Lenguaje Java, Clases, Subclases, Sobrecarga y Polimorfismo

Álvaro Rojas V.

UTFSM

Octubre 2024-2

Table of Contents

Lenguaje Java

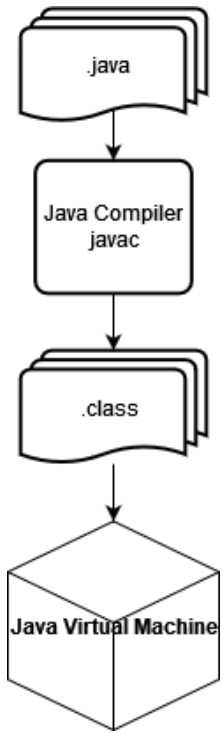
A diferencia de los lenguajes anteriores, en **Java** todo, pero absolutamente todo son **Clases**. Para crear un programa en Java, es necesario crear una clase principal que contenga el método `public static void main(string[] args)`. Se le tiene que avisar a la máquina virtual cuál es la clase que contenga este método.
Luego, Las demás clases e interfaces deben estar contenidas en sus propios archivos `.java`. Cada clase puede contener sus atributos y métodos. La máquina virtual sabrá como acceder a las clases cuando estén en el mismo directorio.

Compilación Java

Java se le denomina un lenguaje compilado híbrido. Primero el **Java Compiler**, denominado *javac* se encarga de compilar el código en *.java* a Bytecode, `.class`. Luego son estos archivos que son pasados a la JVM para ser ejecutado por la máquina.

```
Compiler:
    javac -d classes *.java
RunClass:
    java -classpath classes Main_Class
CompressClass:
    jar cfe Name.jar Main_Class -C
    classes *.class
```

