Práctico 5: Colecciones

Objetivo:

Desarrollar estructuras de datos dinámicas en Java mediante el uso de colecciones y enumeraciones, optimizando su manipulación y recorrido.

Resultados de aprendizaje:

1. **Implementar y gestionar colecciones dinámicas con ArrayList**
   El estudiante podrá **implementar**, **modificar** y **gestionar** estructuras de datos dinámicas en Java utilizando `ArrayList`, aplicando métodos esenciales para la manipulación eficiente de elementos.
2. **Modelar e implementar relaciones 1 a N en UML y Java**
   El estudiante será capaz de **modelar**, **representar** y **programar** relaciones 1:N en diagramas UML e implementarlas en Java mediante colecciones, asegurando encapsulación y buenas prácticas.
3. **Aplicar el ciclo for-each para recorrer colecciones**
   El estudiante podrá **aplicar**, **comparar** y **utilizar** el ciclo `for-each` para recorrer colecciones de manera eficiente, diferenciándolo del `for` tradicional en términos de legibilidad y uso.
4. **Desarrollar algoritmos de búsqueda y ordenamiento en colecciones**
   El estudiante será capaz de **desarrollar**, **optimizar** e **implementar** algoritmos para buscar, ordenar y filtrar datos en colecciones, mejorando la eficiencia en la manipulación de datos.
5. **Definir y utilizar enumeraciones en Java**
   El estudiante podrá **definir**, **utilizar** y **evaluar** `enum` para representar conjuntos de valores predefinidos, mejorando la organización del código y la gestión de estados en aplicaciones.

¿Qué es una Kata y cómo se utiliza en programación?

Una kata es un ejercicio de programación diseñado para mejorar habilidades de codificación mediante la repetición y el aprendizaje progresivo. El término proviene de las artes marciales, donde las katas son secuencias de movimientos que se practican repetidamente para perfeccionar la técnica.

En programación, las katas ayudan a los programadores a reforzar conceptos, mejorar la comprensión del código y desarrollar buenas prácticas. **Se recomienda resolver una kata varias veces, intentando mejorar el código en cada iteración, utilizando mejores estructuras, nombres más claros y principios de diseño.**

Importante:

Intentar resolver cada kata sin mirar la solución.

Comprobar la solución y corregir errores si es necesario.

Repetir 2 o 3 veces para mejorar la comprensión, lógica y el código.

Experimentar con diferentes valores para reforzar el aprendizaje.

Resolver Katas para caso de sistema de stock

A continuación, te presento 3 enunciados de katas en Java para fortalecer los conceptos vistos en el módulo 5. En el Anexo 1(página 6), se deja la explicación del Flujo de Trabajo de un Sistema de Stock.

📌 Katas de Colecciones-ArrayList-Enumeraciones

🔹 Kata 1: Gestión Básica de Productos con Categoría (Nivel: Bajo)

Enunciado

Desarrollar una aplicación en Java que permita gestionar un inventario de productos en una tienda. Se debe agregar una categoría de producto utilizando Enums.

Clases

`Producto`

📌 **Atributos:**

- `id` **(String) → Identificador único del producto.**
- `nombre` **(String) → Nombre del producto.**
- `precio` **(double) → Precio del producto.**
- `cantidad` **(int) → Cantidad en stock.**
- `categoria` **(CategoriaProducto) → Categoría del producto.**

📌 **Métodos:**

- `mostrarInfo()`**: Muestra la información del producto.**

`CategoriaProducto (ENUM)

📌 Valores:

- ALIMENTOS
- ELECTRONICA
- ROPA
- HOGAR

`Inventario`

📌 Atributos:

- `productos` (ArrayList) → Lista de productos en el inventario.

📌 Métodos:

- `agregarProducto(Producto p)`: Agrega un producto a la lista.
- `listarProductos()`: Muestra todos los productos del inventario.

🎯 Tarea a realizar

1. Crear tres productos con diferentes categorías y agregarlos al inventario.
2. Listar los productos en el inventario mostrando su categoría.

🔹 Kata 2: Gestión de Stock con Búsqueda y Filtrado por Categoría (Nivel: Medio)

Enunciado

Ampliar la funcionalidad del sistema de stock agregando búsqueda por categoría y eliminación de productos.

Clases

`Producto` *(Misma estructura que la Kata 1)*

`Inventario`

📌 Nuevos métodos:

- `buscarProductoPorId(String id)`: Retorna un producto según su ID.
- `eliminarProducto(String id)`: Elimina un producto por su ID.
- `actualizarStock(String id, int nuevaCantidad)`: Actualiza la cantidad de un producto.
- `filtrarPorCategoria(CategoriaProducto categoria)`: Devuelve una lista de productos según su categoría.

🎯 Tarea a realizar

1. Agregar cuatro productos con distintas categorías al inventario.
2. Buscar un producto por su ID y mostrar su información.
3. Filtrar los productos que pertenezcan a la categoría ELECTRONICA y listarlos.
4. Eliminar un producto por su ID y listar los productos restantes.

🔹 Kata 3: Reportes Avanzados y Uso de Métodos en ENUM (Nivel: Alto)

Enunciado

Mejorar el sistema de stock agregando reportes avanzados y métodos útiles en la enumeración.

