POO Tema 5. Polimorfismo

Martín González Dios

25 de noviembre de 2024

1. Polimorfismo en herencia

Mecanismo mediante el cual un objeto **se puede comportar de múltiples formas** en cada momento, en función del contexto en el que aparece dicho objeto. Un objeto se puede comportar como:

- una **instancia de la clase a la que pertenece**, es decir, de la clase cuyo constructor se invoca a través de new.
- una **instancia de alguna de las clases que se encuentran en un nivel superior de la jerarquía** en relación a la clase a la cual pertenece realmente el objeto.

1.1. Ejemplo

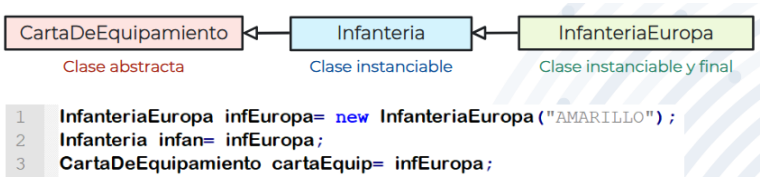

Figura 1: Ejemplo de polimorfismo

En la línea 1 se crea infEuropa, un objeto de la clase InfanteriaEuropa.
En la línea 2 se fuerza a que infEuropa se comporte como un objeto de la clase Infantería, que **es la clase base** de InfanteriaEuropa.
En la línea 3 se fuerza a que infEuropa se comporte como un objeto de la clase CartaDeEquipamiento, **aún cuando esa clase es de tipo abstracta**. (Ya que la restricción que impone el que sea abstracta es que no se pueden crear objetos de dicha clase, no se puede usar new)

2. Upcasting y Downcasting

Cuando se indica que un objeto se comporta como una clase de la jerarquía diferente a la que pertenece, se está forzando un cambio en el tipo de dato del objeto.

En **upcasting** un objeto de una clase derivada se comporta como una de las clases base de la jerarquía.
En **downcasting** un objeto de una clase base se comporta como una de las clases derivadas de la jerarquía.

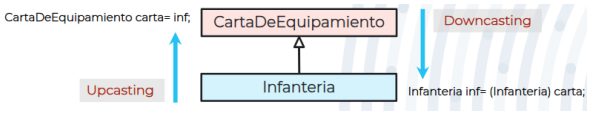

Figura 2: Upcasting y downcasting

Este cambio de tipo de dato **no implica una nueva reserva de memoria para el objeto**, sino que **el objeto seguirá siendo el mismo**, independientemente de los cambios de tipo de dato que tengan lugar a lo largo del programa.

El casting de objetos no solamente conlleva un cambio en el tipo de dato, sino que también **afecta** a la **accesibilidad de los atributos** y de los **métodos** que se pueden invocar desde el objeto. (Los únicos métodos que serán accesibles por el objeto son (1) los que están **definidos en la clase a la cual se ha realizado el cast**; y (2) **los que hereda dicha clase de las clases base de la jerarquía**)

2.1. Upcasting

Mecanismo más habitual, facilita el polimorfismo, ya que parece natural pensar que un objeto de la clase derivada se **comporta como** uno de sus clase base. (Cuando se realiza upcasting no hace falta indicar el cast de forma explícita).

Los métodos y atributos que son propios de la clase a la que pertenece el objeto **no son accesibles**, es decir, se pierde la visibilidad de esos métodos y atributos, ya que el objeto **deja de comportarse** como esa clase.

Si un método está sobrescrito cuando se realiza upcasting el objeto usará la implmentación del método correspondiente **a la clase que pertenece el objeto**, de modo que aunque se comporta como otra clase, no utiliza las implementaciones de los métodos de esa clase. (Al ser así puede haber poliformismo cuando se trabaja con clases abstractas)

El upcasting **nunca** producirá un error, ya que con la **herencia se garantiza** que las clases derivadas tienen los mismos métodos que la clase base. Cuando se ha realizado el upcasting **no hay ningún modo** de acceder a los métodos específicos de la clase a la cual pertenece dicho objeto.

