Modelado y Diseño de Clases

2.4 Clases Abstractas, Concretas e Interfaces

Programación

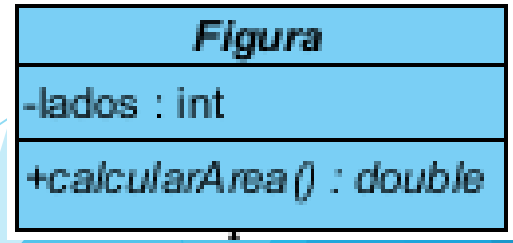
Septiembre 2015

Objetivo

- Abordar y aplicar los conceptos de las clases abstractas, concretas e interfaces en la POO

Clases Abstractas (1/2)

- Este tipo de clases permiten la creación de métodos genéricos que definen un comportamiento en común
- A nivel de codificación las clases abstractas permiten definir únicamente la firma de los métodos, sin especificar un cuerpo de declaración, es decir, se define el qué pero no el cómo se implementa
- La palabra para identificar las clases abstractas es **abstract**
- Las clases abstractas **NO** pueden ser instanciadas, debido a que su nivel de abstracción es alto, por lo tanto, su finalidad es que la clases abstractas sean heredadas o extendidas por clases concretas
- En un diagrama UML la clases abstractas y métodos abstractos se definen con su nombre en ***cursivas***
- Las clases que heredan de la clase abstracta deben implementar **TODOS** los métodos abstractos



Clases Abstractas (2/2)

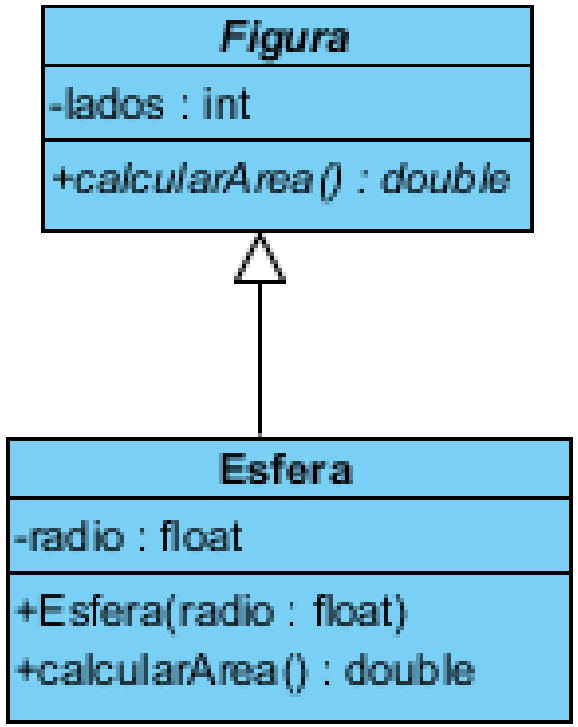
- Las clases abstractas pueden heredar de otras clases
- Si una clase contiene uno o más métodos abstractos está tendría que ser abstracta
- Las clases abstractas pueden tener métodos implementados o no (abstractos)
- Clases Tradicionales:
  - ¿Qué? **Y** ¿Cómo?: Incluyen firma e implementación
- Clases Abstractas:
  - ¿Qué? **O** ¿Cómo?: Pueden tener únicamente firmas o métodos concretos implementados

Ejemplo (1/2)

- **Implementación en Java**

```
public abstract class Figura{
    private int lados;
    //¿Qué? Y no ¿Cómo?
    abstract public void calcularArea();

    //¿Qué y Cómo?
    pubblic void mostrar(){
        System.out.println();
    }
}
```



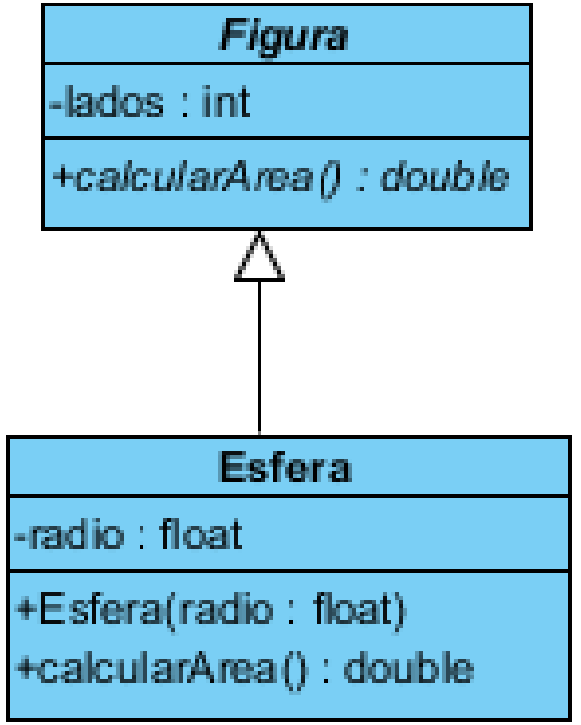
Ejemplo (2/2)

