# Exploring the BiConsumer Functional Interface in Java 8

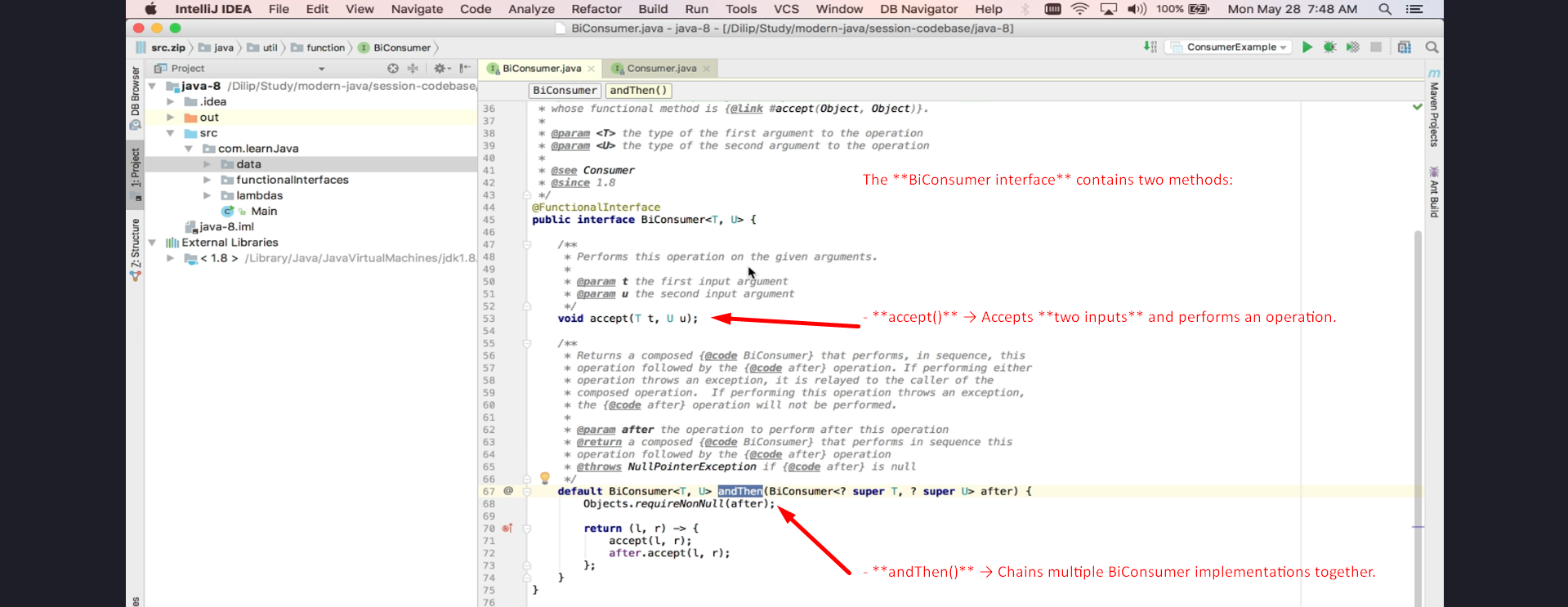
## Introduction

In this tutorial, we will \*\*code and explore\*\* the \*\*BiConsumer Functional Interface\*\* in Java 8. Each functional interface has **an extension**, and \*\***BiConsumer\*\* is an extension of the \*\*Consumer Functional Interface\*\*.**

## Methods in BiConsumer Interface

1. Open \*\*IntelliJ IDEA\*\* and press \*\*Shift\*\* twice to open the \*\*Search Everywhere\*\* window.

2. Search for \*\*BiConsumer Interface\*\* and examine its methods.

3. 

## Difference Between Consumer and BiConsumer

- \*\*Consumer\*\* → Accepts \*\*one input\*\* and performs an operation.

- \*\*BiConsumer\*\* → Accepts \*\*two inputs\*\* and performs an operation.

- If a scenario requires \*\*processing two inputs together\*\*, BiConsumer is preferred.

