Clase 3

Interfaces y Colecciones

Clase abstracta

Definición

Clase que tiene al menos un método abstracto.

Un método abstracto es un método que define solo la descripción del método (nombre, parámetros que devuelve y que recibe) pero no su funcionalidad.

Características

- Clase base para definir otras clases.
- Se define con la palabra clave "abstract".
- No se puede instanciar directamente.
- Puede contener métodos abstractos y concretos.
- Las clases que la extienden deben implementar los métodos abstractos.
- Puede tener atributos y constructores.

Clase abstracta

```java
// Definición de la clase abstracta Persona
public abstract class Persona {
    // Atributo de instancia
    private String nombre;

    // Constructor
    public Persona(String nombre) {
        this.nombre = nombre;
    }

    // Método normal
    public void saludar() {
        System.out.println("Hola, mi nombre es " + nombre);
    }

    // Método abstracto
    public abstract void caminar();
}
```

Clase abstracta

```java
// Definición de la clase Empleado que hereda de Persona
public class Empleado extends Persona {
    // Constructor
    public Empleado(String nombre) {
        super(nombre);
    }

    // Implementación del método abstracto de Persona
    @Override
    public void caminar() {
        System.out.println("Caminando hacia el trabajo");
    }

    // Método estático
    public static void trabajar() {
        System.out.println("Trabajando...");
    }

    // Otro método cualquiera
    public void realizarTarea() {
        System.out.println("Realizando tarea asignada");
    }
}
```

Clase abstracta pura / Interfaces

Definición
Clase que tiene todos sus métodos definidos como métodos abstractos

Interfaz
Conceptualmente una interfaz es una clase abstracta pura, no se define como tipo clase, sino como interfaz.

Una clase implementa una interfaz.

A través de la implementación de una interfaz, una clase se compromete a proporcionar una implementación para cada uno de los métodos definidos en la interfaz (contrato)

Interfaces

Características

- No se puede crear un objeto de una interfaz, pero se puede declarar una variable de su tipo.
- Todos los métodos de una interfaz son públicos y abstractos, por lo que las clases que los implementan deben definir los métodos por sí mismas.
- Una clase puede implementar múltiples interfaces.
- Una interfaz puede contener constantes, pero estas son inmutables (final).
- Una interfaz puede extender otra interfaz utilizando la palabra clave 'extends'.
- Los servicios deberían implementar Interfaces

Interfaz

```java
// Definición de la interfaz Persona
public interface Persona {

    // Definición de métodos
    public void saludar();

    public void caminar();
}
```

Implementación de la interfaz

```java
// Definición de la clase Empleado que implementa Persona
public class Empleado implements Persona {
    // Atributo de instancia
    private String nombre;

    // Constructor
    public Empleado(String nombre) {
        super(nombre);
    }

    // Implementación de métodos de la interfaz Persona
    @Override
    public void caminar() {
        System.out.println("Caminando hacia el trabajo");
    }

    @Override
    public void realizarTarea() {
        System.out.println("Realizando tarea asignada");
    }
}
```

Colecciones

Definición
Conjunto de clases e interfaces que proporcionan una forma de almacenar y manipular grupos de objetos.

- API de colecciones, paquete java.util
- Manipulación de datos eficiente
- Manipular gran cantidad de datos
- Agregar, quitar o buscar elementos
- Hay que elegir la colección adecuada para ser eficiente

Clasificación
Listas (List)
Conjuntos (Set)
Mapas (Map)

Colecciones - Clasificación

Listas (List)
Colección ordenada de elementos, se puede acceder a los elementos por su posición, admite objetos repetidos.
Algunas implementaciones: ArrayList, LinkedList, Vector

Conjuntos (Set)
Colección sin orden específico, no admite elementos repetidos
Algunas implementaciones: HashSet, LinkedHashMap, TreeMap

Mapas (Map)
Colección que almacena pares clave-valor. La clave es única y se utiliza para acceder a los elementos del mapa.
La clave puede ser cualquier tipo de objeto, generalmente String

Clase Persona

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

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

    // Getters y setters
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getEdad() {
        return edad;
    }
    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

Ejemplo de List

```java
// Definir una lista de objetos de tipo Persona
List<Persona> miLista = new ArrayList<>();

Persona persona1 = new Persona("Juan", 25);
Persona persona2 = new Persona("Ana", 30);
Persona persona3 = new Persona("Pedro", 35);

miLista.add(persona1);
miLista.add(persona2);
miLista.add(persona3);

Persona elemento2 = miLista.get(1); // obteniendo el segundo elemento de la lista


miLista.remove(0);


int tamaño = miLista.size();
```

Ejemplo de Set

```
// Definir un Set de Persona
Set<Persona> miSet = new HashSet<>();

Persona persona1 = new Persona("Juan", 25);
Persona persona2 = new Persona("Ana", 30);
Persona persona3 = new Persona("Pedro", 35);

miSet.add(persona1);
miSet.add(persona2);
miSet.add(persona3);

miSet.remove(persona1);

miSet.clear();
```

Ejemplo de Map

```
// Definir un Map con la clase Persona como clave y valores String como valores asociados
Map<Persona, String> miMapa = new HashMap<>();

Persona persona1 = new Persona("Juan", 25);
Persona persona2 = new Persona("Ana", 30);

miMapa.put(persona1, "Programadora");
miMapa.put(persona2, "Diseñador");

String valor = miMapa.get(persona1);

// Definir un Map con la clase String como clave y valores Persona como valores asociados
Map<String, Persona> miMapa2 = new HashMap<>();

miMapa2.put("Programador", persona1);
miMapa2.put("Diseñadora", persona2);

String persona3 = miMapa2.get("Programador");
```

Cómo iterar colecciones

```
for(Persona persona : miLista) {
   System.out.println(persona.getNombre() + " tiene " + persona.getEdad() + " años.");
}

// Iterar sobre el set utilizando un iterator
Iterator<Persona> iterador = miSet.iterator();
while(iterador.hasNext()) {
   Persona persona = iterador.next();
   System.out.println(persona.getNombre() + " tiene " + persona.getEdad() + " años.");
}

// Iterar sobre el mapa utilizando un for-each loop
for(Map.Entry<String, Persona> entrada : miMapa.entrySet()) {
   String clave = entrada.getKey();
   Persona persona = entrada.getValue();
   System.out.println(clave + " tiene " + persona.getEdad() + " años.");
}
```