Java Collections and Generics - Hands-on Activity

This repository contains a series of Java applications designed to demonstrate the fundamental concepts of the Java Collections Framework and Generic programming.

Project Structure

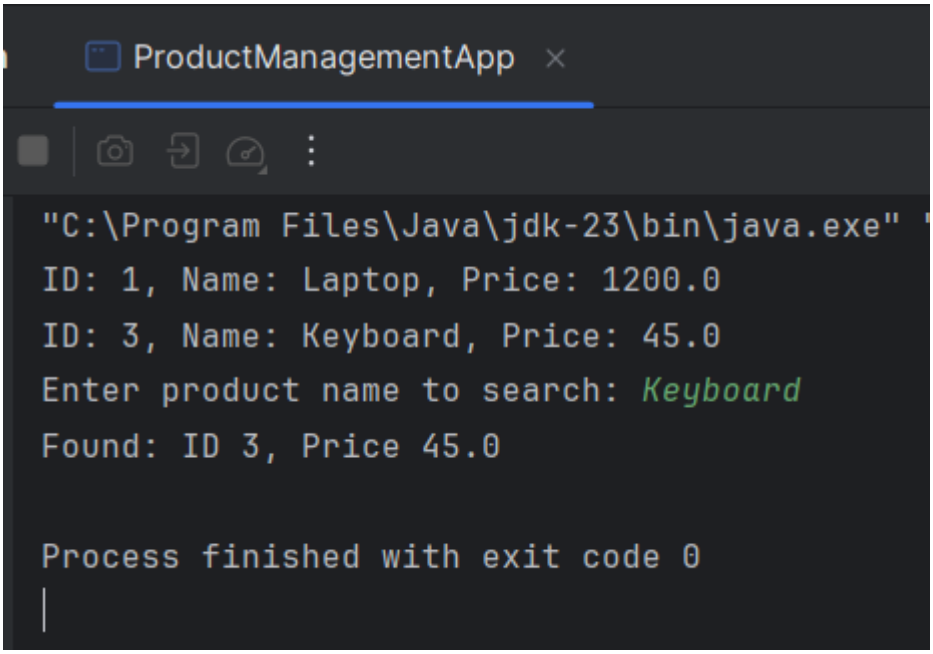
The project is organized into five distinct parts, each focusing on a specific aspect of Java development.

Part 1: Product Management with Lists

Objective: Manage a collection of products using the `ArrayList` class.

- **Key Classes:**
  - `Product`: A model representing a product with an ID, name, and price.
  - `ProductManagementApp`: Demonstrates list operations including adding, removing by index, updating product details, and searching by name.
- **Features:**
  - Dynamic storage using `ArrayList<Product>`.
  - Case-insensitive product search.
  - Basic CRUD (Create, Read, Update, Delete) operations on a list.

Execution Screenshot:



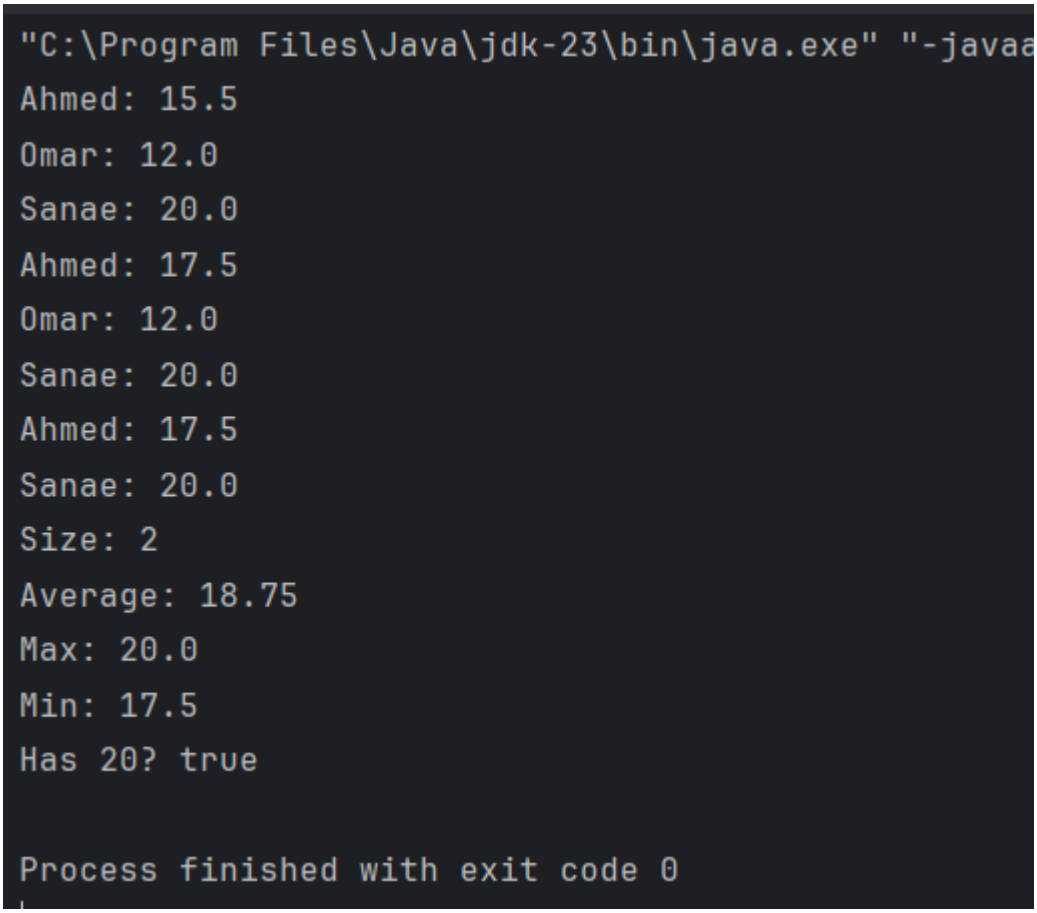
Part 2: Grade Management with Maps

Objective: Store and process student grades using a `HashMap` for efficient lookups.

