**S11.s2 - Trabajar con listas de elementos. Procesamiento de colecciones de objetos**

**Conocimientos previos**

**¿Qué es una colección en Java y para qué se utiliza en programación?  
Colección en Java: Es un conjunto de objetos que se manipulan como una unidad. Se utilizan para almacenar, recuperar, eliminar y ordenar datos de manera eficiente, adaptándose a diferentes tipos de datos y optimizando operaciones como búsqueda y ordenación.**

**¿Cuál es una implementación común de la interfaz List en Java y cómo se diferencia de otras implementaciones, como LinkedList?**

**ArrayList usa un arreglo dinámico y es eficiente en acceso por índice, pero menos en inserciones o eliminaciones en el medio de la lista.**

**LinkedList usa una lista doblemente enlazada, lo que hace eficiente la inserción y eliminación en cualquier parte de la lista, pero es más lento en el acceso por índice.**

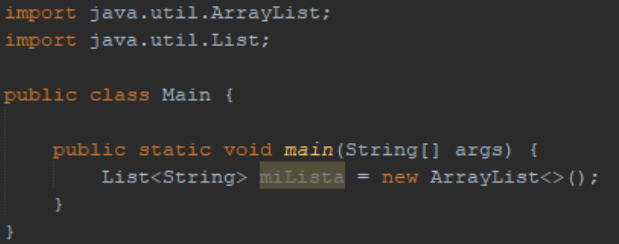
**¿Por qué sería útil crear una clase personalizada cuando trabajas con colecciones de objetos en Java?**

**Clases personalizadas en colecciones: Crear una clase personalizada es útil para modelar datos específicos, encapsular lógica de negocio, optimizar el rendimiento o extender la funcionalidad de las colecciones estándar, adaptándolas a necesidades particulares.  
  
Utilidad**

**¿Cómo puede el conocimiento sobre el procesamiento de listas y colecciones en Java facilitar la gestión de datos en tus programas?  
Procesamiento de listas y colecciones: Facilita la gestión de datos al permitir elegir la estructura adecuada (como ArrayList o LinkedList) para acceder, ordenar y manipular datos de forma eficiente según las necesidades del programa.**

**¿En qué situaciones podrías utilizar listas para organizar información y por qué sería útil?  
Uso de listas: Son útiles cuando necesitas mantener un orden específico de los elementos y cuando las operaciones de acceso y modificación son frecuentes, como en listas de tareas o filas de espera.**

**¿Por qué es importante comprender cómo trabajar con objetos personalizados en colecciones al desarrollar aplicaciones Java?  
Objetos personalizados en colecciones: Es clave para adaptar las colecciones a tus necesidades específicas, permitiendo definir cómo se comparan, ordenan o buscan los objetos, mejorando la eficiencia y precisión en el manejo de datos.  
  
Listas en Java**

**En Java, las listas son colecciones ordenadas de elementos que pueden contener objetos de cualquier tipo. Una de las implementaciones más utilizadas de la interfaz List es ArrayList. Para usar listas en Java, primero debes importar el paquete java.util y luego puedes crear una lista de la siguiente manera:  
**

**Ejemplo:**

**import java.util.Objects;**

**public class Empleado {**

**private String nombre;**

**private int edad;**

**private double salario;**

**public Empleado(String nombre, int edad, double salario) {**

**this.nombre = nombre;**

**this.edad = edad;**

**this.salario = salario;**

**}**

**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;**

**}**

**public double getSalario() {**

**return salario;**

**}**

**public void setSalario(double salario) {**

**this.salario = salario;**

**}**

**@Override**

**public String toString() {**

**return "Empleado{nombre='" + nombre + "', edad=" + edad + ", salario=" + salario + "}";**

**}**

**@Override**

**public boolean equals(Object o) {**

**if (this == o) return true;**

**if (o == null || getClass() != o.getClass()) return false;**

**Empleado empleado = (Empleado) o;**

**return edad == empleado.edad && Double.compare(empleado.salario, salario) == 0 && Objects.equals(nombre, empleado.nombre);**

**}**

**@Override**

**public int hashCode() {**

**return Objects.hash(nombre, edad, salario);**

**}**

**}  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**import java.util.\*;**

**public class GestionEmpleados {**

**public static void main(String[] args) {**

**List<Empleado> empleados = new ArrayList<>();**

**empleados.add(new Empleado("Carlos", 30, 2500));**

**empleados.add(new Empleado("Ana", 25, 3200));**

**empleados.add(new Empleado("Luis", 40, 1500));**

**empleados.sort(Comparator.comparingDouble(Empleado::getSalario).reversed());**

**System.out.println("Empleados ordenados por salario:");**

**for (Empleado empleado : empleados) {**

**System.out.println(empleado);**

**}**

**String nombreBuscado = "Ana";**

**Optional<Empleado> empleadoBuscado = empleados.stream()**

**.filter(emp -> emp.getNombre().equals(nombreBuscado))**

**.findFirst();**

**if (empleadoBuscado.isPresent()) {**

**System.out.println("\nEmpleado encontrado: " + empleadoBuscado.get());**

**} else {**

**System.out.println("\nEmpleado no encontrado.");**

**}**

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Cierre**

**¿Qué beneficios ofrece el uso de listas y colecciones en Java en comparación con el manejo manual de datos?  
Beneficios de listas y colecciones en Java: Ofrecen eficiencia, menor código, flexibilidad y seguridad de tipo, al permitir la gestión optimizada de datos sin necesidad de hacerlo manualmente. Facilitan operaciones como agregar, eliminar o buscar elementos y son más fáciles de mantener.**

**¿Cómo puede la comprensión de cómo funcionan las colecciones de objetos personalizados mejorar la estructura y eficiencia de tus aplicaciones Java?  
Comprensión de colecciones de objetos personalizados: Mejora la estructura y eficiencia de las aplicaciones, ya que optimiza operaciones de búsqueda, ordenación y manejo de datos complejos, y permite una organización más clara y escalable del código.**

**¿En qué situaciones específicas aplicarías las habilidades adquiridas en el procesamiento de colecciones en tu trabajo o proyectos de programación?  
Aplicaciones prácticas: Las habilidades de colecciones se aplican en gestión de grandes volúmenes de datos, sistemas de búsqueda y filtrado, estructuras de datos complejas y en el desarrollo de interfaces de usuario interactivas, mejorando el rendimiento y la flexibilidad en diversos proyectos de programación.**