Grado Superior
Programación

Clases abstractas e Interfaces

Curso 2023/24

Profesor
Jonathan Carrero

Java

Clases abstractas

Clases abstractas

Son clases “molde”, es decir clases de las cuales se puede heredar pero **nunca** se pueden crear objetos de dicha clase.

Aunque no se pueden crear objetos de una clase abstracta, si **es posible** declarar objetos de ese tipo.

Una clase es abstracta si:

- Contiene un método abstracto (un método que no contiene implementación, sólo su declaración). El método puede ser heredado.
- Si va precedida de la palabra `abstract`. En tal caso, podría ocurrir que ni siquiera tuviera métodos abstractos.

Clases abstractas

```
abstract public class A {
	protected int x;
	A(){
		this.x = 0;
	}
	A(int xx){
		this.x = xx;
	}
	public int getX(){
		return this.x;
	}
	abstract public void increment();
}
```

```
abstract public class A1 extends A {
	public A1(int xx){
		super(xx);
	}
	// No se implementa el método increment
	// La clase tiene que ser necesariamente abstracta
}
```

Clases abstractas

```
public class A2 extends A1 {
	private int y;
	public A2(int xx,int yy){
		super(xx);
		this.y = yy;
	}
	public void increment(){
		this.y = this.y + this.x;
	}
	public int getY(){
		return this.y;
	}
}
```

Clases abstractas

```
public class A2 extends A1 {
	private int y;
	public A2(int xx,int yy){
		super(xx);
		this.y = yy;
	}
	public void increment(){
		this.y = this.y + this.x;
	}
	public int getY(){
		return this.y;
	}
}
```

```
A a = new A(2);
A a = new A1(2);
A a = new A2(2,3);
a.increment();
System.out.println(a.getX());
System.out.println(a.getY());
```

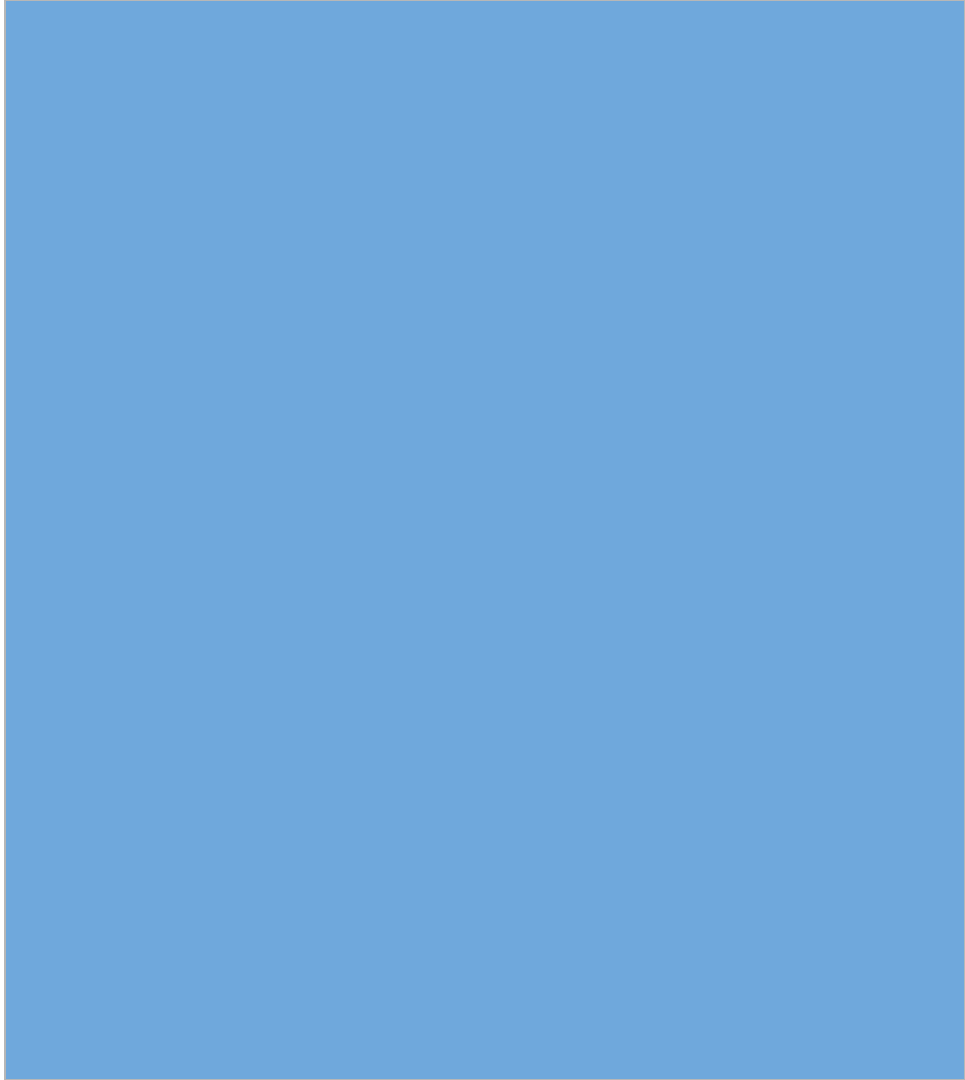
Clases abstractas

```
public class A2 extends A1 {
	private int y;
	public A2(int xx,int yy){
		super(xx);
		this.y = yy;
	}
	public void increment(){
		this.y = this.y + this.x;
	}
	public int getY(){
		return this.y;
	}
}
```

```
A a = new A(2);                 // ¿Correcto?
A a = new A1(2);                // ¿Correcto?
A a = new A2(2,3);              // ¿Correcto?
a.increment();                  // ¿Correcto?
System.out.println(a.getX()); // ¿Correcto?
System.out.println(a.getY()); // ¿Correcto?
```

