Lecture 10 Class Example

Collections Framework & ArrayList: Gaming Inventory System

🎯 Topic Overview

Building a gaming inventory management system that demonstrates ArrayList, the foundation of Java's Collections Framework.

This hands-on example covers:

1. **ArrayList Creation** - Dynamic data structure that grows/shrinks automatically
2. **Adding Elements** - Using `.add()` to insert items
3. **Accessing Elements** - Using `.get()` and `.contains()` to retrieve data
4. **Removing Elements** - Using `.remove()` and `.clear()`
5. **Iterating** - Both traditional for loops and enhanced for-each loops
6. **Practical Application** - Real-world gaming inventory management

---

📚 Learning Goals

ArrayList Basics

- ✅ Understand ArrayList as a dynamic array that grows/shrinks automatically
- ✅ Create ArrayList using generic syntax: `ArrayList<Type> name = new ArrayList<>();`
- ✅ Recognize that ArrayList uses 0-based indexing (like arrays)
- ✅ Understand why ArrayList is better than arrays for most real-world applications

ArrayList Methods - Adding Elements

- ✅ Use `.add(E element)` to add elements to the end of the list
- ✅ Use `.add(int index, E element)` to insert at specific position
- ✅ Understand that ArrayList automatically grows as items are added
- ✅ Know that size is tracked automatically by `.size()` method

ArrayList Methods - Accessing Elements

- ✅ Use `.get(int index)` to retrieve element at specific index
- ✅ Use `.size()` to get total number of elements
- ✅ Use `.contains(Object o)` to check if element exists (returns boolean)
- ✅ Use `.indexOf(Object o)` to find position of element
- ✅ Understand bounds checking and when IndexOutOfBoundsException occurs

ArrayList Methods - Removing Elements

- ✅ Use `.remove(int index)` to remove element at specific position
- ✅ Use `.remove(Object o)` to remove first occurrence of value
- ✅ Use `.clear()` to remove all elements at once

- ✅ Understand that remaining elements shift to fill gaps

ArrayList Methods - Other Operations

- ✅ Use `.isEmpty()` to check if list is empty
- ✅ Use `.set(int index, E element)` to replace element
- ✅ Understand the difference between `.size()` and `.isEmpty()`

Iteration Techniques

- ✅ Iterate using traditional for loop with `.get()` method
- ✅ Iterate using enhanced for-each loop syntax
- ✅ Know which iteration method works best for different scenarios
- ✅ Understand loop counter management in different loop types

Integration & Application

- ✅ Build a dynamic inventory system using ArrayList
- ✅ Combine user input with ArrayList operations
- ✅ Perform practical operations: search, add, remove, display
- ✅ Recognize when to use ArrayList vs. array

🔑 Key Concepts Demonstrated

Concept	Example in Code
ArrayList Creation	`ArrayList<String> inventory = new ArrayList<>();`
Adding Items	`inventory.add("Legendary Sword")`
Getting Size	`inventory.size()` returns current element count
Accessing Item	`inventory.get(0)` returns first item
Checking Existence	`inventory.contains("Health Potion")` returns true/false
Removing Item	`inventory.remove(0)` removes first item
For Loop	`for (int i = 0; i < inventory.size(); i++)`
For-Each Loop	`for (String item : inventory)`
Clearing List	`inventory.clear()` removes all items

💡 Real-World Application

This example simulates a gaming inventory where:

- **Dynamic growth** - New items are collected during gameplay without predefined capacity
- **Quick lookup** - Check if you have a specific item using `.contains()`
- **Easy removal** - Drop items or use consumables with `.remove()`
- **Organized access** - Display inventory with automatic indexing and iteration

- **Flexible management** - Add/remove items without worrying about array bounds

---

🔄 ArrayList vs. Array

Feature	Array	ArrayList
Size	Fixed at creation	Dynamic (grows/shrinks)
Declaration	`type[] name = new type[size]`	`ArrayList<type> name = new ArrayList<>()`
Access	`array[index]`	`list.get(index)`
Add	Manual resizing needed	`.add(element)`
Remove	Manual resizing needed	`.remove(index)`
Check contains	Loop required	`.contains(element)`
Get size	`.length` property	`.size()` method
Performance	Faster	Slightly slower
Memory	More efficient	More memory overhead

When to use ArrayList: Nearly always in modern Java code! Flexibility and built-in methods outweigh minor performance cost.

🎮 Gaming Concepts Used

- **Inventory**: Collection of items player possesses
- **Item Addition**: Finding/collecting new items during gameplay
- **Item Removal**: Using consumables or dropping unwanted items
- **Inventory Check**: Verifying possession of required items
- **Dynamic Management**: Inventory size changes during gameplay

---

🚀 Extensions to Try

After completing this example:

1. Create an ArrayList of GameItem objects (from Lecture 9!)
2. Add a feature to sort the inventory alphabetically
3. Create a search function to find items by partial name
4. Implement a "max inventory size" with overflow warnings
5. Create multiple inventories (one per player character)