Lenguajes, Tecnologías y Paradigmas de la programación (LTP)

Práctica 3: Polimorfismo e interfaces

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

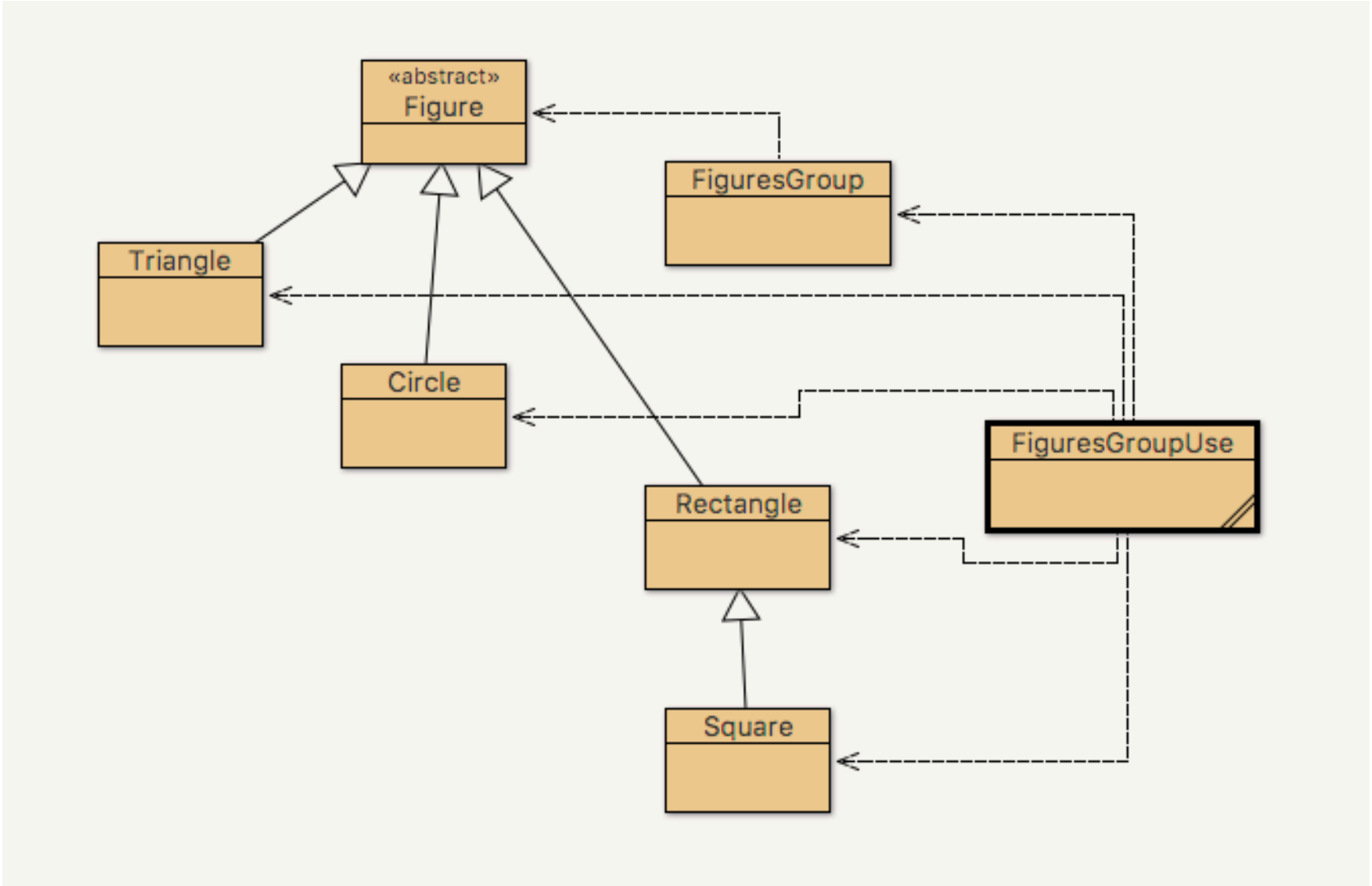
Sergio Pérez
serperu@dsic.upv.es

En episodios anteriores…

Herencia

Sobrecarga

Clases Abstractas



¿De que va la práctica 3?

Interfaces

¿Qué problema resuelven?