Java

Interfaces



Interfaces y clases abstractas: diferencias

- Todos los métodos de una interfaz se declaran **implícitamente** abstractos y públicos.
- Una interfaz no puede implementar **ningún** método y una clase abstracta sí.
- Una interfaz **no tiene** atributos **salvo** que sean estáticos o constantes. Además, implícitamente siempre son públicos.
- Una clase puede implementar **varias** interfaces pero sólo puede extender a **una** clase.
- Una clase abstracta puede pertenecer a una jerarquía de clases mientras que una interfaz no. Por lo tanto, clases no relacionadas pueden implementar la misma interfaz.
- Una interfaz sólo puede **extender** a otras interfaces. Una clase puede **extender** otras clases e **implementar** distintas interfaces.

Sintaxis

La sintaxis de una interfaz es la siguiente:

Deben declararse en un archivo con el mismo nombre y extensión `.java`.

Sintaxis

La sintaxis de una interfaz es la siguiente:

```
public interface nombre_interfaz [extends I1, …, In] {
    [public abstract] tipo_retorno nombre_método (argumentos);
}
```

Deben declararse en un archivo con el mismo nombre y extensión `.java`.

Sintaxis

```
public interface FiguraGeometrica {
	final float PI = 3.1416;
	float area();
	void drawFiguraGeometrica();
}
```

Sintaxis

```
public interface FiguraGeometrica {
	final float PI = 3.1416;
	float area();
	void drawFiguraGeometrica();
}
```

```
public class Cuadrado implements FiguraGeometrica {
	private float lado;
	public Cuadrado(floatlado) {
		this.lado = lado;
	}
	public float area() {
		return lado * lado;
	}
	public void drawFiguraGeometrica(){
		System.out.println(“Longitud lado cuadrado: “ + this.lado);
	}
}
```

```
public class Circulo implements FiguraGeometrica {
	private float radio;
	public Circulo(float radio) {
		this.radio = radio;
	}
	public float area() {
		return PI * radio * radio;
	}
	public void drawFiguraGeometrica(){
		System.out.println(“Radio del círculo: “ + this.radio);
	}
}
```

Implementación

Una clase **puede** implementar una o varias interfaces.

Implementar una interfaz significa **implementar** métodos de la interfaz. Si la clase no implementa algún método de la interfaz, entonces la clase es necesariamente abstracta.

La clase, además, puede tener métodos propios. Si no implementa algún método, entonces es abstracta.

Implementación

Para poder utilizar una constante declarada en una interfaz, o un atributo estático, es necesario **anteponer** el nombre de la interfaz a la constante.

Tarea: crea una interfaz y una clase que la implemente. Declara constantes y/o atributos estáticos en ella y observa cómo funcionan.

La jerarquía entre interfaces permite la herencia simple y múltiple.

- Una clase puede implementar varias interfaces.
- Una interfaz puede “extender” a otras interfaces, pero no puede implementarlas.
- Una interfaz no puede tener métodos estáticos (un método estático no puede ser abstracto).

Implementación

Una clase puede, **simultáneamente**, heredar de otra clase e implementar **una o varias** interfaces.

Una interfaz es un tipo, por lo tanto, el tipo se puede utilizar en cualquier lugar donde se pueda utilizar el identificador de un tipo o de una clase.

Implementación

Una clase puede, **simultáneamente**, heredar de otra clase e implementar **una o varias** interfaces.

```
public class Clase extends SuperClase implements I1, … In {
    // Implementación de la clase
}
```

Una interfaz es un tipo, por lo tanto, el tipo se puede utilizar en cualquier lugar donde se pueda utilizar el identificador de un tipo o de una clase.

Grado Superior
Programación

Clases abstractas e Interfaces

Curso 2023/24

Profesor
Jonathan Carrero