UNIVERSIDAD DE LIMA

CLASES Y MÉTODOS

UNIDAD 2: INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

SEMANA 5

UNIVERSIDAD DE LIMA

El error de software que convirtió un lanzamiento espacial en carísimos fuegos artificiales

Temario

- Referencia al objeto actual.
- Constructor sobrecargado y predeterminado.
- Composición entre clases.
- Variable de instancia final.

Referencia al objeto actual

- La referencia al objeto actual conocido en Java como "this" es una referencia que un objeto utiliza para acceder a sus propias variables y métodos.
- Puede ser usado de manera implícita o explícita en el código de una clase.
- En el ejemplo proporcionado, se muestra cómo "this" se utiliza para referirse a variables de instancia y métodos en una clase llamada "SimpleTime".

```
class SimpleTime
{
   private int hour; // 0-23
   private int minute; // 0-59
   private int second; // 0-59

   // if the constructor uses parameter names identical to
   // instance variable names the "this" reference is
   // required to distinguish between the names
   public SimpleTime(int hour, int minute, int second)
   {
      this.hour = hour; // set "this" object's hour
      this.minute = minute; // set "this" object's minute
      this.second = second; // set "this" object's second
   }

   // use explicit and implicit "this" to call toUniversalString
   public String buildString()
   {
      return String.format("%24s: %s%n%24s: %s",
         "this.toUniversalString()", this.toUniversalString(),
         "toUniversalString()", toUniversalString());
   }

   // convert to String in universal-time format (HH:MM:SS)
   public String toUniversalString()
   {
      // "this" is not required here to access instance variables,
      // because method does not have local variables with same
      // names as instance variables
      return String.format("%02d:%02d:%02d",
         this.hour, this.minute, this.second);
   }
} // end class SimpleTime
```

Constructor sobrecargado y predeterminado

- Los constructores **sobrecargados** son métodos en una clase que tienen el mismo nombre pero diferentes parámetros. Permiten inicializar objetos de la clase de diferentes maneras.
- Los constructores **predeterminados** son aquellos que se proporcionan automáticamente si no se define ningún constructor en la clase. Estos constructores predeterminados inicializan las variables de instancia a sus valores por defecto.

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

    // Constructor por defecto (sin argumentos)
    public Persona() {
        this.nombre = "Sin nombre";
        this.edad = 0;
    }

    // Constructor sobrecargado (con argumentos)
    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }
```

Composición entre clases

- La **composición** en programación orientada a objetos se refiere a la capacidad de una clase para tener referencias a objetos de otras clases como parte de sus atributos.
- Esto se conoce como una relación "tiene un" (has-a relationship). Por ejemplo, un objeto de la clase RelojDespertador necesita conocer la hora actual y la hora a la que debe sonar la alarma, por lo que sería razonable incluir dos referencias a objetos de la clase Tiempo en un objeto de RelojDespertador. Esto permite combinar objetos más simples para crear objetos más complejos, lo que fomenta la reutilización de código, la flexibilidad y la encapsulación en la programación orientada a objetos.
- Un ejemplo práctico de composición sería una clase Empleado que contiene objetos de la clase Fecha para representar la fecha de nacimiento y la fecha de contratación de un empleado. De esta manera, se puede modelar información más detallada y estructurada en el programa al combinar diferentes objetos en una clase principal.

Variable de instancia final

- El principio de "menor privilegio" en la ingeniería de software implica que las variables de instancia deben tener solo los privilegios necesarios para su tarea y no más.
  - En Java, se utiliza la palabra clave "final" para declarar variables de instancia como constantes, lo que evita modificaciones accidentales o maliciosas y garantiza que se inicializan en la declaración o en cada constructor de la clase para que cada objeto tenga su propio valor constante.

```
private final int INCREMENT;
```

UNIVERSIDAD DE LIMA

Referencia

- Deitel, H. M. (2016). Java: como programar.
- Arturo Rozas Huacho, Algoritmos y Estructuras de Datos, 2002, Grupo Liebre, Cusco, Perú.

UNIVERSIDAD DE LIMA