|  |  |
| --- | --- |
| Polimorfismo | Un concepto en la programación orientada por objetos que les permite a las clases tener varias formas y comportarse como sus superclases. |
| Sobrescritura | Implementar métodos en una subclase que tiene el mismo prototipo (los mismos parámetros, nombre de método, y tipo de retorno) que otro método en la superclase. |
| Final | Una palabra clave en Java usada para limitar las subclases a partir de la extensión de una clase, sobrescribiendo métodos o cambiando datos. |
| Final (en una clase) | Una propiedad de una case estática que hace que la clase no pueda ser extendida o que sus datos sean cambiados. |
| Sobrecarga | Implementar un método con el mismo nombre que otro método en la misma clase que tiene diferentes parámetros o un tipo de retorno diferente. |
| Enlace dinámico | El proceso por el cual Java es capaz de determinar qué método invocar cuando los métodos han sido sobrescritos. |
| Abstract | Una palabra clave en Java que permite que las clases sean extendidas, pero las clases no pueden ser instanciadas (construidas)y cuando se aplican los métodos, se indica que los métodos deberían ser implementados en todas las subclases de la clase. |

1.

class A

{

void callthis() {

System.out.println(“Inside Class A's Method!”);

}

}

class B extends A

{

void callthis() {

System.out.println(“Inside Class B's Method!”);

}

}

class C extends A

{

void callthis() {

System.out.println(“Inside Class C's Method!”);

}

}

class DynamicDispatch {

public static void main(String args[]) {

A a = new A();

B b = new B();

C c = new C();

A ref;

ref = b;

ref.callthis();

ref = c;

ref.callthis();

ref = a;

ref.callthis();

}

}

Respuesta

Inside Class B's Method!, Inside Class C's Method!, Inside Class A's Method!

2. ¿Cuál es la diferencia entre una clase abstract y una interfaz? ¿Cuándo es apropiado el uso de una clase abstract o una interfaz?

Una clase abstracta es una clase que no puede ser instanciada por sí sola, sino que sirve como una plantilla para otras clases que la extienden (heredan de ella). Las clases abstractas pueden tener métodos concretos (con implementación) y métodos abstractos (sin implementación), que deben ser implementados por las clases que las extienden. Las clases abstractas pueden contener atributos, constructores y métodos normales.

Una interfaz es una colección de métodos abstractos (sin implementación) que definen un conjunto de comportamientos o funcionalidades que deben ser implementados por las clases que la implementan (no heredan de ella, sino que la implementan). A diferencia de las clases abstractas, las interfaces no pueden tener atributos, constructores ni métodos normales concretos.

Es apropiado utilizar una clase abstracta cuando se quiere compartir código común y comportamiento entre las clases que la extienden, y cuando se necesita tener una jerarquía de clases con polimorfismo.

En cuanto a las interfaces, son apropiadas cuando se quiere definir un conjunto de comportamientos que deben ser implementados por diferentes clases de manera independiente, sin necesidad de compartir código común.

3.

|  |  |
| --- | --- |
| **Código:** | **¿Siempre compila, a veces compila, no compila?** |
| a = new B(); | Siempre compila |
| d = new C(); | No compila |
| b.methodA(); | A veces compila |
| e.methodA(); | Siempre compila |
| c = new C(); | No compila |
| (D)c.methodC(); | A veces compila |