Nayankumar Dhome
nayankumardhome@gmail.com

ArrayList in Java

Swipe Right →

Nayankumar Dhome
nayankumardhome@gmail.com

Introduction

ArrayList is a dynamic array implementation of the List interface in Java. Unlike arrays, which have a fixed size, an ArrayList can dynamically grow or shrink. It is part of java.util and is widely used in applications where random access and frequent additions/removals are required.

Swipe Right →

Nayankumar Dhome
nayankumardhome@gmail.com

How ArrayList Works Internally?

1. Underlying Data Structure
- ArrayList uses a resizable array (default capacity: 10 in Java 8+).
- Internally, it stores elements in an Object array (Object[] elementData).

2. Resizing Mechanism (Dynamic Growth)
- When the array is full, a new array with 1.5x the previous capacity is created, and elements are copied over.
- This growth mechanism prevents frequent resizing but still maintains performance.

3. Load Factor
- Unlike HashMap, ArrayList does not use a load factor.
- It simply increases size when required.

→

Swipe Right

Nayankumar Dhome
nayankumardhome@gmail.com

Pros & Cons

Advantages

- Fast random access (O(1)) using get(index).
- Auto-resizable, unlike arrays.
- Better performance than LinkedList for frequent retrievals.

Disadvantages

- Slower insertions/deletions in the middle (O(n)).
- More memory consumption due to resizing overhead.

Swipe Right

Nayankumar Dhome
nayankumardhome@gmail.com

When to Use ArrayList?

- When fast retrieval is needed (e.g., caching, lookups).
- When you don't know the size in advance.
- When insertion/removal at the end is frequent.

🚫 **Avoid ArrayList if:**

- Frequent insertions/deletions at the middle → Use LinkedList.
- Thread safety is needed → Use Vector or CopyOnWriteArrayList.

Swipe Right →

Nayankumar Dhome
nayankumardhome@gmail.com

Thread-Safety in ArrayList

ArrayList is not thread-safe 🚫. Use synchronized alternatives:

1. Collections.synchronizedList Approach

```java
List<String> syncList = Collections.synchronizedList(new ArrayList<>());
```

2. Using CopyOnWriteArrayList (Best for Multi-threading)

```java
CopyOnWriteArrayList<String> list = new CopyOnWriteArrayList<>();
```

