Tecnicatura en Desarrollo de Software
Paradigmas de Programación
Trabajo Práctico Nro 2

Clases y Objetos

Profesor:

Santiago DELLA TORRE

Integrantes:

Agustin Teresa Garcia

Leonardo Victor Paz

Rocio Guadalupe Manquillan

Ludmila Araceli Pereyra

Diego Maximiliano Orias

Índice

1) Defina: Objeto, Clase, Mensaje, Atributo, Método e Instancia.

2) ¿Qué es un constructor? ¿Cómo ocurre la inicialización de atributos en la creación de un objeto? ¿Qué es un constructor por defecto?
Nota: investigar acerca del bloque de inicialización

3) ¿Qué es la sobrecarga de métodos? ¿Por qué es necesaria? ¿Cómo decide Java con qué método responderá a cada mensaje?

4) Cree una clase en java que represente a la raza de los increíbles Hulk, cada Hulk tiene como atributos su peso y el color de su piel. Un Hulk es una persona normal pero cuando recibe el estímulo provocar cambian algunos de sus atributos: su color de piel que cambia del color original al verde y su peso se quintuplica; además cambia su comportamiento, en su estado normal al recibir el mensaje saludar dice “hola, soy fulano” (donde fulano es su nombre), pero transformado responde con un grito “aghhhhh”. Cuando está transformado sólo vuelve a su estado normal al calmarse a través del mensaje calmar. Asegure que los atributos de Hulk estén debidamente encapsulados y sólo puedan ser modificados a través de los mensajes definidos en este enunciado.

5) Escriba una clase que represente un barco transportador de granos con su nombre, calado, manga, eslora, capacidad de carga, carga actual y velocidad crucero. Un barco puede encontrarse en uno de dos estados: atracado en el puerto X o navegando hacia el puerto X (donde X es algún puerto). Permite que un barco pueda ser cargado o descargado.

Punto uno:

Clase: Una clase es "un tipo definido por el usuario; son los bloques de construcción fundamentales de los programas orientados a objetos" (Joyanes & Zahonero, 2011, p. 192). Es la unidad básica (o estructura) que encapsula toda la información de un Objeto.Las clases en Java (Java Class) son plantillas para la creación de objetos.

Objeto: Un objeto es una instancia específica de una clase que encapsula datos y comportamientos relacionados.

Mensaje: Pasar mensajes en Java es como enviar un objeto, es decir, un mensaje de un hilo a otro . Se utiliza cuando los subprocesos no tienen memoria compartida y no pueden compartir monitores, semáforos o cualquier otra variable compartida para comunicarse.

Atributo: Los atributos en Java, también conocidos como campos o variables de instancia, son variables que pertenecen a una clase y representan las características o propiedades que describen el estado de un objeto. Estos atributos definen las características únicas de cada objeto creado.

Método: Los métodos son bloques de código que se utilizan para realizar tareas específicas. Cada método tiene un nombre que lo identifica y puede recibir una serie de parámetros, los cuales son variables que proporcionan información necesaria para que el método realice su tarea.

Instancia: Las instancias son la materialización de una clase, es decir, son objetos concretos que tienen su propio estado (los valores de sus atributos) y su propio comportamiento (los métodos que pueden ejecutar).

Punto dos:

Un **constructor** es un método especial en la programación orientada a objetos que se usa para preparar un objeto en el momento en que se crea. Básicamente, cuando instanciamos un objeto, el constructor se encarga de darle valores iniciales a sus atributos y de hacer cualquier otra configuración que sea necesaria para que el objeto esté listo para usarse.

La Inicialización de atributos puede hacerse :

1. **Asignando Valores Iniciales**: El constructor puede recibir valores (a través de parámetros) y asignarlos a los atributos del objeto. También puede tener valores por defecto para estos atributos.
2. **Configuración Inicial**: Además de asignar valores a los atributos, el constructor puede hacer otras tareas como establecer conexiones a bases de datos, preparar estructuras internas, etc.

Constructor por Defecto

Un **constructor por defecto** es el constructor que no recibe parámetros. Si no que definís un constructor en tu clase, el compilador te proporciona uno automáticamente. Este constructor por defecto inicializa los atributos del objeto con valores predeterminados, como 0 para los enteros, `false` para los booleanos y `null` para los objetos.

