🎓 ArrayList Quick Reference Sheet

📌 Basic Syntax

```
// Create an ArrayList
ArrayList<Type> name = new ArrayList<>();

// Example
ArrayList<String> items = new ArrayList<>();
ArrayList<Integer> scores = new ArrayList<>();
ArrayList<Double> prices = new ArrayList<>();
```

➕ Adding Elements

```
// Add to end
items.add("sword");          // Adds to position: size
items.add("potion");

// Add at specific index
items.add(0, "shield");      // Inserts at beginning, shifts others

// Result: [shield, sword, potion]
```

🔍 Getting Elements

```
// Get by index (0-based)
String first = items.get(0);        // "shield"
String last = items.get(items.size() - 1);  // "potion"

// Safe access pattern
if (index >= 0 && index < items.size()) {
    String item = items.get(index);
}
```

❌ Removing Elements

```
// Remove by index
items.remove(0);              // Removes "shield"
```

```
// Remove by value (first occurrence)
items.remove("potion");       // Removes first "potion"

// Remove everything
items.clear();                // Now empty: []
```

🔎 Searching

```
// Check if contains
if (items.contains("sword")) {
    System.out.println("We have a sword!");
}

// Find index of item
int index = items.indexOf("sword");  // Returns index or -1
```

📏 Size & State

```
// Get number of elements
int size = items.size();     // How many items?

// Check if empty
if (items.isEmpty()) {       // true or false?
    System.out.println("No items!");
}

// Safe size check
if (size > 0) {
    // Safe to access items
}
```

🔄 Iteration Methods

Method 1: Traditional For Loop

```
for (int i = 0; i < items.size(); i++) {
    String item = items.get(i);
    System.out.println((i+1) + ". " + item);
}
// Use when: You need the index
```

Method 2: Enhanced For-Each

```
for (String item : items) {
    System.out.println(item);
}
// Use when: You only need the item (cleaner!)
```

Method 3: While Loop

```
int i = 0;
while (i < items.size()) {
    String item = items.get(i);
    System.out.println(item);
    i++;
}
// Use when: Need complex loop control
```

🎲 Collections Methods

```
// Import first!
import java.util.Collections;

// Shuffle (randomize)
Collections.shuffle(items);

// Sort (alphabetical/numeric)
Collections.sort(items);

// Reverse
Collections.reverse(items);
```

🛡️ Common Patterns

Pattern 1: Safe Get

```
if (index >= 0 && index < list.size()) {
    item = list.get(index);
}
```

Pattern 2: Remove While Iterating

```
// DON'T do this - causes skipping!
for (int i = 0; i < list.size(); i++) {
```

```
    list.remove(i);  // ❌ WRONG!
}

// DO this instead - loop backwards
for (int i = list.size() - 1; i >= 0; i--) {
    list.remove(i);  // ✅ CORRECT
}
```

Pattern 3: Copy Before Modifying

```
ArrayList<String> original = new ArrayList<>();
// ... add items ...

ArrayList<String> copy = new ArrayList<>(original);
Collections.shuffle(copy);
// original is unchanged!
```

Pattern 4: User Index to Code Index

```
// User sees: 1, 2, 3
// Code uses: 0, 1, 2

int userChoice = scanner.nextInt();  // User enters 1
int index = userChoice - 1;          // Convert to 0
String item = list.get(index);
```

⚠ Common Mistakes

❌ Wrong	✅ Correct
`for (int i = 0; i <= list.size(); i++)`	`for (int i = 0; i < list.size(); i++)`
`list.get(list.size())`	`list.get(list.size() - 1)`
`list.length`	`list.size()`
Modifying while looping forward	Loop backwards when removing
Not checking bounds	Always check `i < size()`
Forgetting generic type	`ArrayList<String>` not `ArrayList`

🎮 Gaming Examples

Example 1: Inventory Management

```
ArrayList<String> inventory = new ArrayList<>();
inventory.add("Sword");
inventory.add("Shield");
inventory.add("Potion");

// Display
for (int i = 0; i < inventory.size(); i++) {
    System.out.println((i+1) + ". " + inventory.get(i));
}

// Use item
int choice = scanner.nextInt();
String used = inventory.remove(choice - 1);
System.out.println("Used: " + used);
```

Example 2: Team Management

```
ArrayList<String> team = new ArrayList<>();
team.add("Aragorn");
team.add("Gandalf");
team.add("Legolas");

// Check if member exists
String search = "Frodo";
if (team.contains(search)) {
    System.out.println(search + " is on the team!");
} else {
    System.out.println(search + " is NOT on the team.");
}
```