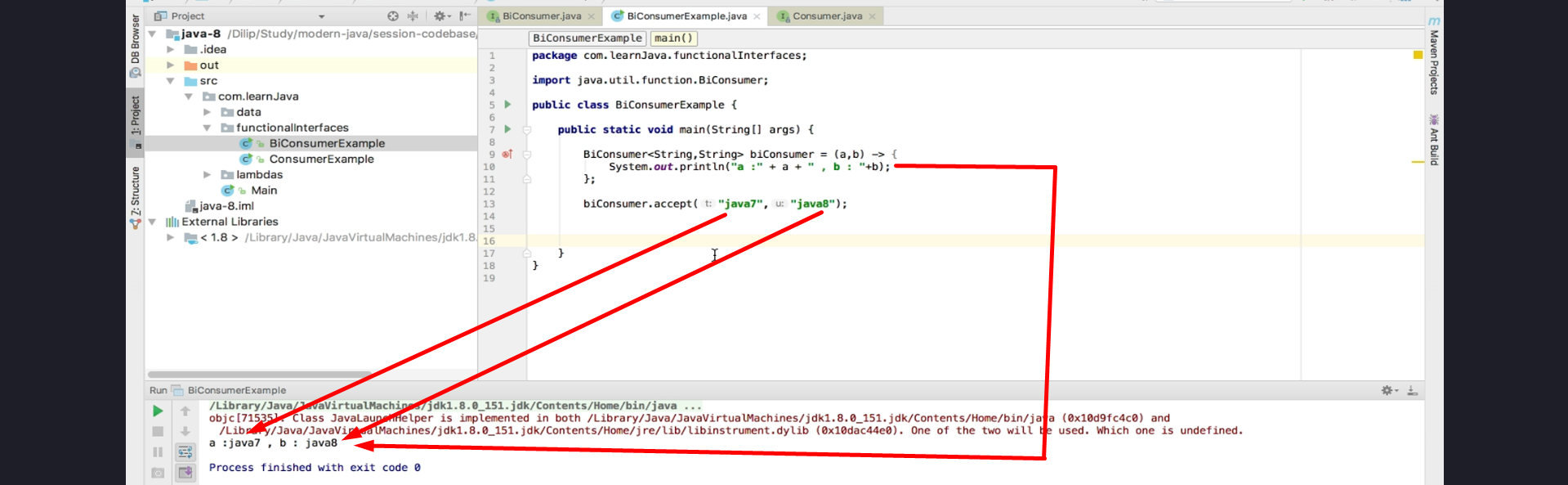
## Implementing BiConsumer Functional Interface

