UNIDAD 8: HERENCIA

CLASE 3: CLASES ABSTRACTAS Y EJERCICIOS

JERARQUÍA DE CLASES

PERSONA

Más arriba,
más general

ALUMNO

Más abajo,
más específico

ALUMNO_de_FP

JERARQUÍA DE CLASES

PERSONA

mostrarDatosAcademicos ( ) { }

ALUMNO

Override mostrarDatosAcademicos ( ){...}

MÉTODOS ABSTRACTOS

Un método que se define en una clase pero cuya implementación se delega a clases hijas.

abstract mostrarDatosAcademicos ( );

CLASES ABSTRACTAS

Si una clase tiene al menos 1 método abstracto, será abstracta

```
abstract public class Persona {
    abstract public String mostrarDatosAcademicos();
}
```

CLASES ABSTRACTAS

- Las clases abstractas no son instanciables
- Existen para que otras hereden de ellas.
- Si una clase hereda de una clase abstracta y no implementa todos sus métodos también será abstracta.

CLASES ABSTRACTAS

- Las clases abstractas podrán tener atributos y métodos implementados que sus hijas podrán heredar y utilizar o sobreescribir.

```
abstract public class Persona {
    abstract public String mostrarDatosAcademicos();
    public void queClaseEs() {
        System.out.println("Es la clase Persona");
    }
}
```

CLASES ABSTRACTAS

```java
abstract public class A {
    void method1() {
        System.out.println("Método implementado en A");
    }
    abstract void method2();
}
```

```java
public class B extends A{
    @Override void method2() {
        System.out.println("Método implementado en B");
    }
}
```

```java
public class C extends A{
    @Override void method2() {
        System.out.println("Método implementado en C");
    }
}
```

CLASES ABSTRACTAS

```
public static void main(String[] args) {
    B b =  new B();
    C c =  new C();
    b.method1();
    b.method2();
    c.method1();
    c.method2();
}
```

```
Método implementado en A
Método implementado en B
Método implementado en A
Método implementado en C
```

CLASES ABSTRACTAS

```
A a = b;
a.method2();
```

```
Método implementado en B
```

```
a = c;
a.method2();
```

```
Método implementado en C
```

CLASES ABSTRACTAS

```
A a = b;
a.method2();
```

```
a = c;
a.method2();
```

SELECCIÓN DINÁMICA DE MÉTODOS

EJERCICIOS

Ejercicio 1:

Crea la clase **Poligono** que tendrá:

- Los atributos **base** y **altura**.
- Los métodos:
    - **abstract double area**( ): Será sobrescrito por las clases hijas.

Después crea las clases **Triángulo** y **Rectángulo** que sobrescribirán el método area para calcularla con sus propias fórmulas y mostrarán por pantalla "El area del polígono -ClaseQueSea- es -Area-"

También crea la clase **PoligonoRegular** que hereda de **Poligono** y tendrá**:**

- Los atributos **numeroLados** y **apotema**. Utilizaremos altura como longitud del lado.
- También tendrá el método area que sobrescribirá de la misma manera.

EJERCICIOS

Ejercicio 2:

Crea el enumerado **Nota** que pueda tener valor de una de las notas musicales (DO-SI).

Crea la clase abstracta **Instrumento** que tendrá:

- Una tabla con las notas musicales de una obra.
- Los métodos:
  - **add**( ): Añade nuevas notas musicales.
  - **abstract interpretar**( ): Cada subclase heredará y rescribirá este método a su manera.

Crea la clase **Piano** que es subclase de **Instrumento** que tendrá:

- La cantidad de octavas del teclado.
- El método interpretar( ) mostrará la tecla seguida de la octava en la que se toca, para ello se eligirá un número random desde 1 hasta la cantidad de octavas. “do2, fa5, mi1...”

EJERCICIOS

Crea la clase **Campana** que es subclase de **Instrumento** que tendrá:

- La reverberación de la campana que será un entero.
- El método **interpretar**( ) mostrará la nota con la cantidad de vocales que indique reverberación: R=3 -> “dooo, faaa, miii...”

Crea la clase **Guitarra** que es subclase de **Instrumento** que tendrá:

- La cantidad de cuerdas de la guitarra.
- El método **interpretar**( ) mostrará la nota como resultado de la combinación de dos cuerdas que se eligirán aleatoriamente entre 1 y el máximo de cuerdas “1+3=do, 4+5=fa, 1+6=mi...”