



ADVANCED TOPICS

OBJECT ORIENTED PROGRAMMING I

Sercan Külçü | Object Oriented Programming I | 10.01.2023

Contents

Introduction	2
Generics.....	3
Concurrency	4
Performance Optimization	5
Annotations.....	6
Conclusion	7

Introduction

Java is a powerful and versatile programming language that can be used to develop a wide range of applications and systems. However, to fully leverage its potential, developers need to have a deep understanding of its advanced features and topics. In this chapter, we will explore some of the advanced topics of Java programming that can help developers build more sophisticated, complex applications and systems.

Generics

Generics allow developers to write code that can work with a variety of different data types. This is particularly useful when working with collections such as lists, maps, and sets. Generics can help developers avoid type casting errors and improve the safety and readability of their code.

To use generics in Java, you need to define a generic type parameter using the angle bracket syntax. For example, to define a list of strings, you would use the following code:

```
List<String> myList = new ArrayList<>();
```

Concurrency

Concurrency is the ability of a program to perform multiple tasks simultaneously. Java provides several features and libraries to support concurrent programming, such as threads, locks, and synchronized blocks. Concurrent programming can improve the performance and responsiveness of applications, particularly in multi-core environments.

To create a new thread in Java, you need to define a class that implements the `Runnable` interface and override the `run()` method. For example:

```
public class MyThread implements Runnable {  
    public void run() {  
        // thread code here  
    }  
}
```

To start the thread, you would use the following code:

```
public static void main(String[] args)  
{  
    Thread t = new Thread(new MyThread());  
    t.start();  
}
```

Performance Optimization

Performance optimization is the process of improving the speed and efficiency of an application. Java provides several tools and techniques for performance optimization, such as profiling, caching, and code optimization.

Profiling tools, such as the Java VisualVM, can help identify performance bottlenecks in your code. Caching can help reduce the number of database queries or network requests, improving the speed of your application. Code optimization techniques, such as loop unrolling and inlining, can help improve the efficiency of your code.

Annotations

Annotations are a way to add metadata to your code, such as documentation or runtime behavior. Java provides several built-in annotations, such as `@Override`, `@Deprecated`, and `@SuppressWarnings`. You can also define your own custom annotations.

To define a custom annotation in Java, you need to use the `@interface` keyword. For example:

```
@interface MyAnnotation {  
    String value();  
}
```

To use the annotation, you would apply it to a class, method, or variable using the `@MyAnnotation` syntax. For example:

```
@MyAnnotation("example value")  
public class MyClass {  
    // class code here  
}
```

Conclusion

The advanced topics of Java programming provide developers with a deeper understanding of the language, its features, and its potential applications. Generics can help improve the safety and readability of code. Concurrency can improve the performance and responsiveness of applications. Performance optimization techniques can help improve the speed and efficiency of code. Annotations can help add metadata to code and improve its documentation and behavior. By mastering these advanced topics, developers can build more sophisticated, complex applications and systems, optimize performance, and stay up-to-date with the latest developments and best practices in the field of Java programming.