Conceptos avanzados de POO en Java

Pedro O. Pérez M., PhD.

Desarrollo de videojuegos con Java
Tecnológico de Monterrey

pperezm@tec.mx

10-2023

Conceptos avanzados de POO en Java

- Clases abstractas
- Interfaces
- Clases internas

Examen Rápido 3

Clases abstractas

- Las clases abstractas funcionan como una clase que declara la existencia de métodos pero no su implementación. Eso es algo que haremos después en las diferentes subclases derivadas de la clase abstracta. Una clase abstracta puede contener métodos abstractos y no abstractos. En Java, la clase abstracta se declara con la palabra clave `abstract`.
- Un método abstracto es un método que no tiene implementación. Se declara utilizando la palabra clave `abstract` y termina con un punto y coma en lugar del cuerpo del método. Las subclases de una clase abstracta deben proporcionar una implementación concreta de todos los métodos abstractos definidos en la clase principal.
- No se puede crear una instancia de una clase abstracta.
- Se permiten constructores.

Clases abstractas

- Podemos tener una clase abstracta sin ningún método abstracto.
- Podemos definir métodos estáticos en una clase abstracta.
- Si una clase contiene al menos un método abstracto, entonces es obligatorio declarar una clase como abstracta.
- Si la clase derivada no puede implementar todos los métodos abstractos de la clase padre, entonces debemos declarar esa clase heredada como abstracta para que la clase que derive de éste proporcione implementación al método abstracto restante.

Listing 1: Clases abstractas

```
abstract class Subject {
        Subject() {
                System.out.println("Learning Subject");
        }

        abstract void syllabus();

        void Learn(){
                System.out.println("Preparing Right Now!");
        }
}

class IT extends Subject {
        void syllabus(){
                System.out.println("C , Java , C++");
        }
}

class Application {
        public static void main(String args[]) {
                Subject x = new IT();

                x.syllabus();
                x.Learn();
        }
}
```

Interfaces

- Al igual que una clase, una interfaz puede tener métodos y variables, pero los métodos declarados en la interfaz son abstractos por defecto (solo firma del método, nadie).
  - Las interfaces especifican qué debe hacer una clase y no cómo. Es el modelo de la clase.
  - Una interfaz se trata de capacidades como "Player" puede ser una interfaz y cualquier clase que implemente "Player" debe poder (o debe implementar) "move()". Entonces especifica un conjunto de métodos que la clase debe implementar.
  - Si una clase implementa una interfaz y no proporciona cuerpos de métodos para todas las funciones especificadas en la interfaz, entonces la clase debe declararse abstracta.
- Su declaración es similar a la de una clase; en la cabecera utiliza la palabra reservada `interface` en vez de `class`.

Clases	Interfaces
La palabra clave utilizada para crear una clase es çlass".	La palabra clave utilizada para crear una interfaz es ïnterface".
Se puede crear una instancia de una clase, es decir, se pueden crear objetos de una clase.	No se puede crear una instancia de una interfaz, es decir, no se pueden crear objetos.
Las clases no admiten herencia múltiple.	La interfaz admite herencia múltiple.
Se puede heredar de otra clase.	No puede heredar una clase.
Puede contener constructores.	No puede contener constructores.
No puede contener métodos abstractos.	Contiene sólo métodos abstractos.

Clases	Interfaces
Las variables y métodos de una clase se pueden declarar utilizando cualquier especificador de acceso (público, privado, predeterminado, protegido).	Todas las variables y métodos de una interfaz se declaran públicos.
Las variables de una clase pueden ser estáticas, finales o ninguna de las dos.	Todas las variables son estáticas y finales.

Listing 2: Ejemplo de interfaz

```
public interface GeometricFigure {
        public double area();
}

public class Square implements GeometricFigure {
        private double side;

        ...

        public double area() {
                return (side * side);
        }
}

public class Circle implements GeometricFigure {
        private double radious;

        ...

        public double area() {
                return (radious * radious) * Math.PI;
        }
}
```

- Java no permite que una clase derive de dos o más clases, es decir, no permite la herencia múltiple; sin embargo, una clase sí puede implementar más de una interfaz y tener el comportamiento común de varias de ellas.
- Las interfaces no son clases porque especifican un comportamiento mediante métodos para la clase que las implementa; por ello, una clase puede heredar de su clase base y a la vez implementar una interfaz.

Listing 3: Múltiples interaces

```
public class ClassA implementes Interface1, Interface2, ... Interfacen {
        }
```

Clases internas

- Una clase interna es la que se declara dentro de otra clase; se puede decir que es anidada.
- Las clases internas declaran atributos y métodos de igual forma que las externas o de nivel superior, con la peculiaridad de que sus métodos pueden acceder a los atributos de su clase externa.

Listing 4: Clases internas

```
public class Student {
        private int age;
        String name;
        Direction direction;

        public Student(String name, int age, String street, String city,
             String postalCode) {
                this.name = name;
                this.age = age;
                direction = new Direction(street, city, postalCode);
        }

        class Direction {
                String street, city, code;

                Direction(String street, String city, String code) {
                        this.street = street;
                        this.city = city;
                        this.code = code;
                }
        }

        ...
}
```

Un objeto de la clase Direction puede hacer referencia a los miembros de Student; los objetos de la clase interna disponen de una referencia implícita al objetos que los contiene.

- En java, las clases internas tienen tres ventajas. Son los siguientes:
  - La clase interna solo puede ser utilizada por la clase externa. Las clases internas representan un tipo especial de relación que permite el acceso a todos los miembros (miembros de datos y métodos) de la clase externa, incluida la clase privada.
  - Las clases internas se utilizan para desarrollar un código más legible y fácil de mantener porque agrupan lógicamente clases e interfaces en un solo lugar.
  - El acceso fácil, ya que el objeto interno, está implícitamente disponible dentro de un objeto externo. La optimización del código requiere menos código para escribir. Puede evitar tener una clase separada.

Examen Rápido 3 (Conceptos avanzados de POO en Java)