(a) Comando javac, java y jar



(b) Proceso Compilado y Ejecución Java

Estructura de Clases

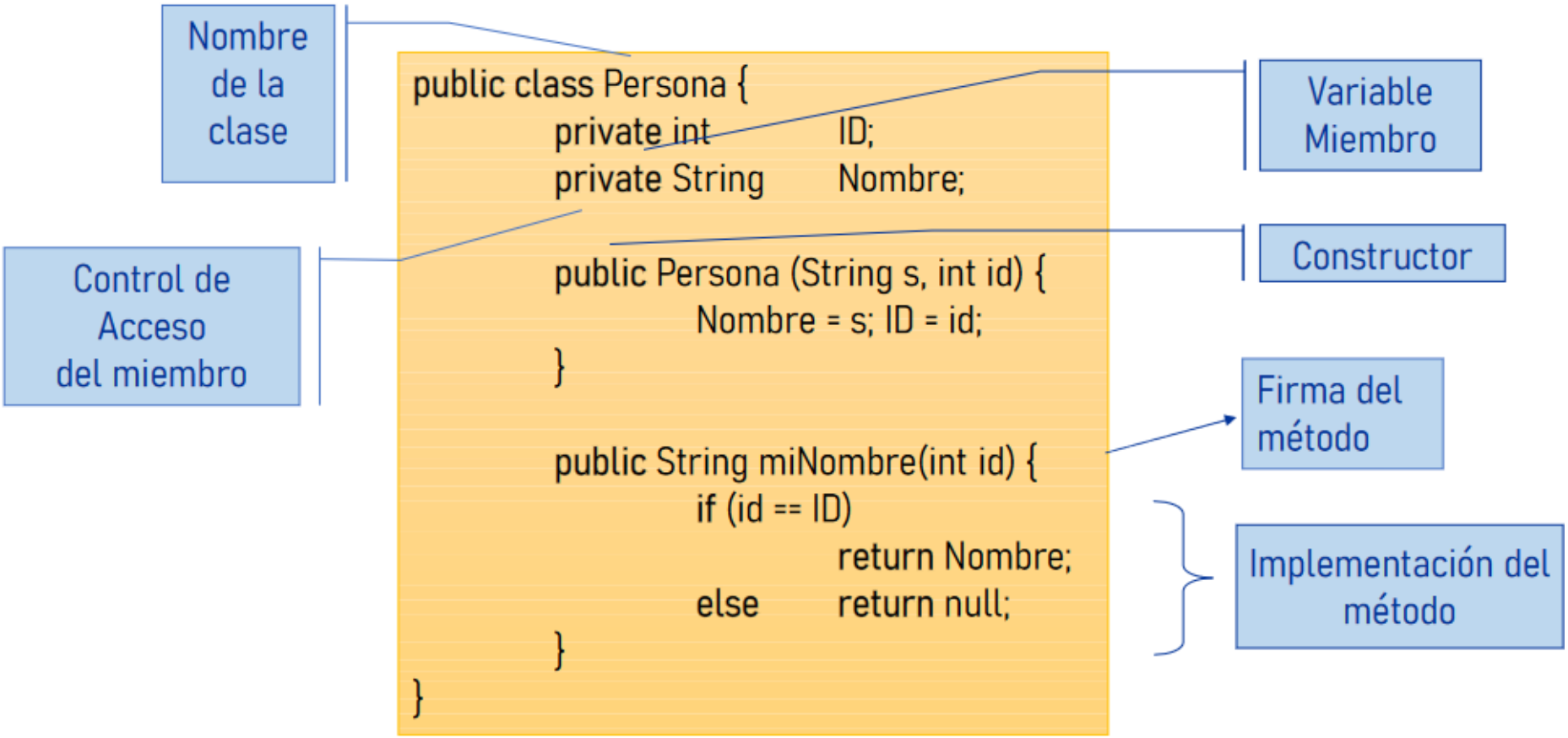


Figure: Estructura de una Clase

El diagrama de clases UML

El diagrama de clases en lenguaje unificado de modelamiento (UML) es un tipo de un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, métodos, y las relaciones entre los objetos. En el contexto de la tarea, este modela el sistema del programa esperado.

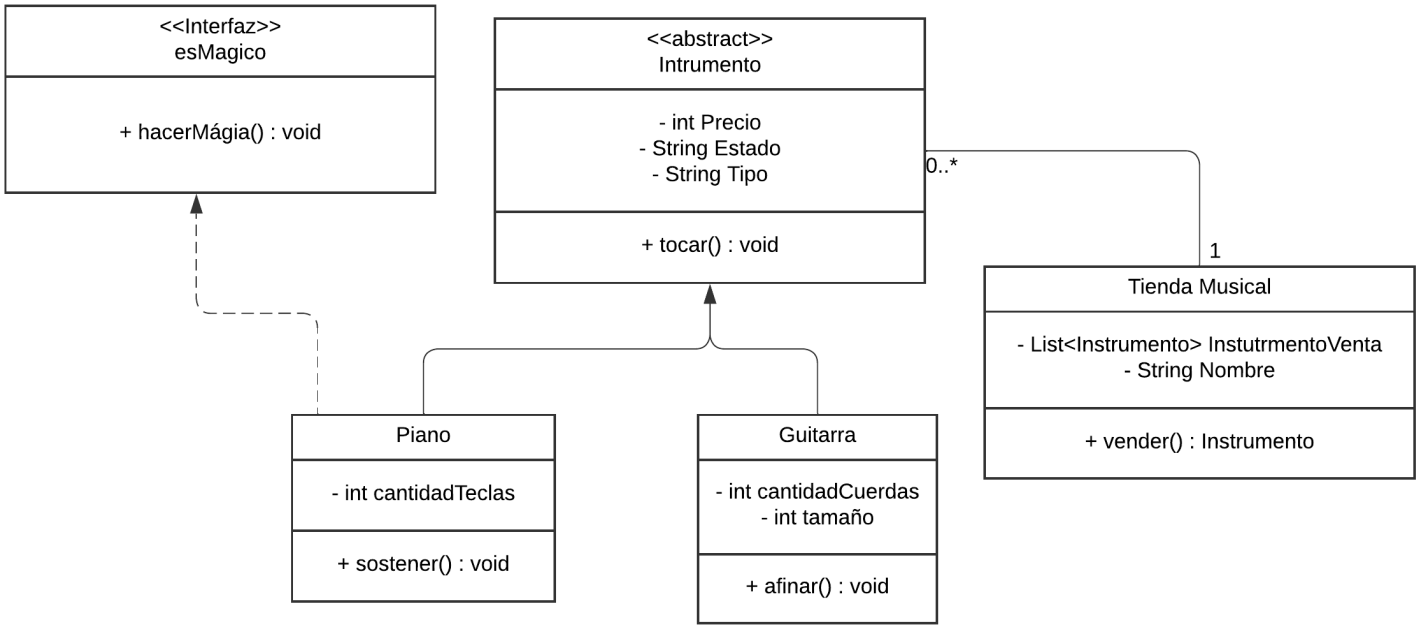


Figure: Ejemplo un diagrama de clases de una tienda musical.

