UT07. POO. UTILIZACIÓN AVANZADA DE CLASES

Programación de 1DAW

C.I.F.P. Carlos III - Cartagena

Polimorfismo

- *"Propiedad de ciertos cuerpos de **cambiar de forma** sin variar su naturaleza."* RAE
- **Herencia** ➔ se establece entre las clases implicadas una relación del tipo "**es-un**".
  - Un objeto de la subclase es un objeto de la superclase y puede ser tratado como tal.
  - Aunque, un objeto de la superclase no puede ser tratado como un objeto de la subclase.
- **Polimorfismo** => posibilidad de que toda referencia a un objeto de una superclase pueda tomar la forma de una referencia a un objeto de una subclase heredada de la anterior.

```
public class Persona{
   private String nombre;

   public void setNombre (String n){
      this.nombre = n;
   }

   public String getNombre(){
      return nombre;
   }
}

public class Empleado extends Persona{
   protected int sueldoBase;

   public void setSueldoBase (int s){
      this.sueldoBase=s;
   }

   public int getSueldo(){
      return this.sueldoBase;
   }
}
```

```
public class Encargado extends Empleado{

   public int getSueldo(){
      Double d = new Double(sueldoBase * 1.1);
      return d.intValue();
   }
}
```

Polimorfismo

```
public class Test{
   public static void main(String[] args){
      Persona p1;
      p1 = new Empleado();
      p1.setNombre("Isaac Sánchez");
      p1.setSueldoBase(100); //dará error al compilar. Solo puede hacer
referencia a métodos de la clase Persona
      Empleado e1;
      e1 = new Encargado();
      e1.setSueldoBase(500);
      e1.setPuesto("Jefe de almacén"); //dará error al compilar. Solo
puede hacer referencia a métodos de la clase Empleado
      System.out.println (e1.getSueldo);
   }
}
```

¿Qué dará getSueldo()?

Creación de una referencia **Persona** que apunta a un objeto de la clase **Empleado**

Creación de una referencia **Empleado** que apunta a un objeto de la clase **Encargado**

Polimorfismo

- **Polimorfismo**
  - Una variable tiene un tipo definido en **tiempo de compilación.**
  - En t**iempo de ejecución** puede estar referenciando a objetos de tipos distintos, con la única condición de que estos tipos sean subclases directas o indirectas del tipo definido en tiempo de compilación para la variable.

Polimorfismo

- **¿Qué dará getSueldo? ➔ Solución: 550**
  - **getSueldo()** está sobrescrito y como apunta a **Encargado,** se ejecuta el método de Encargado. Esta característica se denomina **Enlace o ligadura dinámica.**
  - **Ligadura dinámica:** capacidad de los lenguajes orientados a objetos de determinar, en tiempo de ejecución, qué versión o implementación de un método tiene que ser ejecutada cuando dicho método ha sido redefinido en las subclases y, además, se invoca sobre una variable polimórfica.
- Invocar un método exclusivo de una subclase sobre una variable del tipo de la superclase produce un **error en tiempo de compilación**.
- Si una variable ha sido declarada con el tipo de la superclase, sólo se podrán **invocar métodos propios de la superclase** sobre el objeto al que hace referencia, aunque en tiempo de ejecución dicha variable esté haciendo referencia a un objeto de una de sus subclases.

Polimorfismo

- **Solución:** podemos usar dos recursos
  - **instanceof**
    - Un operador lógico (devuelve verdadero o falso), llamado **instanceof**, que me permite saber si un objeto dado es de una clase determinada.

```
variable instanceof NombreClase
```

  - **Casting explícito:** conversión entre tipos (clases)
    - Sólo si la variable origen está referenciando en tiempo de ejecución a un objeto de la clase de la variable destino, pues, de otro modo, se produciría un error definido por la excepción **ClassCastException**.
- Ya se puede referenciar a los métodos de la subclase

```java
public class Persona {
    private String nombre;

    public Persona(){
        this.nombre="Desconocido";
    }

    public void setNombre(String nombre){
        this.nombre=nombre;
    }

    public String getNombre(){
        return this.nombre;
    }
}
```

```java
public class Empleado extends Persona{
    protected int sueldo; //para que pu

    public Empleado(){
        this.sueldo=0;
    }

    public void setSueldo(int sueldo){
        this.sueldo=sueldo;
    }

    public int getSueldo(){
        return this.sueldo;
    }
}
```

```java
public class Encargado extends Empleado{
    private int categoria; //categoría 1, 2 ó 3

    public Encargado(){
        this.categoria=3;
    }

    public void setCategoria(int categoria){
        this.categoria=categoria;
    }

    public int getCategoria(){
        return this.categoria;
    }

    @Override
    public int getSueldo(){
        Double sueldoEncargado = new Double (this.sueldo * 1.1);
        return sueldoEncargado.intValue();
    }
}
```

```java
public class Main {

    /**...3 lines */
    public static void main(String[] args) {
        Persona unaPersona, otraPersona;
        unaPersona = new Empleado();
        otraPersona= new Encargado();
        Empleado unEmpleado = new Encargado();

        unaPersona.setNombre("Alfonso XII");
        otraPersona.setNombre("María de las Mercedes");
        if (unaPersona instanceof Empleado){
            Empleado otroEmpleado = (Empleado)unaPersona;
            System.out.println("Sueldo empleado: " + otroEmpleado.getSueldo());
        }
        //Enlace Dinámico
        unEmpleado.setSueldo(2000);
        System.out.println("Sueldo encargado: " + unEmpleado.getSueldo());
    }
}
```

Creación de referencias **Persona** que apuntan a un objeto de la clase **Empleado** y a otro de la clase **Encargado**

Creación de una referencia **Empleado** que apunta a un objeto de la clase **Encargado**

setNombre es un método de **Persona**

Para acceder a los métodos de Empleado o Encargado, como es una referencia de tipo Persona:

- Con **instanceof** determino el la clase del elemento referenciado
- Aplico un **casting explícito** para poder acceder a los elementos de Encargado o Empleado

A través del enlace dinámico se invoca getSueldo() de Encargado, versión **redefinida** e invocada sobre una **variable polimórfica**