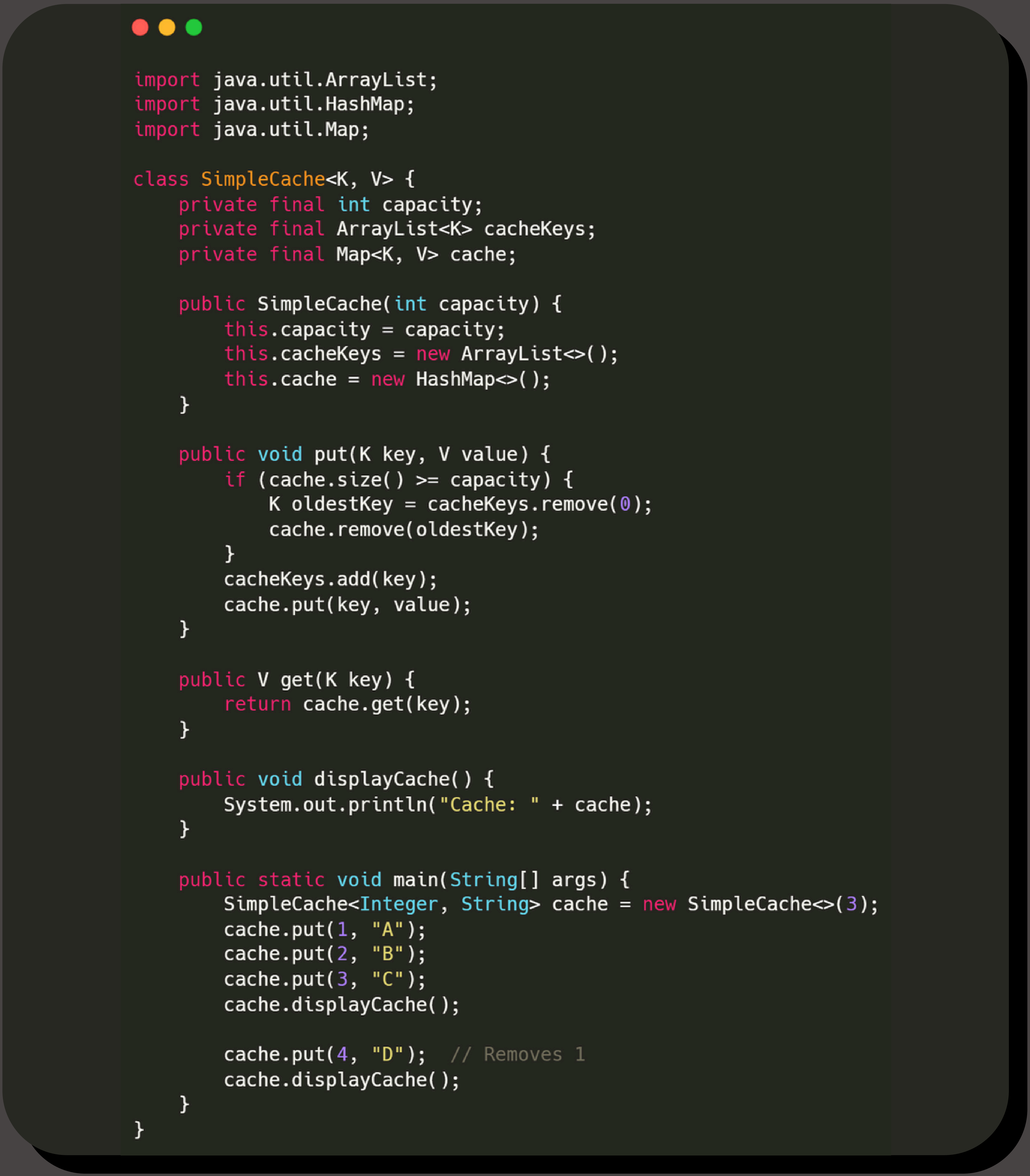
Swipe Right →

Nayankumar Dhome
nayankumardhome@gmail.com

Real-World Use Cases

Swipe Right →

1. Caching Mechanism



```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

class SimpleCache<K, V> {
    private final int capacity;
    private final ArrayList<K> cacheKeys;
    private final Map<K, V> cache;

    public SimpleCache(int capacity) {
        this.capacity = capacity;
        this.cacheKeys = new ArrayList<>();
        this.cache = new HashMap<>();
    }

    public void put(K key, V value) {
        if (cache.size() >= capacity) {
            K oldestKey = cacheKeys.remove(0);
            cache.remove(oldestKey);
        }
        cacheKeys.add(key);
        cache.put(key, value);
    }

    public V get(K key) {
        return cache.get(key);
    }

    public void displayCache() {
        System.out.println("Cache: " + cache);
    }

    public static void main(String[] args) {
        SimpleCache<Integer, String> cache = new SimpleCache<>(3);
        cache.put(1, "A");
        cache.put(2, "B");
        cache.put(3, "C");
        cache.displayCache();

        cache.put(4, "D");  // Removes 1
        cache.displayCache();
    }
}
```

