Informatics Sport and Java Collections

Generated by LectureMate

February 22, 2025

1 Introduction

The lecture began with a brief introduction from Chris about Informatics Sport, a new organization offering various sports teams for students. The emphasis was on the benefits of joining these teams for social interaction, exercise, and enhancing CVs.

2 Course Updates

2.1 Quizzes

The lecturer addressed confusion regarding a quiz that was mistakenly based on last year's material. The quiz was opened for a week to allow students ample time to complete it.

2.2 Feedback

The lecturer reviewed feedback from the previous week, emphasizing the importance of effort and experience in the course. Most students found the course manageable and enjoyable.

3 Java Collections

The main topic of the lecture was Java Collections, specifically focusing on arrays and ArrayLists.

3.1 Arrays

Arrays are fixed in size, which can be limiting. They require a predetermined length at creation, making them rigid structures.

3.2 ArrayLists

ArrayLists are dynamic collections that can grow and shrink as needed. They provide more flexibility compared to arrays.

3.2.1 Declaration and Initialization

To declare an ArrayList, the syntax is as follows:

```
ArrayList<Type> listName = new ArrayList<Type>();
```

For example, to create an ArrayList of Strings:

```
ArrayList<String> cheers = new ArrayList<>();
```

3.2.2 Adding Elements

Elements can be added using the `add` method:

```
cheers.add("hip");
cheers.add("hooray");
```

3.2.3 Accessing Elements

To access elements, the `get` method is used:

```
String firstElement = cheers.get(0);
```

3.2.4 Removing Elements

Elements can be removed using the `remove` method:

```
cheers.remove("hip");
```

3.2.5 Iterating Over ArrayLists

Enhanced for loops can be used to iterate over elements:

```
for (String word : cheers) {
    System.out.println(word);
}
```

3.3 Primitive Types and Wrapper Classes

ArrayLists cannot directly hold primitive types (e.g., int, double). Instead, wrapper classes (e.g., Integer, Double) are used to convert primitives into objects.

3.4 Nested ArrayLists

ArrayLists can contain other ArrayLists, allowing for complex data structures. For example:

```
ArrayList<ArrayList<Double>> dailyTemperatures = new ArrayList<>();
```

4 HashMaps

HashMaps are used for associative arrays where keys are not limited to integers.

4.1 Declaration and Initialization

To declare a HashMap:

```
HashMap<String, Integer> wordLength = new HashMap<>();
```

4.2 Adding Key-Value Pairs

Key-value pairs can be added using the `put` method:

```
wordLength.put("example", 7);
```

4.3 Iterating Over HashMaps

To iterate over keys in a HashMap:

```
for (String key : wordLength.keySet()) {
    System.out.println(key + ": " + wordLength.get(key));
}
```

5 Conclusion

The lecture concluded with a summary of when to use ArrayLists versus HashMaps. ArrayLists are ideal for dynamic collections, while HashMaps are suited for key-value pair storage. The importance of iterators for traversing collections was also highlighted.

6 Questions and Future Lectures

The lecturer encouraged students to ask questions and reminded them of the upcoming lecture schedule, noting that there would be no teaching the following week.