Herencia y Subclase

La herencia permite reutilizar métodos y atributos ya existentes de otra clase. Se puede decir que una clase **extiende** a otra clase. Formando una relación subclase y superclase.

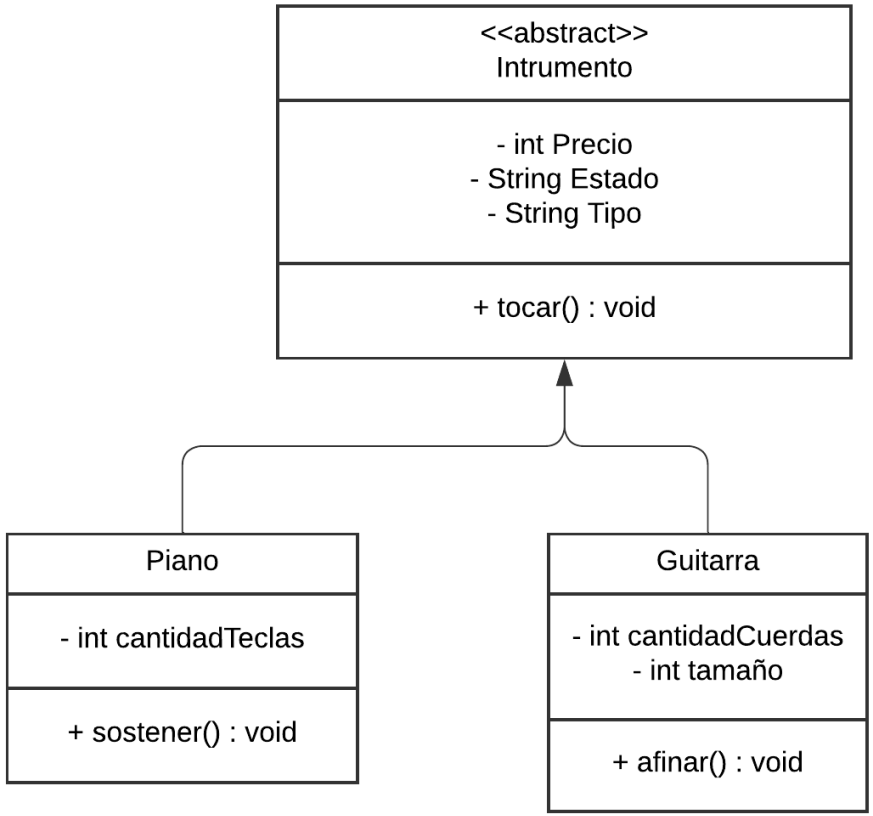


Figure: Instrumento es superclase de Piano y Guitarra

Herencia y Subclase, implementación

Realizar una herencia, referida como extensión, se utiliza la palabra clave `extends className` en el nombre de la clase. En java, solamente puede extender una clase.

```java
public abstract class
    Instrumento {
    private int Precio;
    private String Estado;
    private String Tipo;

    public void tocar()
    {
        /* Comportamiento
    Abstracto de tocar*/
    }
}
```

(a) Instrumento.java

```java
public class Piano extends
    Instrumento{
    private int
    cantidadTeclas;

    public void sostener()
    {
        /* Comportamiento
    Pedal Sostener */
    }
}
```

(b) Piano.java

Redefinición/Polimorfismo de Métodos

Gracias a la herencia, permite realizar redefinición a los métodos de una superclase. Es decir, redefinir la implementación de los métodos ya existente de una superclase. Permite modificar el comportamiento de un método para adaptarse a las necesidades de la subclase.

```java
public class Guitarra {
    private int
    cantidadCuerdas;
    public void tocar()
    {
    /* Comportamiento tocar
     para la Guitarra*/
    }
}
```