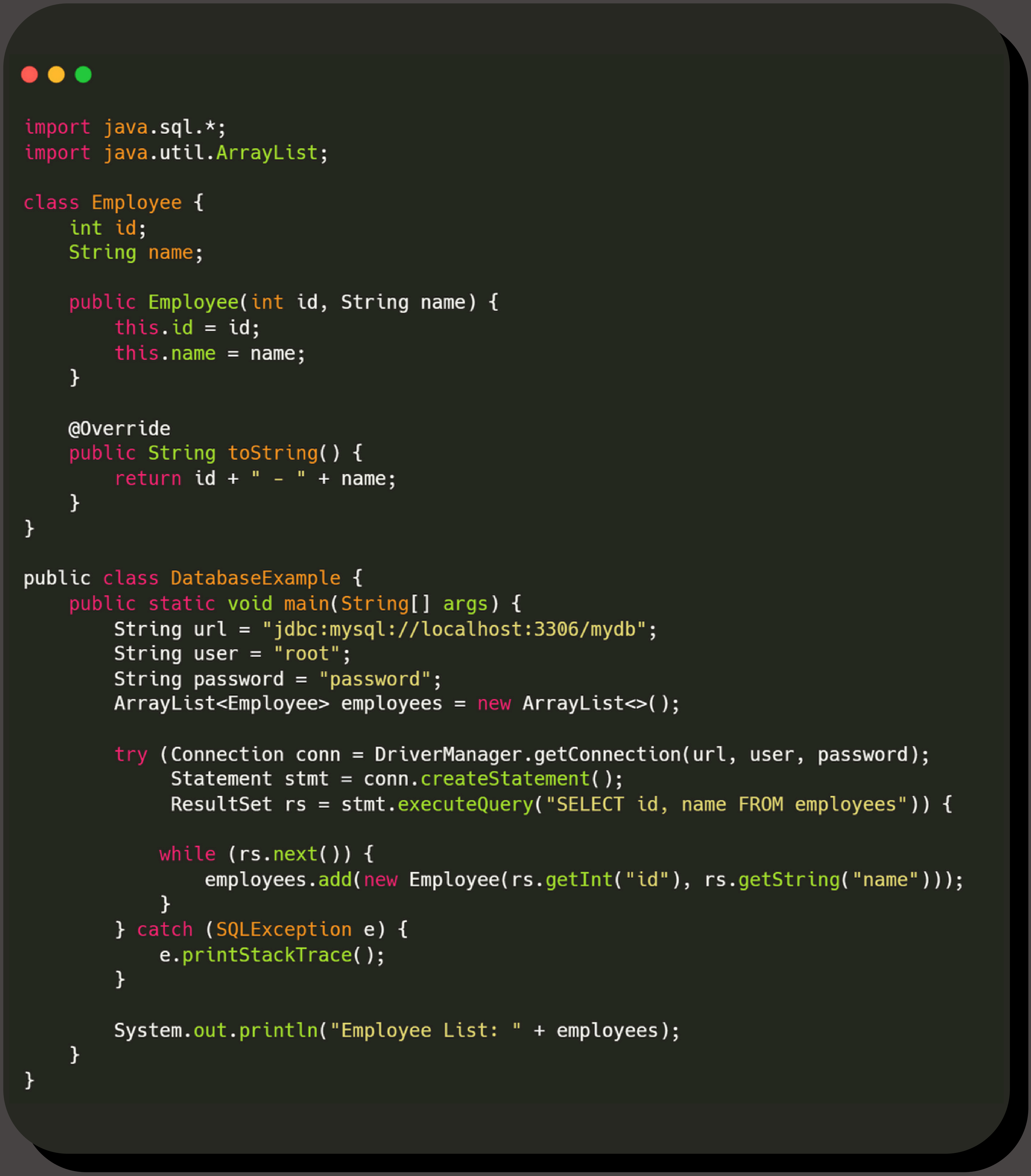
- **Use Case:** Fast, lightweight in-memory cache to store recent API responses.

Swipe Right →

Nayankumar Dhome
nayankumardhome@gmail.com

2. Storing Database Results

Nayankumar Dhome



```java
import java.sql.*;
import java.util.ArrayList;

class Employee {
    int id;
    String name;

    public Employee(int id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public String toString() {
        return id + " - " + name;
    }
}

public class DatabaseExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydb";
        String user = "root";
        String password = "password";
        ArrayList<Employee> employees = new ArrayList<>();

        try (Connection conn = DriverManager.getConnection(url, user, password);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT id, name FROM employees")) {

            while (rs.next()) {
                employees.add(new Employee(rs.getInt("id"), rs.getString("name")));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        System.out.println("Employee List: " + employees);
    }
}
```