Punto tres:

```java
public class Hulk {                                         // Inicializo la clase
    private int peso;                                       // Inicializo sus atributos
    private String colorPiel;                               // Inicializo sus atributos
    private String transformacion;                          // Inicializo sus atributos
    private String nombre;                                  // Inicializo sus atributos
public Hulk(String nombre, int peso, String colorPiel) {    // Creo un constructor
        this.nombre = nombre;                               // Le asigno a sus atributos los parámetros
        this.peso = peso;                                   // Le asigno a sus atributos los parámetros
        this.colorPiel = colorPiel;                         // Le asigno a sus atributos los parámetros
        this.transformacion = "Humano";                     // Inicialmente es "Humano"
}
```

```java
public void transformar() {                                 // Creo un método para transformar a Hulk
        this.peso *= 5;
        this.colorPiel = "Verde";
        this.transformacion = "Bestia";
    }
}
public void calmar(){                                       // Creo un método para calmar a Hulk
    this.peso /= 5;
    this.colorPiel = "Normal";
    this.transformacion = "Humano";
}
public void saludar(){                                      // Creo un método para saludar // Creo un constructor
    if(this.transformacion == "Humano") {
        System.out.println("Hola, mi nombre es " + nombre);
    } else {
        System.out.println("Aaaaahg");
    }
}
public int getPeso(){
    return peso;
}
public String getColorPiel(){
    return colorPiel;
}
```

Explicación del código realizado:

La clase Hulk define un objeto con atributos y métodos para modelar a un personaje que puede transformarse entre dos estados: "Humano" y "Bestia".

Los atributos que utilizamos son :

- **peso**: Un entero que representa el peso del Hulk.
- **colorPiel**: Un String que indica el color de la piel del Hulk.
- **transformación**: Un String que representa el estado actual del Hulk (por ejemplo, "Humano" o "Bestia").
- **nombre**: Un String que almacena el nombre del Hulk.

Los métodos métodos que creamos son:

Constructor: Inicializa los atributos nombre, peso, y colorPiel con los valores proporcionados y establece el atributo transformación como "Humano".

Método transformar: Cambia el estado del Hulk a "Bestia". Multiplica el peso por 5, cambia el colorPiel a "Verde" y actualiza transformación a "Bestia".

Método calmar: Revierte la transformación de Hulk a "Humano". Divide el peso por 5, cambia el colorPiel a "Normal" y restablece transformación a "Humano".

Método saludar: Imprime un saludo basado en el estado de transformación. Si Hulk está en "Humano", saluda usando su nombre; si está en "Bestia", emite un grito. Nota: Utiliza el operador == para comparar transformación con "Humano", pero es recomendable usar equals() para comparar cadenas en Java.

Métodos Getters: Estos métodos son ejemplos de **encapsulamiento**, ya que permiten acceder a los atributos privados de manera controlada.

- **getPeso()**: Retorna el valor del atributo peso.
- **getColorPiel()**: Retorna el valor del atributo colorPiel.

Punto cuatro:

```
package BARCO;

public class BTRANSPORTADOR {
    private String nombre;
    private double calado;
    private double manga;
    private double eslora;
    private double capacidadCarga;
    private double cargaActual;
    private double velocidadCrucero;
    private String estado;
```

Explicación del código realizado:

La clase “BTRANSPORTADOR” define un objeto con sus atributos, en este caso un barco que transporta granos de un puerto hacia otro.

Los atributos que utilizamos son :

- String nombre: Define el nombre del barco, en este caso se llama “TUDS”.
- double calado: Se refiere a la parte del barco que se sumerge durante la navegación.
- double manga: Se refiere al ancho del barco.
- double eslora: Determina el largo del barco.
- double capacidadCarga: La capacidad de carga que el barco tiene.
- double cargaActual: Valor actual de la carga.
- double velocidadCrucero: Velocidad del barco.
- String estado: Puede estar atracado o navegando.

Los métodos que creamos son:

Constructor: Inicializa los atributos con los valores proporcionados.

Método atracarEnPuerto: Recibe como parámetro el nombre del puerto de valor tipo string, donde se accede al atributo estado, modificandolo y mostrando un mensaje donde se detalla que el barco está atracado en el puerto que se le asignó en este ejemplo “Puerto Madryn”.

Método navegarHaciaPuerto: Su funcionalidad es la misma que el anterior método, solo que modifica el estado a navegando hacia el puerto que se le asignó.