Example 3: Battle Opponents

```
ArrayList<String> opponents = new ArrayList<>();
opponents.add("Goblin");
opponents.add("Orc");
opponents.add("Dragon");

// Randomize battle order
ArrayList<String> battleOrder = new ArrayList<>(opponents);
Collections.shuffle(battleOrder);

// Face first opponent
String firstEnemy = battleOrder.get(0);
System.out.println("Facing: " + firstEnemy);
```

📋 Method Reference Table

Method	What It Does	Returns	Example
`.add(e)`	Add to end	void	`list.add("item")`
`.add(i, e)`	Add at index	void	`list.add(0, "first")`
`.get(i)`	Get element	Element	`String s = list.get(0)`
`.remove(i)`	Remove at index	Element	`list.remove(0)`
`.remove(o)`	Remove object	boolean	`list.remove("item")`
`.contains(o)`	Check if exists	boolean	`if (list.contains("x"))`
`.indexOf(o)`	Find index	int	`int i = list.indexOf("x")`
`.size()`	Count elements	int	`int n = list.size()`
`.isEmpty()`	Check if empty	boolean	`if (list.isEmpty())`
`.clear()`	Remove all	void	`list.clear()`
`.set(i, e)`	Replace element	Element	`list.set(0, "new")`

🆚 ArrayList vs Array

Feature	Array	ArrayList
Size	Fixed	Dynamic
Declaration	`String[] arr = new String[3]`	`ArrayList<String> list = new ArrayList<>()`
Add	Manually shift	`.add()`
Remove	Manually shift	`.remove()`
Access	`arr[i]`	`list.get(i)`
Length	`.length` property	`.size()` method
Real use	Rarely (legacy)	Most Java code

👉 **Use ArrayList in almost all cases!**

🧠 Mental Model

Think of ArrayList as a **smart, growing list**:

```
ArrayList grows as needed:

Initial:     [              ]
Add 1 item:  [ Sword         ]
Add 2 items: [ Sword, Shield ]
```

```
Add 3 items: [ Sword, Shield, Potion ]
Remove 1:    [ Sword, Potion ]  (Shield removed, Potion shifted)
```

Key Insight: It's like a line of people - when someone leaves, everyone shifts forward!

💡 When to Use What

`.add()` - Adding items

```
// Adding new items to collection
inventory.add("Gold");
```

`.get()` - Using existing items

```
// Accessing specific item
String weapon = inventory.get(0);
```

`.remove()` - Taking items out

```
// Removing used/dropped items
inventory.remove(0);
```

`.contains()` - Checking inventory

```
// Checking if you have something
if (inventory.contains("Health Potion")) { ... }
```

`.size()` - Checking inventory space

```
// Knowing how much you're carrying
if (inventory.size() < MAX_SIZE) { ... }
```

🐛 Debugging Tips

Issue: IndexOutOfBoundsException

```
```

```
// ❌ Index is out of bounds
item = list.get(10);  // If list only has 5 items!

// ✅ Check bounds first
if (index < list.size()) {
    item = list.get(index);
}
```

Issue: Items appear to skip

```
// ❌ Removing while looping forward
for (int i = 0; i < list.size(); i++) {
    list.remove(i);  // Causes skipping!
}

// ✅ Loop backwards
for (int i = list.size() - 1; i >= 0; i--) {
    list.remove(i);
}
```

Issue: Wrong index when removing

```
// ❌ User enters 1, code uses as index 1
int userChoice = 1;
list.remove(userChoice);  // Removes wrong item!

// ✅ Convert to 0-based indexing
int index = userChoice - 1;
list.remove(index);
```

📞 Quick Help

I need to:

Task	Method
Add an item	`.add(item)`
Get an item	`.get(index)`
Remove an item	`.remove(index)`
Check if I have something	`.contains(item)`
Count items	`.size()`
Empty the list	`.clear()`

Task	Method
Loop through items	`for (... : list)` or `for (int i...`
Randomize order	`Collections.shuffle(list)`
Sort alphabetically	`Collections.sort(list)`

🎯 One-Liner Reference

```
// Create
ArrayList<String> list = new ArrayList<>();

// Add
list.add("item");

// Get
String item = list.get(0);

// Check
if (list.contains("item")) { }

// Size
int n = list.size();

// Remove
list.remove(0);

// Clear
list.clear();

// Loop
for (String item : list) { System.out.println(item); }
```

🎮 Remember

ArrayList is your friend for:

- ✅ Collections of items
- ✅ Dynamic sizing
- ✅ Easy add/remove
- ✅ Quick search
- ✅ Game inventories
- ✅ Lists of anything