- **Implementación en Java: en la clase que implementa es necesario que la firma del método abstracto sea igual, de lo contrario sería un nuevo método**

```
public class Esfera extends Figura{
    private float radio;

    public Esfera(float radio){
        this.radio = radio;
    }

    public void calcularArea(){
        double area = (4)*(3.1416)*radio*radio;
        return area;
    }
}
```



Clases Concretas

- En lenguajes POO es posible definir clases que ya no puedan ser heredades
- Esto se conoce como clases selladas o finales (sealed class)
- De igual forma, también se pueden definir métodos que ya no puedan ser sobre-escritos en las clases derivadas
- La palabra reservada utilizada en Java para ello es ***final***
  - Esta palabra permite que:
    - Una **clase** ya no pueda ser derivada o heredada por otra
    - Un **método** no pueda ser sobre-escrito
    - Una **variable** solo pueda ser inicializada una sola vez (constante)

Ejemplo (1/2)

```
//Utilizando final en clases y variables
public final class A{
    private static final int x = 10; //Constante,
                                     // una vez asignado un valor ya
                                    //este no podrá cambiar
    public class A{
    }
}

public class B extends A{ //NO está permitido porque A es definida con final
                          //Error de compilación
}
```

Ejemplo (1/2)

```
//Utilizando final en métodos
public class A{
    public class A{
    }
    public final void mostrar(){
        System.out.println("Mostrando");
    }
}

public class B extends A{
    public void mostrar(){        //Error de compilación, en la superclase fue
        System.out.println("Mostrando"); // definido como final, por lo tanto existiría
    }                                    // error de compilación, NO ES POSIBLE SOBRE-ESCRIBIR
}
```

Interfaces

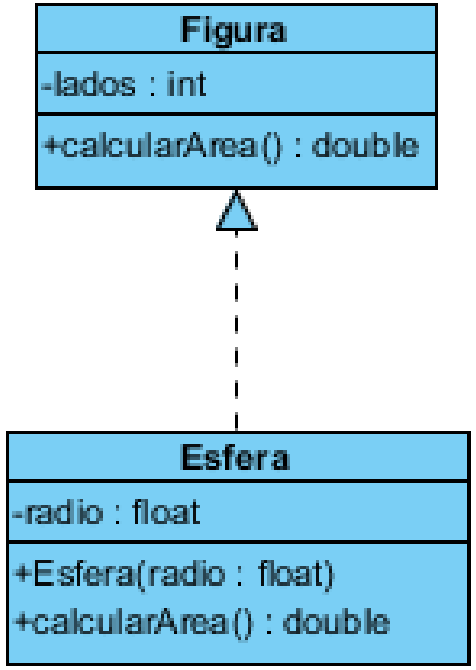
- Al igual que las clases abstractas las interfaces no pueden ser instanciadas
- Los métodos de un interfaz deben ser implementados por la clase que implemente
- También es posible definir propiedades pero serían constantes y no variable de instancias
- Una interfaz solo puede tener la firma de los métodos (pero no implementaciones)
- Clases Tradicionales:
  - ¿Qué? **Y** ¿Cómo?: Incluyen firma e implementación
- Clases Abstractas:
  - ¿Qué? **O** ¿Cómo?: Pueden tener únicamente firmas o métodos concretos implementados
- Interfaz
  - ¿Qué? : únicamente firmas o constantes
- A diferencia de las clases, donde solo existe herencia simple y una clase solo puede heredar al mismo tiempo de una clase, con las interfaces una clase puede implementar múltiples interfaces al mismo tiempo

Ejemplo

- **Implementación en Java**
- En Java, cuando una clase implementa una interfaz se utiliza la palabra reservada **implements**

```
public interface Figura{
     private static int lados = 10;
     //¿Qué?
     public void calcularArea();
}

public class Esfera implements Figura{
     public void calcularArea(){
          double area = (4)*(3.1416)*radio*radio;
          return area;
     }
}
```



Actividad 6 – Rediseñando tu Diagrama de Clases

- Dado los conceptos abordados modifica tu diagrama para establecer clases abstractas, concretas, o interfaces
- Una vez modificado tu diagrama de clases implementar tus nuevos cambios
- Entregable: diagrama y código fuente