Método cargar: Este método recibe como parámetro “cantidad” de valor tipo double, dentro del mismo se ejecuta un una condicion IF , donde el estado es distinto a null y el estado es “Atracado”, se ejecuta otro IF, que suma la carga actual y la cantidad a cargar si es menor o igual a la capacidad si se cumple, la carga actual se suma con la cantidad a cargar, luego se imprime un mensaje, mostrando la carga y su estado actual.

De lo contrario, se imprime un mensaje “No se puede cargar, sobrepasaria la capacidad de carga..” y si no llegase a cumplirse el primer IF, es por que el barco no se encuentra atracado, se muestra un mensaje dando aviso “El barco debe estar atracado…”.

Método descargar: Este método es similar al anterior, solo cambia su lógica en su segunda condición IF, si la carga es mayor o igual a la cantidad ambas se restan, y se imprime un mensaje mostrando la cantidad descargada y el valor actual de la carga.

De lo contrario no se puede descargar, y se imprime un aviso por pantalla, “La cantidad es mayor a la actual.”

Si la primera condición no se cumple es por que el barco se encuentra navegando, de la misma manera se imprime un mensaje “El barco debe estar atracado en un puerto”.

IMAGENES DEL CÓDIGO.

```
// Constructor
public BTRANSPORTADOR(String nombre, double calado, double manga, double eslora, double capacidadCarga, double velocidadCrucero) {
    this.nombre = nombre;
    this.calado = calado;
    this.manga = manga;
    this.eslora = eslora;
    this.capacidadCarga = capacidadCarga;
    this.cargaActual = 0;  // Inicialmente, el barco está vacío
    this.velocidadCrucero = velocidadCrucero;
    this.estado = null;  // El barco puede estar "Atracado en X" o "Navegando hacia X"
}
```

```java
// Métodos
public void atracarEnPuerto(String puerto) {
    this.estado = "Atracado en el puerto " + puerto;
    System.out.println(nombre + " ha atracado en el puerto " + puerto + ".");
}

public void navegarHaciaPuerto(String puerto) {
    this.estado = "Navegando hacia el puerto " + puerto;
    System.out.println(nombre + " está navegando hacia el puerto " + puerto + ".");
}

public void cargar(double cantidad) {
    if (estado != null && estado.contains("Atracado")) {
        if (cargaActual + cantidad <= capacidadCarga) {
            cargaActual += cantidad;
            System.out.println(cantidad + " toneladas de carga han sido cargadas. Carga actual: " + cargaActual + " toneladas.");
        } else {
            System.out.println("No se puede cargar, sobrepasaría la capacidad máxima del barco.");
        }
    } else {
        System.out.println("El barco debe estar atracado en un puerto para cargar.");
    }
}

public void descargar(double cantidad) {
    if (estado != null && estado.contains("Atracado")) {
        if (cargaActual >= cantidad) {
            cargaActual -= cantidad;
            System.out.println(cantidad + " toneladas de carga han sido descargadas. Carga actual: " + cargaActual + " toneladas.");
        } else {
            System.out.println("No se puede descargar, la cantidad es mayor a la carga actual.");
        }
    } else {
        System.out.println("El barco debe estar atracado en un puerto para descargar.");
    }
}
```

```java
    public static void main(String[] args) {
        // Crear un barco transportador de granos
        BTRANSPORTADOR barco = new BTRANSPORTADOR("TUDS", 12.5, 20, 180, 50000, 25);

        // Atracar en un puerto
        barco.atracarEnPuerto("Puerto Madryn");

        // Cargar el barco
        barco.cargar(20000);

        // Navegar hacia otro puerto
        barco.navegarHaciaPuerto("Buenos Aires");


        // Intentar descargar mientras navega (no funcionará)
        barco.descargar(10000);

        // Atracar en un nuevo puerto y descargar
        barco.atracarEnPuerto("Buenos Aires");
        barco.descargar(10000);
    }
}
```

```
<terminated> BTRANSPORTADOR [Java Application] C:\Users\Leo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\
TUDS ha atracado en el puerto Puerto Madryn.
20000.0 toneladas de carga han sido cargadas. Carga actual: 20000.0 toneladas.
TUDS está navegando hacia el puerto Buenos Aires.
El barco debe estar atracado en un puerto para descargar.
TUDS ha atracado en el puerto Buenos Aires.
10000.0 toneladas de carga han sido descargadas. Carga actual: 10000.0 toneladas.
```