¿De que va la práctica 3?

Interfaces

¿Qué problema resuelven? **La herencia múltiple**

¿De que va la práctica 3?

Interfaces

¿Qué problema resuelven? **La herencia múltiple**

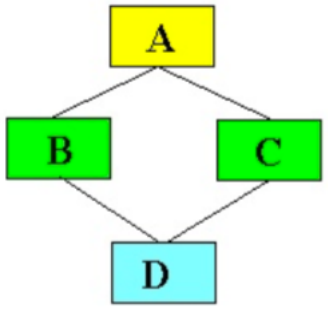
No se puede utilizar **extends** con más de una clase ¿Por qué?

¿De que va la práctica 3?

Interfaces

¿Qué problema resuelven? **La herencia múltiple**

No se puede utilizar **extends** con más de una clase ¿Por qué?




```
class B {
   public int area(){ … }
}
```

```
class C {
   public int area(){ … }
}
```

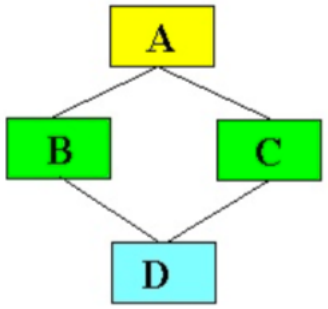
```
public static void main(String[] args){
    D d = new D();
    d.area();
}
```

¿De que va la práctica 3?

Interfaces

¿Qué problema resuelven? **La herencia múltiple**

No se puede utilizar **extends** con más de una clase ¿Por qué?




```
class B {
   public int area(){ … }
}
```

```
class C {
   public int area(){ … }
}
```

¿?

¿?

```
public static void main(String[] args){
   D d = new D();
   d.area();
}
```

Definición de interfaces

- Una interfaz es una clase completamente abstracta
- Sus métodos son públicos y abstractos por defecto
- Puede incluir la declaración de constantes
- No puede tener constructor

```
[modifVisibilidad] interface nomInterfaz [extends listaInterfaces]
   {
    [CONSTANTES]
    [ [modificador] tipoDevuelto nombreMetodo1([parámetros]);
                  ...
      [modificador] tipoDevuelto nombreMetodoN([parámetros]); ]
   }
```

Ejemplo

```
public interface MyInterface extends OtherInterfaces
   {
    final int myConstant = 10;
    int myMethod(int a, int b);
   }
```

Uso de interfaces

```
[listaModificadores] class NomClase [extends NomClase]
                          [implements listaInterfaces]
   {
        ...
   }
```

Ejemplo

```
public class MyClass extends ExtendedClass implements MyInterface {
        ...
}
```

Los métodos definidos en **`MyInterface`** se deben implementar en **`MyClass`**

Uso de interfaces predefinidas

- Interfaz **Comparable**:

  Definida en Java solo tiene un método abstracto (**compareTo**)

```
public interface Comparable<T>{
   int compareTo(T o);
}
```

Uso de interfaces predefinidas

- Interfaz **`Comparable`**:

  Definida en Java solo tiene un método abstracto (**compareTo**)

```
public interface Comparable<T>{
   int compareTo(T o);
}

public class Figure implements Comparable<Figure>{
   public int compareTo(Figure o) { … }
}
```

Extensión de interfaces

```
public interface Interface1{
     int method1(Object o);
}
```

Extensión de interfaces

```
public interface Interface1{
     int method1(Object o);
}

public interface Interface2 extends Interface1{



}
```

Extensión de interfaces

```
public interface Interface1{
     int method1(Object o);
}

public interface Interface2 extends Interface1{

     int method2(Object o);
}
```

Extensión de interfaces

```
public interface Interface1{
     int method1(Object o);
}

public interface Interface2 extends Interface1{
     int method1(Object o);
     int method2(Object o);
}
```

Extensión de interfaces

```
public interface Interface2 extends Interface1{
     int method2(Object o);
}

public class Triangle implements Interface2{



}
```

Extensión de interfaces

```
public interface Interface2 extends Interface1{
     int method2(Object o);
}

public class Triangle implements Interface2{
     int method2(Object o) {…}

}
```

Extensión de interfaces

```
public interface Interface2 extends Interface1{
     int method2(Object o);
}

public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
```

Tipos de un mismo objeto

```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
```

Tipos de un mismo objeto

```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
public static void main(String args[]){
     Triangle t = new Triangle(1, 2, 3, 4);
}
```

Tipos de un mismo objeto

```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
public static void main(String args[]){
     Triangle t = new Triangle(1, 2, 3, 4);
}
```

¿De qué tipo es el objeto **t**?

