las habilidades de un objeto.
- En un lenguaje, los métodos son verbos.
- **`Lassie`** es un **`Perro`**, que tiene la habilidad de ladrar. Por tanto, **`ladrar()`**, es un método de **`Lassie`**.
- Podría tener otros métodos también como, **`sentar()`**, **`comer()`**, **`caminar()`**, o **`correr()`**.
- Un método afecta solo a un objeto en particular. Todos los perros ladran, pero se necesita un solo perro concreto para que ladre.

Paso de Mensajes

- Es le proceso mediante el cual un objeto envía datos a otro objeto o pide a otro objeto que invoque a un método.
- En los lenguajes de programación es crear una interfaz.
- Ejemplo, el objeto llamador **`Antonio`** podría decir al objeto **`Lassie`** que se siente mediante el paso del mensaje **`sentar`**, que invoca el método **`sentar`** de **`Lassie`**.
- La sintaxis varía según los lenguajes de programación. [Lassie sit] en Objective-C. En Java el mensaje a nivel de código corresponde al "método llamado". Algunos lenguajes dinámicos usan otros mecanismos.


mensaje
Objeto B
Objeto A


CambiaPinyon(ElMasPequenyo)
Tu bicleta
Tu

Abstracción

- Abstracción es simplificar la realidad compleja mediante el modelado de una clase apropiada al problema, y trabajar al nivel más apropiado de herencia para un aspecto concreto del problema.
- Por ejemplo, **`Lassie`** el **`Perro`** podría ser tratado como un perro durante mucho tiempo, un **`Collie`** (su raza) cuando se necesite acceder a atributos o propiedades especificas de esa raza, y como un **`Animal`** cuando se encuentra en una tienda de animales.
- La abstracción se consigue también mediante la "composición". Por ejemplo, una clase **`Coche`** debería tener los objetos **`Motor`**, **`Ruedas`**, **`Llantas`**, y muchos más componentes. No necesitamos los componentes sino la interfaz.

Encapsulación (I)

- La Encapsulación está relacionada con los detalles de una clase de objetos que envía mensajes a él.
- La clase **`Perro`** tiene un método , **`ladrar()`**. El códiog del método **`ladrar()`** define exactamente como sucede (por ejemplo, **`inhala()`** y **`exhala()`**). **`Pedro`** el amigo de **`Lassie`** , no necesita saber como ladra.
- La encapsulación se consigue especificando qué clases podría usar los miembros de un objeto.
- El resultado es que cada obeto expone a cualquier clase un cierto número de **interfaces**, estos miembros pueden acceder a esta clase.

Encapsulación (II)

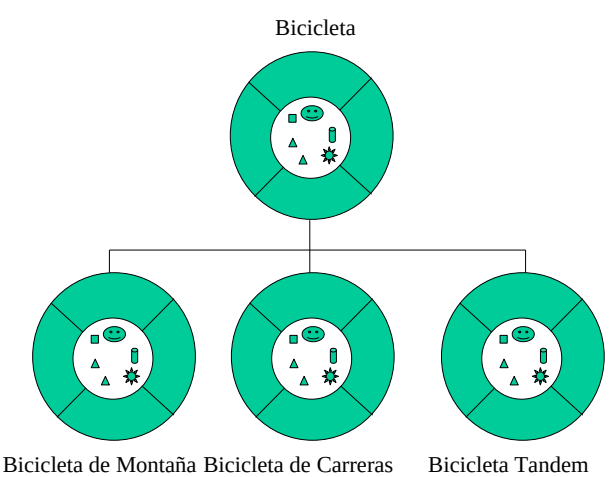
- La encapsulación previene a los clientes de una interfaz depender de las partes de la implementación, facilitando los cambios futuros.
- Los miembros a menudo se especifican como **`public`**, **`protected`** o **`private`** determinado si están disponibles para todas las clase, subclases o solo la definición de las clases.
- Java usa un acceso por defecto que es dentro del mismo paquete, C# y VB.NET reserva algunos miembros a clases en el mismo montaje usando **internal** (C#) o **Friend** (VB.NET). Eiffel y C++ permiten especificar que clases pueden acceder a cualquier miembro.