Ejemplo:

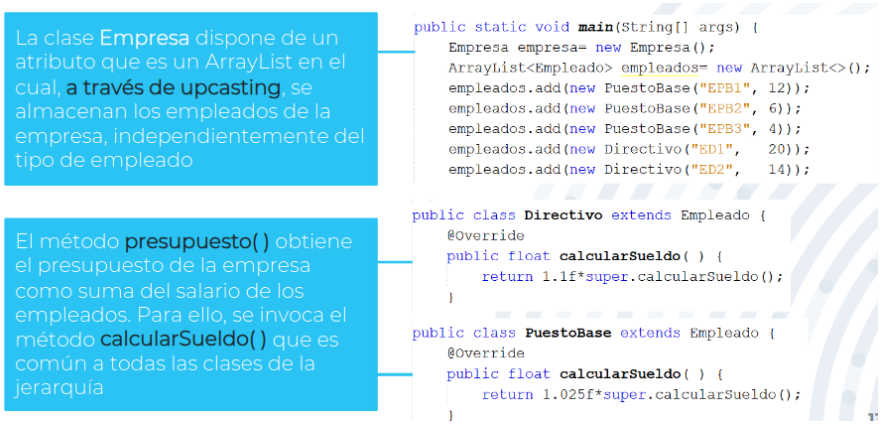

Figura 3: Ejemplo de upcasting

2.2. Downcasting

Downcasting **fuerza** el comportamiento de los objetos en las jerarquías de clases, ya que por lo general **no se puede asegurar** que un objeto de una clase base se comporte como una derivada. (downcastig no es una operación segura (**unsafe**), sino que es una operación en la cual se puede generar un error o una excepción en tiempo de ejecución)

Normalmente se hace downcasting cuando **se necesita deshacer** el resultado del upcasting, es decir, cuando se quiere recuperar la asignación del objeto a la clase a la cual pertenece. El compilador **no comprueba** en tiempo de diseño si el cast tendrá lugar de forma correcta, ni tan siquiera comprueba que el objeto se encuentre en la misma jerarquía que la clase derivada a la que se "convierte" (Es necesario indicar de forma explícita a qué clase se çonvertirá.[el] objeto de la clase base [Infanteria inf = (Infanteria) carta;])

El principal problema de la operación de downcasting es que **no se puede garantizar** que el objeto disponga de los mismos métodos y atributos que los de la clase derivada a la cual se realiza el casting. Se debe comprobar en tiempo de ejecución que el tipo de dato del objeto sea la propia clase derivada a la que se realiza el casting (operador instanceof)

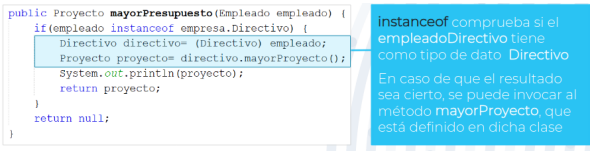

Figura 4: Ejemplo de uso de instanceof

El operador **instanceof** determina en tiempo de ejecución si el tipo de dato de un objeto es una clase dada, de manera que si devuelve true se podrán invocar los métodos de dicha clase.

Ejemplo:

Figura 5: Ejemplo de downcasting

3. Beneficios del polimorfismo

- Facilita la **simplificación del código del programa**, puesto que los objetos se pueden manejar como objetos de las clases base, de modo que la invocación de los métodos comunes a todas las clases se realiza desde una misma clase (la clase base).
- Facilita la **extensibilidad de los programas**, ya que la inclusión de nuevas clases en la jerarquía que implementan los métodos comunes de las clases base no supondrá modificaciones en el código del programa.

Buenas prácticas: En los conjuntos de datos, como ArrayList o HashMap, **se deben usar las clases derivadas** para almacenar los objetos de las clases de la jerarquía.
En la medida de lo posible se debe **favorecer al uso del polimorfismo**, para la simplificación de código y mejora de diseño.