Tipos de un mismo objeto

```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
public static void main(String args[]){
     Triangle t = new Triangle(1, 2, 3, 4);
}
```

¿De qué tipo es el objeto **t**?

t

Tipos de un mismo objeto

```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
public static void main(String args[]){
     Triangle t = new Triangle(1, 2, 3, 4);
}
```

¿De qué tipo es el objeto **t**?

Tipos de un mismo objeto

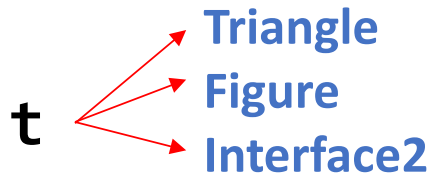
```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
public static void main(String args[]){
     Triangle t = new Triangle(1, 2, 3, 4);
}
```

¿De qué tipo es el objeto **t**?

Tipos de un mismo objeto

```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
public static void main(String args[]){
     Triangle t = new Triangle(1, 2, 3, 4);
}
```

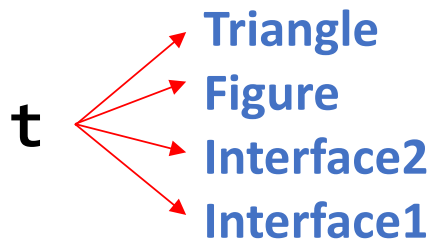
¿De qué tipo es el objeto **t**?



Tipos de un mismo objeto

```
public class Triangle implements Interface2{
     int method2(Object o) {…}
     int method1(Object o) {…}
}
public static void main(String args[]){
     Triangle t = new Triangle(1, 2, 3, 4);
}
```

¿De qué tipo es el objeto **t**?



Ejercicio 2

Objetivo:

Poder comparar figuras en función de su área

¿Como?:

La clase `Figure` debe implementar la interfaz `Comparable<T>`

¿qué método(s) hay que implementar en `Figure`?

Ejercicio 3

Objetivo:

Implementar un método en `FiguresGroup` que devuelva una lista ordenada de menor a mayor de sus figuras usando el método `compareTo`

¿Como?:

- Importar la librería `java.util.*`
- Implementar en la clase el método `orderedList()`

NOTA: `FiguresGroup` no debe implementar la interfaz `List`, sino utilizar objetos que ya la implementan (`LinkedList` o `ArrayList`)

Ejercicio 4

Objetivo:

Definir una nueva interfaz `ComparableRange` que herede de la interfaz `Comparable`

¿Como?:

- Definiendo la nueva interfaz (una clase)
- Incluyendo la cabecera del método `compareToRange` que exigiremos a las clases que la implementen

Ejercicio 5

Objetivo:

Los rectángulos y cuadrados deben poder compararse utilizando el método `compareToRange`

¿Como?:

- La clase `Rectangle` implementará la interfaz `ComparableRange`
- Implementaremos el método `compareToRange`:
- Utilizaremos el método `compareTo` para comparar en los casos que no cumplan la condición impuesta en `compareToRange` (diferencia de areas <= 10% suma de areas)

Ejercicio 6

Objetivo:

Definir una nueva interfaz `Printable` que defina el método:

```
void print (char c)
```

¿Como?:

- Definiendo la nueva interfaz (una clase)
- Incluyendo la cabecera del método `print` que exigiremos a las clases que la implementen

Ejercicio 7

Objetivo:

Añadir a algunas figuras la impresión por pantalla implementando la interfaz `Printable`

¿Como?:

- Implementaremos el método `print` en las clases que puedan imprimir la figura (`Circle` y `Rectangle`)

Ejercicio 8

Objetivo:

Queremos imprimir los objetos que se pueda en un objeto de tipo `FiguresGroup`

¿Como?:

- Implementar el método `print` partiendo del código dado
- Corregir el método para imprimir por pantalla solo aquellas figuras que sean "`Printables`" (puedes usar `instanceof Printable`)