- **Key Class:**

    - `GradeManagementApp`: Manages a map where student names are keys and their grades are values.
- **Features:**
    - Adding and updating grades using `put()`.
    - Calculating class statistics: Average, Maximum, and Minimum grades.
    - Searching for specific values using `containsValue()`.
    - Iterating through map entries using `forEach()`.

Execution Screenshot:

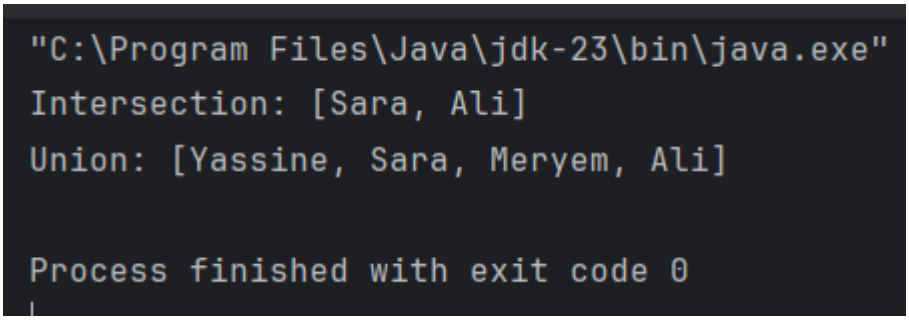


Part 3: Student Groups with Sets

Objective: Perform mathematical set operations using `HashSet`.

- **Key Class:**
    - `StudentSetApp`: Manages two separate sets of student names to identify relationships between them.
- **Features:**
    - **Intersection:** Uses `retainAll()` to find students common to both groups.
    - **Union:** Uses `addAll()` to combine students from both groups while automatically handling duplicates.

Execution Screenshot:

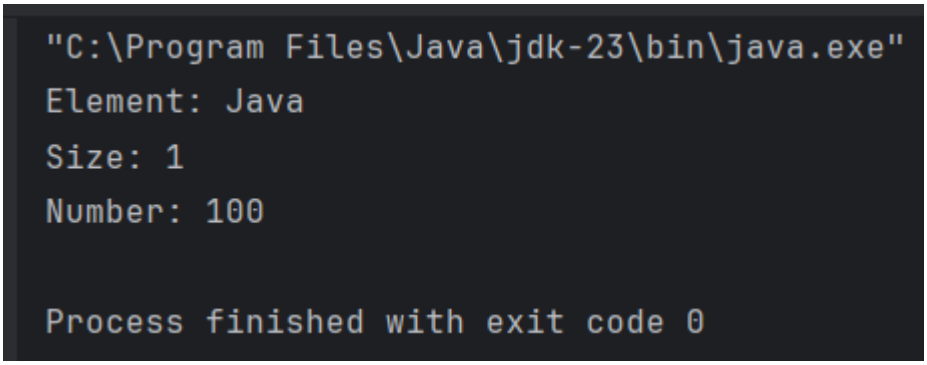


Part 4: Basic Generics

Objective: Create a reusable, type-safe storage container using Java Generics.

- **Key Classes:**
  - `GenericStorage<T>`: A generic class that can hold a list of any object type `T`.
  - `Application`: Demonstrates the class's flexibility by using it for both `String` and `Integer` types.
- **Features:**
  - Type safety at compile-time.
  - Generic methods for adding, retrieving, and removing elements.

Execution Screenshot:



Part 5: Advanced Generics and Interfaces

Objective: Implement a professional data access layer (DAO pattern) using generic interfaces.

- **Key Components:**
  - `IMetier<T>`: A generic interface defining standard data operations (`add`, `getAll`, `findById`, `delete`).
  - `MetierProduitImpl`: A concrete implementation of the interface specifically for `Product` objects.
  - `Application`: A console-based menu system that allows users to interact with the product database.
- **Features:**
  - Decoupling logic from implementation through interfaces.
  - Interactive command-line interface for real-time data management.

- Advanced search and deletion logic by object ID.

Execution Screenshot:

```
"C:\Program Files\Java\jdk-23\bin\java.exe"
1. Display Products
2. Search Product by ID
3. Add Product
4. Delete Product by ID
5. Exit
3
ID: 1
Name: Monitor
```

```
1. Display Products
2. Search Product by ID
3. Add Product
4. Delete Product by ID
5. Exit
1
1: Monitor
```

```
1. Display Products
2. Search Product by ID
3. Add Product
4. Delete Product by ID
5. Exit
2
ID: 1
Found: Monitor
```

```
1. Display Products
2. Search Product by ID
3. Add Product
4. Delete Product by ID
5. Exit
4
ID to delete: 1
```

```
1. Display Products
2. Search Product by ID
3. Add Product
4. Delete Product by ID
5. Exit
5

Process finished with exit code 0
```