CRUD Operations Using ArrayList and Map in Java

Performing CRUD operations (Create, Read, Update, Delete) using an `ArrayList` and a `Map` in Java involves manipulating elements or key-value pairs through built-in methods.

What is an ArrayList?

An `ArrayList` is a resizable array implementation of the `List` interface in Java. It allows duplicate elements and maintains insertion order. Elements can be accessed by index.

What is a Map?

A `Map` is a data structure that stores key-value pairs. Each key is unique, and it maps to exactly one value. `HashMap` is a commonly used implementation.

Part 1: ArrayList

1. Create (Add Items)

```
import java.util.ArrayList;

public class CrudArrayList {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();

        // Create
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");

        System.out.println("Fruits after creation: " + fruits);
```

2. Read (Access Items)

```
        // Read
        System.out.println("First fruit: " + fruits.get(0));
        System.out.println("All fruits: " + fruits);
```

3. Update (Modify Items)

```
        // Update
        fruits.set(1, "Mango");  // Replaces "Banana" with "Mango"
        System.out.println("Fruits after update: " + fruits);
```

4. Delete (Remove Items)

```
        // Delete by value
        fruits.remove("Orange");

        // Delete by index
        fruits.remove(0);  // Removes "Apple"

        System.out.println("Fruits after deletion: " + fruits);
    }
}
```

Output:

```
Fruits after creation: [Apple, Banana, Orange]
First fruit: Apple
All fruits: [Apple, Banana, Orange]
Fruits after update: [Apple, Mango, Orange]
Fruits after deletion: [Mango]
```

Part 2: HashMap

1. Create (Put Key-Value Pairs)

```
import java.util.HashMap;

public class CrudMap {
    public static void main(String[] args) {
        HashMap<Integer, String> users = new HashMap<>();

        // Create
        users.put(1, "Alice");
        users.put(2, "Bob");
        users.put(3, "Charlie");

        System.out.println("Users after creation: " + users);
```

2. Read (Access Values by Key)

```
        // Read
        System.out.println("User with ID 2: " + users.get(2));
        System.out.println("All users: " + users);
```

3. Update (Modify Value by Key)

```
        // Update
        users.put(2, "Bobby");  // Updates the value for key 2
        System.out.println("Users after update: " + users);
```

4. Delete (Remove by Key)

```
        // Delete
        users.remove(1);  // Removes user with key 1
        System.out.println("Users after deletion: " + users);
    }
}
```

Output:

```
Users after creation: {1=Alice, 2=Bob, 3=Charlie}
User with ID 2: Bob
All users: {1=Alice, 2=Bob, 3=Charlie}
Users after update: {1=Alice, 2=Bobby, 3=Charlie}
Users after deletion: {2=Bobby, 3=Charlie}
```

Summary of Key Methods

ArrayList

Operation	Method	Example
Create	`add(value)`	`list.add("item")`
Read	`get(index)`	`list.get(0)`
Update	`set(index, value)`	`list.set(1, "new")`
Delete	`remove(value)`	`list.remove("item")`
Delete	`remove(index)`	`list.remove(0)`

HashMap

Operation	Method	Example
Create	`put(key, value)`	`map.put(1, "Alice")`
Read	`get(key)`	`map.get(1)`
Update	`put(key, value)`	`map.put(1, "Bob")`
Delete	`remove(key)`	`map.remove(1)`