(a) Guitarra.java

```java
public class Piano extends
    Instrumento {
    private int
    cantidadTeclas;
    public void tocar()
    {
    /* Comportamiento tocar
     para el Piano*/
    }
}
```

(b) Piano.java

Se puede invocar el método redefinido desde la subclase con la referencia

Sobrecarga

La sobrecarga permite definir más de un método con el mismo nombre, pero con **diferente firma**. Esto permite cambiar el comportamiento dependiendo del tipo de parámetro que recibe un método/función.

```
public class Guitarra {
    private int cantidadCuerdas;
    public void tocar(int cuerda) {
    /* Tocar cuerda*/
    }
    public void tocar(int cuerda_1, int cuerda_2) {
    /* Tocar cuerda 1 y 2 a la vez*/
    }
    public int tocar(int cuerda_1, int cuerda_2, int n) {
    /* Tocar cuerda 1 y 2 a la vez y que retorne el n-esimo
    fib*/
    }
}
```

Figure: Guitarra.java

Interfaz

Las interfaces es una forma de declarar métodos abstractos y constantes. Son útiles para el diseño, permitiendo que las clases decidan cómo implementarlas. Haciendo que una interfaz pueda tener muchas implementaciones diferentes.

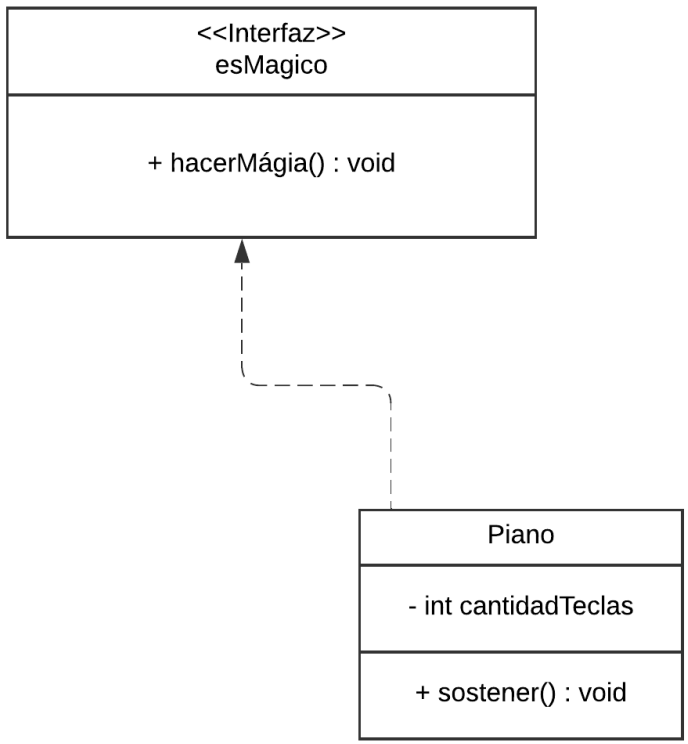


Figure: Piano Implementa la interfaz esMagico

Implementación Interfaz

Todas las interfaces puede tener **constantes**, que son públicas, estáticas y finales. Como también **métodos**, que son público y abstractos. Se utiliza la palabra clave `implements`, y en java permite implementar más de una interfaz.

```java
public interface
    EsMagico {
    public void
    hacerMagia();
}
```

(a) EsMagico.java

```java
public class Piano extends Instrumento
    implements EsMagico{
    public void hacerMagia()
    {
        /* Hacer Magia */
    }
}
```

(b) Piano.java

Otros

- Las clases abstractas no pueden ser instanciadas.
- Los getters y setters son métodos que permiten obtener y establecer un atributo privado de una clase.
- El principio de sustitución permite tener una referencia a una superclase y utilizarse para referenciar una subclase. Se aplica de igual manera para las interfaces.
- Las asociaciones permiten decir que una clase tiene relación con otra clase. Estos pueden ser 1-1, 1-N y N-M.
- Se puede usar `this` para referenciar la clase, y `super` para referenciar a la superclase. Estos pueden ser atributos, métodos y constructores.

Bibliografía

- https://www.oracle.com/java/technologies/downloads/
- https://es.wikipedia.org/wiki/Diagrama_de_clases
- https://www.w3schools.com/java/