Clases

`Producto` *(Misma estructura que las Katas anteriores)*

`CategoriaProducto (ENUM)

📌 Valores: *(Mismos que antes)*
📌 Métodos:

- `getDescripcion()`: Retorna una descripción amigable de la categoría.

📌 Ejemplo de método dentro del Enum:

```
public enum CategoriaProducto {
   ALIMENTOS("Productos comestibles"),
   ELECTRONICA("Dispositivos electrónicos"),
   ROPA("Prendas de vestir"),
   HOGAR("Artículos para el hogar");

   private final String descripcion;

   CategoriaProducto(String descripcion) {
      this.descripcion = descripcion;
   }

   public String getDescripcion() {
      return descripcion;
   }
}
```

`Inventario`

📌 Nuevos métodos:

- `obtenerTotalStock()`: Devuelve la sumatoria de todas las cantidades de productos.
- `obtenerProductoConMayorStock()`: Retorna el producto con mayor stock.

- `filtrarProductosPorPrecio(double min, double max)`: Devuelve los productos dentro de un rango de precios.
- `mostrarCategoriasDisponibles()`: Muestra las categorías disponibles con su descripción.

🎯 Tarea a realizar

1. Agregar cinco productos con diferentes categorías al inventario.
2. Mostrar el total de stock disponible.
3. Obtener el producto con mayor stock y mostrarlo.
4. Filtrar productos con precios entre $1000 y $3000.
5. Mostrar las categorías disponibles con sus descripciones.

Anexo 1

📌 Flujo de Trabajo de un Sistema de Stock

Antes de empezar con las katas, es importante que el programador entienda cómo funciona un sistema de stock y qué elementos clave lo componen. A continuación, se explica el flujo de trabajo típico en un sistema de stock.

1 Concepto General

Un Sistema de Stock permite gestionar los productos de una tienda o empresa, controlando su disponibilidad, precios y ubicación. La información se almacena en estructuras de datos y se actualiza según las operaciones realizadas.

---

2 Flujo de Trabajo

El sistema de stock sigue el siguiente flujo:

📌 1. Agregar Productos al Inventario

Un usuario (ej. administrador de la tienda) ingresa nuevos productos al sistema.
Cada producto tiene atributos como:

- ID (único)
- Nombre
- Cantidad disponible
- Precio
- Categoría (ej. electrónica, alimentos, ropa)

Ejemplo:
📌 Un vendedor ingresa "Laptop HP" con ID `"HP123"`, precio `$1000` y cantidad `10 unidades`.

📌 2. Listar y Consultar Productos

El sistema debe permitir ver todos los productos registrados, mostrando la información de cada uno.

Ejemplo:
📌 El usuario solicita una lista de productos y el sistema devuelve:

ID: HP123 | Nombre: Laptop HP | Precio: $1000 | Cantidad: 10
ID: TV456 | Nombre: Smart TV | Precio: $700 | Cantidad: 5

📌 3. Buscar Productos por Criterios

El usuario puede buscar productos por:

- ID
- Nombre
- Categoría

Ejemplo:
📌 El usuario busca productos de la categoría ELECTRONICA, y el sistema devuelve:

ID: HP123 | Nombre: Laptop HP | Precio: $1000 | Cantidad: 10
ID: TV456 | Nombre: Smart TV | Precio: $700 | Cantidad: 5

📌 4. Actualizar Stock

Cada vez que se vende un producto, su cantidad disponible debe actualizarse.

Ejemplo:
📌 Se venden 2 laptops HP. El sistema actualiza la cantidad:

ID: HP123 | Nombre: Laptop HP | Precio: $1000 | Cantidad: 8

Si un producto queda con stock 0, puede notificarse al usuario.

📌 5. Eliminar Productos del Stock

Si un producto ya no se vende, puede eliminarse del sistema.

Ejemplo:
📌 Se decide eliminar un televisor del inventario.
El producto con ID "TV456" se elimina y ya no aparece en la lista.

📌 6. Reportes y Análisis de Stock

El sistema debe permitir generar reportes sobre: ✅ Total de productos en stock
✅ Producto con mayor stock disponible
✅ Productos en rango de precios (ej. $500 - $2000)
✅ Alertas de productos con stock bajo

Ejemplo:
📌 El usuario pide el producto con mayor stock y el sistema responde:

ID: XYZ789 | Nombre: Mouse Logitech | Stock: 50 unidades

3️⃣ Resumen Visual del Flujo de Trabajo

📌 Flujo General del Sistema de Stock
1️⃣ Agregar Producto → 2️⃣ Listar Productos → 3️⃣ Buscar Producto → 4️⃣ Actualizar Stock → 5️⃣ Eliminar Producto → 6️⃣ Generar Reportes

---

4️⃣ Relación con las Katas

Ahora que comprendemos el flujo de trabajo, podemos construir cada kata enfocándonos en una funcionalidad:

- **Kata 1: Agregar y listar productos**
- **Kata 2: Búsqueda, actualización y eliminación**
- **Kata 3: Reportes avanzados y uso de Enums**

📌 **Objetivo final: Crear un sistema de stock en Java paso a paso, con una base sólida para futuras mejoras.**

🚀 **¡Ahora sí, a programar!**