1. Create a new class \*\*BiConsumerExample\*\*.

2. Make the class executable by adding the \*\*public static void main()\*\* method.

3. Implement \*\*BiConsumer<String, String>\*\* to accept two String values and print them.

### \*\*Code Implementation:\*\*

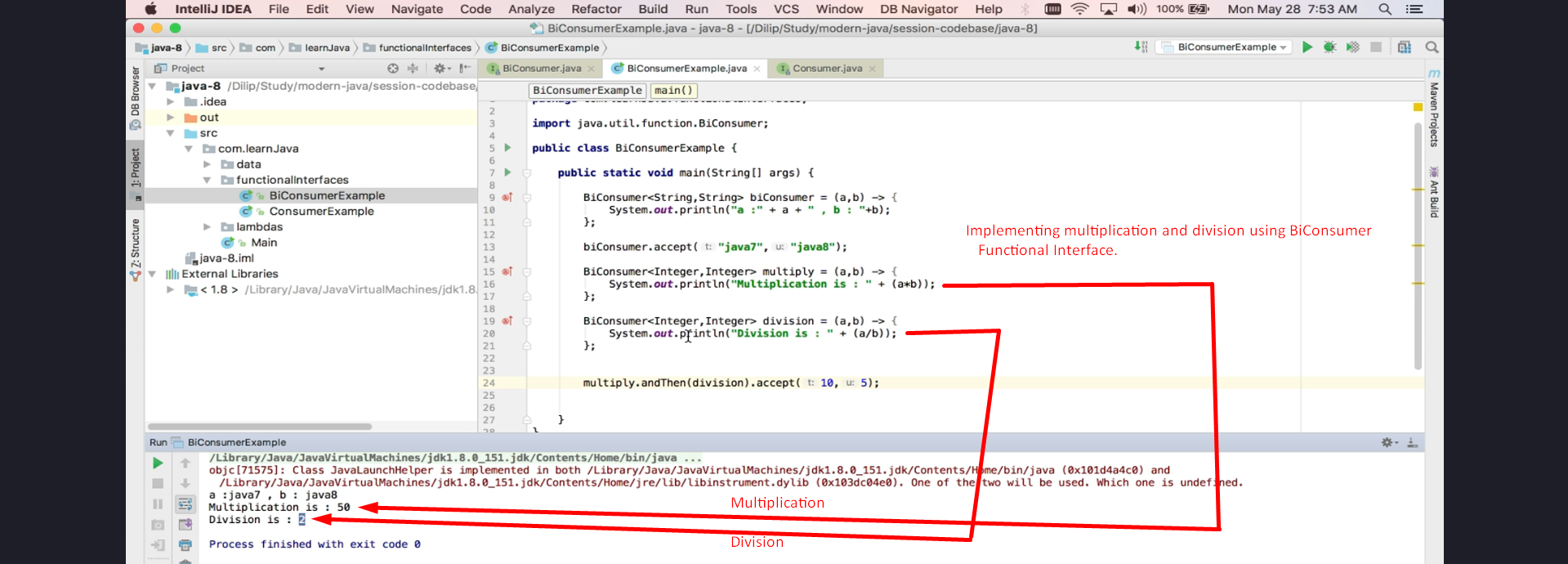


## Implementing Mathematical Operations Using BiConsumer

1. Modify \*\*BiConsumer\*\* to perform mathematical operations like multiplication and division.

2. Use `andThen()` method to \*\*chain operations together\*\*.

### \*\*Code Implementation:\*\*



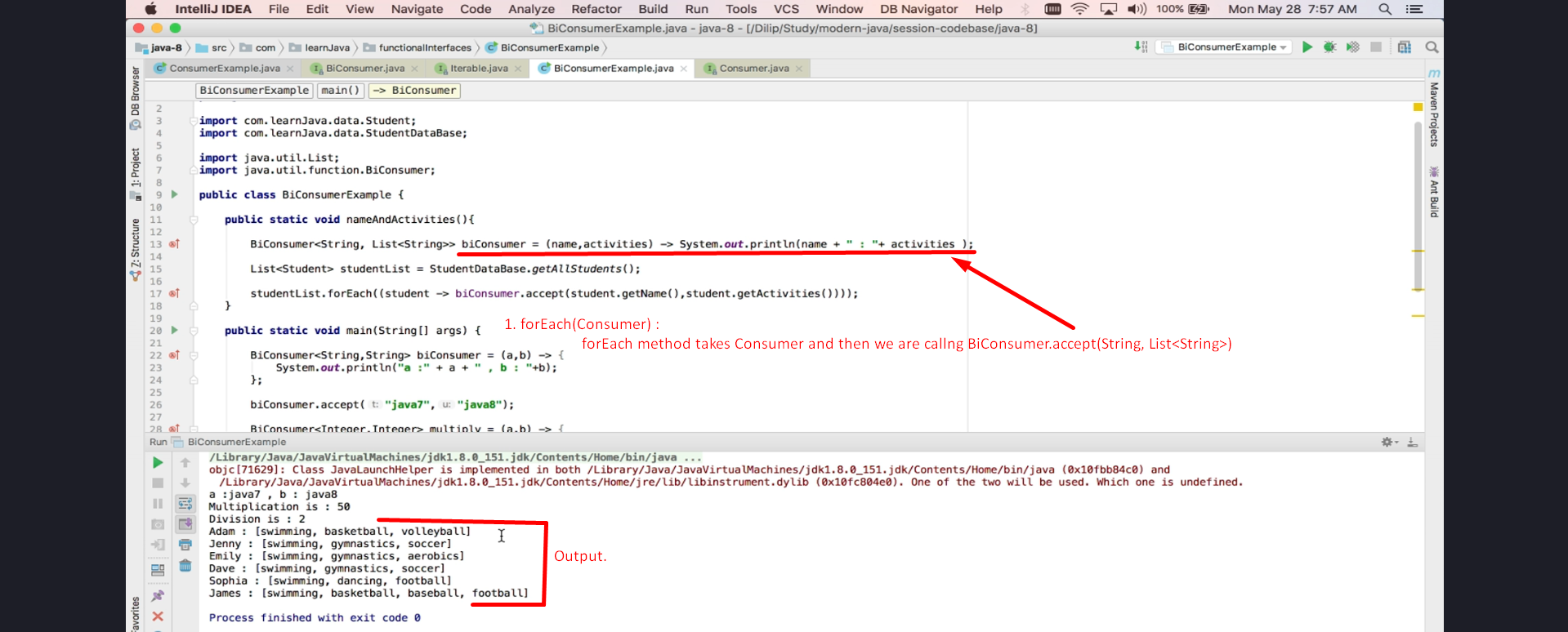
## Real-World Use Case: Student Name and Activities

1. Modify the previous \*\*ConsumerExample\*\* implementation to use \*\*BiConsumer\*\*.

2. Instead of handling \*\*name\*\* and \*\*activities\*\* separately, use BiConsumer to process both together.

3. Use `forEach()` to iterate over student data and print details.

### \*\*Code Implementation:\*\*



## Comparison Between Consumer and BiConsumer

1. \*\*Use Consumer\*\* when only \*\*one input\*\* is needed (e.g., printing student names).

2. \*\*Use BiConsumer\*\* when \*\*two inputs\*\* are required (e.g., name + activities).

3. There is \*\*no performance difference\*\* between Consumer and BiConsumer; use them as per your use case.

## Chaining Multiple BiConsumer Functional Interfaces

1. The \*\***BiConsumer.andThen()**\*\* method allows chaining multiple BiConsumers.

2. You can \*\*add N number of BiConsumers\*\* using `**andThen()**`.

### \*\*Code Implementation:\*\*

BiConsumer<Integer, Integer> add = (a, b) -> System.out.println("Addition: " + (a + b));  
BiConsumer<Integer, Integer> subtract = (a, b) -> System.out.println("Subtraction: " + (a - b));  
  
multiply.**andThen**(division).**andThen**(add).**andThen**(subtract).accept(10, 5); 🡸 **Chaining** **Multiple BiConsumer**.

## Conclusion

- \*\*BiConsumer\*\* **is an extension of the** \*\*Consumer Functional Interface\*\* that accepts \*\*two inputs\*\*.

- It is useful for \*\*processing two related inputs\*\* together (e.g., name and activities, mathematical operations).

- The \*\*andThen()\*\* method **allows chaining multiple BiConsumers** for sequential operations.

- Understanding \*\*when to use Consumer vs. BiConsumer\*\* helps in writing cleaner, reusable, and efficient Java 8 code.

### \*\*End of Tutorial\*\*

Thank you for watching!