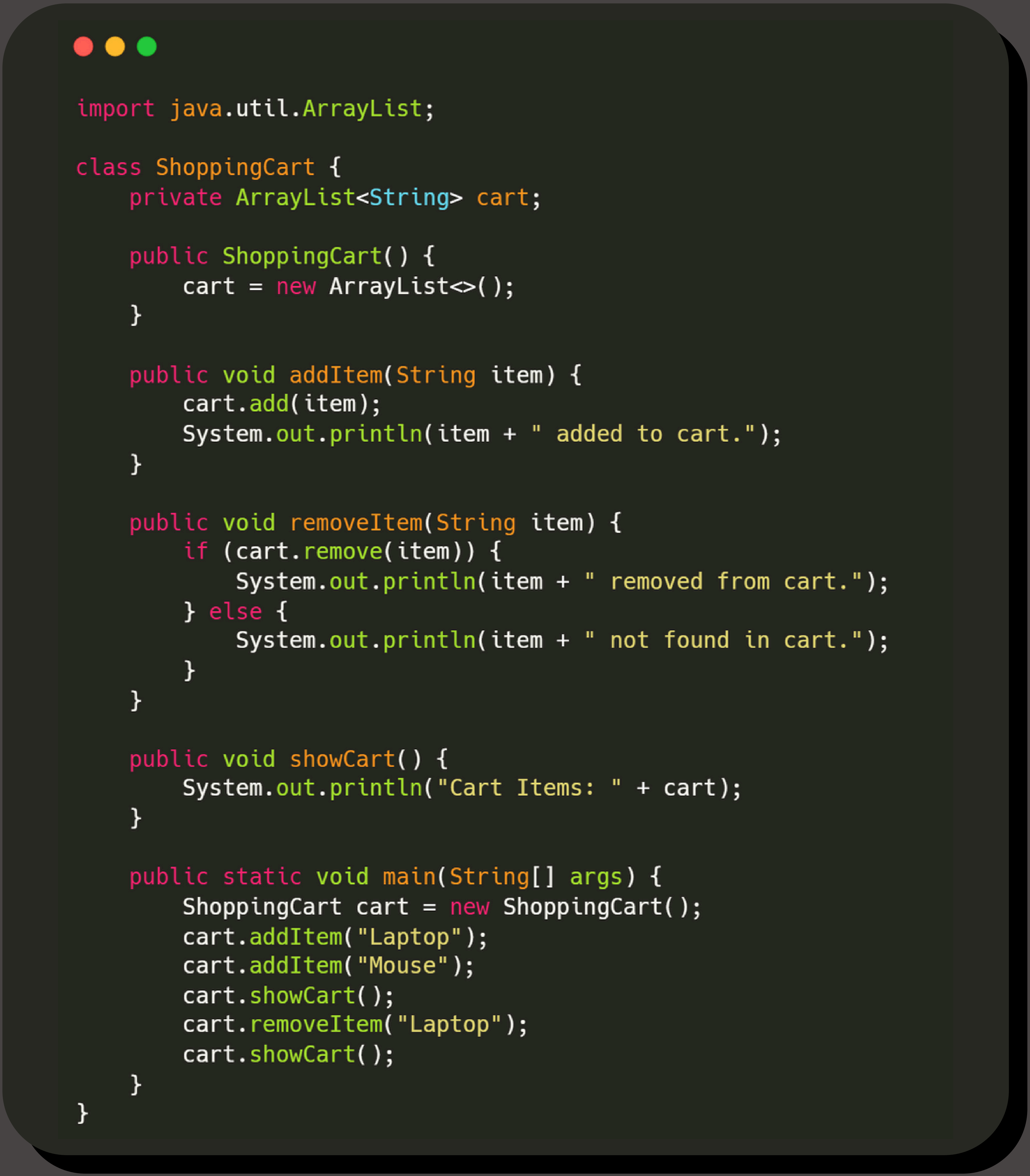
nayankumardhome@gmail.com

- **Use Case:** Storing database query results efficiently.

Swipe Right →

3. Maintaining a Dynamic List (e.g., Shopping Cart)

Nayankumar Dhome



```java
import java.util.ArrayList;

class ShoppingCart {
    private ArrayList<String> cart;

    public ShoppingCart() {
        cart = new ArrayList<>();
    }

    public void addItem(String item) {
        cart.add(item);
        System.out.println(item + " added to cart.");
    }

    public void removeItem(String item) {
        if (cart.remove(item)) {
            System.out.println(item + " removed from cart.");
        } else {
            System.out.println(item + " not found in cart.");
        }
    }

    public void showCart() {
        System.out.println("Cart Items: " + cart);
    }

    public static void main(String[] args) {
        ShoppingCart cart = new ShoppingCart();
        cart.addItem("Laptop");
        cart.addItem("Mouse");
        cart.showCart();
        cart.removeItem("Laptop");
        cart.showCart();
    }
}
```

nayankumardhome@gmail.com

- **Use Case:** Managing shopping cart items dynamically.

Swipe Right →

Nayankumar Dhome
nayankumardhome@gmail.com

Conclusion

- ArrayList is a powerful and flexible dynamic array.
- Best for frequent lookups and sequential insertions/removals.
- Avoid when frequent deletions/insertions at the middle are needed.
- Use thread-safe alternatives in multi-threaded applications.

Nayankumar Dhome
nayankumardhome@gmail.com

Follow us to get more information and tips like this.

@nayankumar-dhome

Repost