Java 8 programs for practice

Java 8 programs for practice are essential for both beginners and experienced developers looking to sharpen their skills in this widely-used programming language. With the introduction of Java 8, numerous features were added that significantly changed how developers approach coding. These features include lambda expressions, the Stream API, and new date/time APIs, which enhance the functional programming capabilities of Java. In this article, we will explore various Java 8 programs that you can practice to solidify your understanding and application of these features.

Understanding Java 8 Features

Before diving into specific programs, it's crucial to understand the key features introduced in Java 8:

1. Lambda Expressions

Lambda expressions allow you to implement functional interfaces in a concise way. They enable you to write more readable and maintainable code.

Example:
A simple lambda expression can be used to create a thread:
```java
Runnable r = () -> System.out.println("Hello from a thread!");
new Thread(r).start();
```

2. Stream API

The Stream API provides a high-level abstraction for processing sequences of elements (like collections) in a functional style. This can lead to cleaner and more efficient code.

Example:
Filtering a list of integers:
```java
List numbers = Arrays.asList(1, 2, 3, 4, 5);
numbers.stream()
.filter(n -> n % 2 == 0)
.forEach(System.out::println);
```

3. New Date and Time API

Java 8 introduced a new date and time API, which is more intuitive and comprehensive than the previous `java.util.Date` and `java.util.Calendar`.

Example:
Getting the current date:
```java
LocalDate today = LocalDate.now();
System.out.println("Today's date: " + today);
```

```

# Java 8 Programs for Practice

Here are some Java 8 programs that you can practice to enhance your understanding of the new features.

## 1. Implement a Simple Calculator Using Lambda Expressions

Create a simple calculator that can perform addition, subtraction, multiplication, and division using lambda expressions.

```java
interface Calculator {
double operation(double a, double b);
}

public class CalculatorDemo {
public static void main(String[] args) {
Calculator addition = (a, b) -> a + b;
Calculator subtraction = (a, b) -> a - b;
Calculator multiplication = (a, b) -> a b;
Calculator division = (a, b) -> a / b;

System.out.println("Addition: " + addition.operation(10, 5));
System.out.println("Subtraction: " + subtraction.operation(10, 5));
System.out.println("Multiplication: " + multiplication.operation(10, 5));
System.out.println("Division: " + division.operation(10, 5));
}
}
```

2. Stream API: Filter and Sort a List of Employees

Suppose you have a list of employees with their names and salaries. Write a program to filter employees with a salary greater than a certain amount and sort them by name.

```java
import java.util.;
import java.util.stream.;

class Employee {
String name;
double salary;

Employee(String name, double salary) {
this.name = name;
this.salary = salary;
}

public String getName() {
return name;
}
```
```

```
public double getSalary() {
return salary;
}
}

public class EmployeeFilter {
public static void main(String[] args) {
List employees = Arrays.asList(
new Employee("John", 60000),
new Employee("Jane", 80000),
new Employee("Jack", 50000),
new Employee("Jill", 70000)
);

double threshold = 55000;
List filteredEmployees = employees.stream()
.filter(e -> e.getSalary() > threshold)
.sorted(Comparator.comparing(Employee::getName))
.collect(Collectors.toList());

filteredEmployees.forEach(e -> System.out.println(e.getName() + ": " + e.getSalary()));
}
}
```

3. New Date and Time API: Calculate Days Between Two Dates

Write a program that calculates the number of days between two dates using the new Date and Time API.

```java
import java.time.LocalDate;
import java.time.temporal.ChronoUnit;

public class DaysBetweenDates {
public static void main(String[] args) {
LocalDate startDate = LocalDate.of(2023, 1, 1);
LocalDate endDate = LocalDate.of(2023, 12, 31);

long daysBetween = ChronoUnit.DAYS.between(startDate, endDate);
System.out.println("Days between: " + daysBetween);
}
}
```

4. Creating a List of Squares Using Stream API

Create a program that generates a list of squares of numbers from 1 to 10 using the Stream API.

```java
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

public class SquareList {
public static void main(String[] args) {
```

```
List squares = IntStream.rangeClosed(1, 10)
.map(x -> x x)
.boxed()
.collect(Collectors.toList());

System.out.println("Squares: " + squares);
}
}
```

5. Grouping Employees by Salary Using Stream API

Write a program that groups employees based on their salary ranges using the Stream API.

```java
import java.util.;
import java.util.stream.Collectors;

public class EmployeeGrouping {
public static void main(String[] args) {
List employees = Arrays.asList(
new Employee("John", 60000),
new Employee("Jane", 80000),
new Employee("Jack", 50000),
new Employee("Jill", 70000),
new Employee("Joe", 60000)
);

Map groupedEmployees = employees.stream()
.collect(Collectors.groupingBy(e -> {
if (e.getSalary() < 60000) return "Low";
else if (e.getSalary() < 80000) return "Medium";
else return "High";
}));

groupedEmployees.forEach((k, v) -> {
System.out.println(k + ": " + v.stream().map(Employee::getName).collect(Collectors.joining(", ")));
});
}
}
```

Conclusion

Practicing Java 8 programs for practice is an effective way to familiarize yourself with the new features of Java 8. By working through various programs, you can apply concepts such as lambda expressions, the Stream API, and the new Date and Time API in real-world scenarios.

As you explore these examples, you will gain a deeper understanding of functional programming principles, which can lead to more efficient and effective coding practices. Keep experimenting with different problems and solutions to further enhance your Java skills. Happy coding!

Frequently Asked Questions

What are some basic Java 8 features I should practice with?

You should practice with lambda expressions, streams, optional, and the new Date and Time API.

How can I create a simple lambda expression in Java 8?

You can create a lambda expression by using the syntax (parameters) -> expression. For example, (a, b) -> a + b is a simple lambda that adds two numbers.

What is a stream in Java 8 and how can I use it?

A stream is a sequence of elements that can be processed in a functional style. You can use streams to filter, map, and reduce data by calling methods like filter(), map(), and reduce() on a collection.

Can you give an example of using Optional in Java 8?

Sure! You can use Optional to avoid null checks. For example: Optional<String> name = Optional.ofNullable(getName()); name.ifPresent(n -> System.out.println(n));

What is the new Date and Time API in Java 8?

The new Date and Time API is part of the java.time package, which provides a more comprehensive and user-friendly way to work with dates and times than the old java.util.Date.

How do I filter a list using streams in Java 8?

You can filter a list by calling the stream() method followed by filter(). For example: list.stream().filter(x -> x > 10).collect(Collectors.toList());

What are method references in Java 8?

Method references are a shorthand notation of a lambda expression to call a method. For example, instead of writing (s) -> System.out.println(s), you can use System.out::println.

How can I sort a list using streams in Java 8?

You can sort a list using the sorted() method of streams. For example: list.stream().sorted().collect(Collectors.toList());

What are default methods in interfaces introduced in Java 8?

Default methods allow you to add new methods to interfaces without breaking existing implementations. You define them using the 'default' keyword.

How can I parallelize operations using streams in Java 8?

You can parallelize operations by calling the parallelStream() method on a collection. For example: list.parallelStream().filter(x -> x > 10).collect(Collectors.toList());

Java 8 Programs For Practice

Java 8 Programs For Practice

Related Articles

- jira advanced roadmaps training
- italy travel guide
- john maxwell leadership training materials

Back to Home