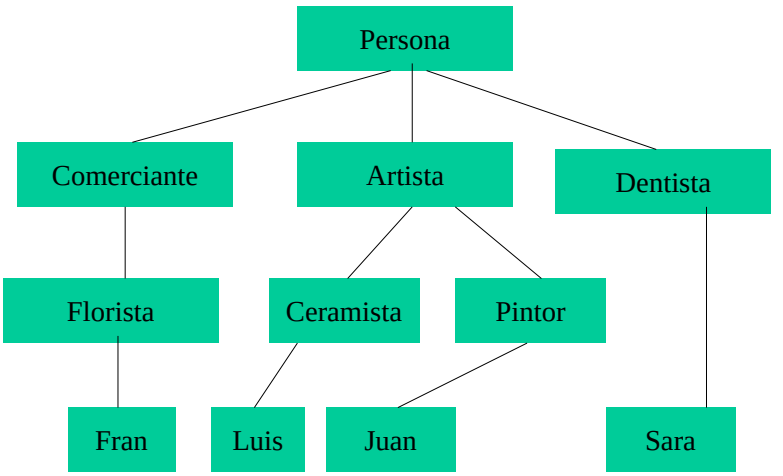
Herencia

- Las 'Subclases' son versiones más especializadas de una clase, que **hereda** atributos y **propiedades** de las clases **padres**, y pueden introducir las suyas propias.
- Por ejemplo, la clase **`Perro`** podría tener las subclases **`Collie`**, **`Chihuahua`** y **`GoldenRetriever`**.
- En este caso, **`Lassie`** debería ser una instancia de **`Collie`**.
- Suponga que la clase **`Perro`** define un método **`ladrar()`** y la propiedad **`colorPelo`**. Cada una de sus subclases también lo heredarán. El programador solo deberá escribir una sola vez el código.
- Las subclases pueden alterar las propiedades tratadas.

Herencia Múltiple

- Es una herencia de más de una clase antecesora, con las antecesoras sin ser antecesoras de las otras.
- Ejemplo, se declaran las clases **`Perro`** y **`Gato`**, y el objeto **`LosDos`**, que se crea de los anteriores con las propiedades de ambos.
- No siempre se puede realizar.
- Es difícil de implementar.
- Es difícil de usar.





Polimorfismo

- Permite al programador tratar miembros de las clases derivadas como los miembros de las clases padres.
- (en OOP) Es la habilidad de los objetos a responder con llamadas a **con el mismo nombre**, cada uno con un propiedades específicas, dependiendo de los tipos de datos de la llamada.
- Un método o un operador (tal y como +, - o *) puede estar asociado de forma abstracta a varias situaciones diferentes.
- La herencia puede originar sobreescritura.
- Ejemplo el operador '+' puede realizarse para varias funciones dependiendo de la implementación para sumar enteros, sumar reales, concatenar listas, o concatenar cadenas.
- La mayoría de los lenguajes OOP soportan algún nivel de polimorfismo.

Diferencias entre C/C++ y Java (I)

- Java no tiene punteros
- Java no incluye estructuras ni uniones
- No permite la sobrecarga de operadores
- No tiene directiva de preprocesado
- Java no realiza ninguna conversión de tipos automática que signifique una pérdida de precisión
- Todo código está encapsulado en una clase
- No se permiten argumentos por defecto
- No permite la herencia múltiple

Diferencias entre C/C++ y Java (y II)

- Java no tiene destructores (incorpora el método `finalize()`). Java no tiene el operador **delete**. (incorpora **new** como C++, pero no **delete**)
- Java no utiliza **typedef**
- En Java no es posible declarar enteros sin signo.
- Java no incluye la sentencia **goto**
- Los operadores << y >> no están sobrecargados en operaciones de E/S
- Los objetos sólo se pasan por referencia. (C++, los objetos se pueden pasar por valor o por referencia)

```
// HelloWorld.java
public class HelloWorld {
 public static void main(String[] args)
 {
  System.out.println("Hello, world!");
 }
}
```

Applet

```
// Hello.java
import java.applet.Applet;
import java.awt.Graphics;

public class Hello extends Applet {
 public void paint(Graphics gc) {
   gc.drawString("Hello, world!",
   65, 95);
 }
}
```

```
<!DOCTYPE HTML PUBLIC
 "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
 <!-- Hello.html -->
 <html>
  <head>
   <title>Hello World Applet</title> </head>
  <body>
  <applet code="Hello" width="200" height="200">
  </applet>
</body>
</html>
```