

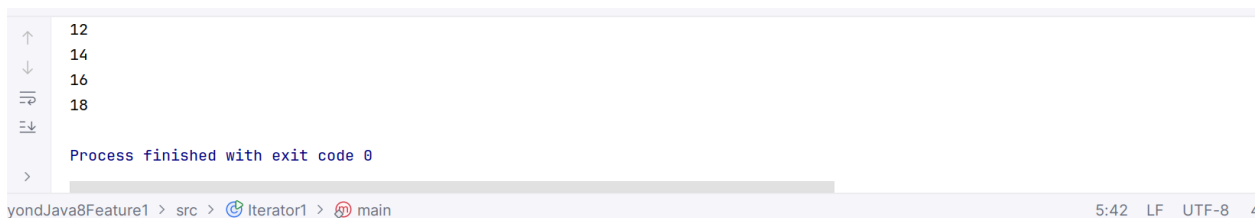
## 1. Use iterator stream method to generate a stream



The screenshot shows the IntelliJ IDEA interface. On the left, the 'Project' view displays the file structure of 'BeyondJava8Feature1', including 'src' and 'out' directories. The 'src' directory contains 'Iterator1', 'Main', and 'BeyondJava8Feature1.iml'. The 'out' directory is highlighted. On the right, the 'Main.java' file is open, showing the following code:

```
1 import java.util.stream.Stream;
2
3 public class Iterator1 {
4     public static void main(String[] args) {
5         Stream.iterate(0, Integer n -> n + 2)
6             .limit(10)
7             .forEach(System.out::println);
8     }
9 }
10
11
```

### Output

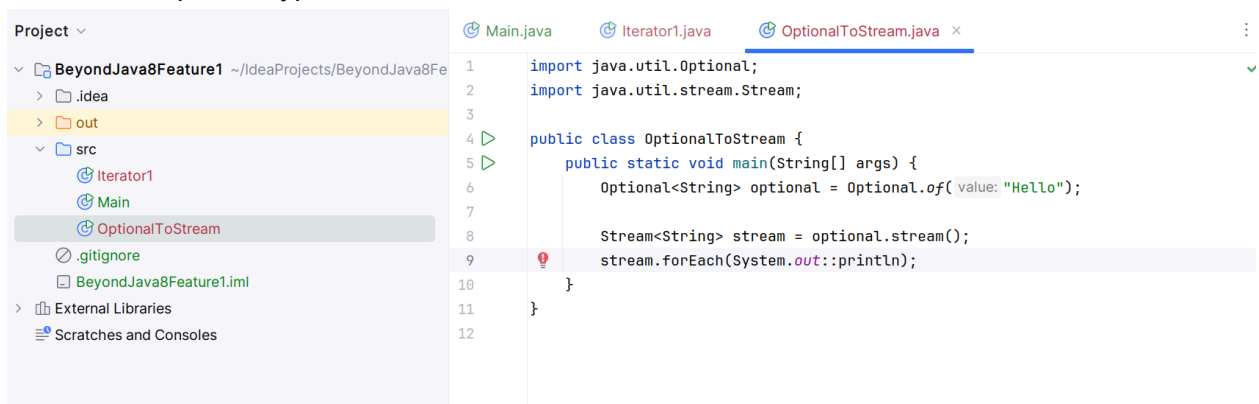


The screenshot shows the IntelliJ IDEA output console. The output is as follows:

```
12
14
16
18

Process finished with exit code 0
```


## 2. Convert an Optional type into Stream



The screenshot shows the IntelliJ IDEA interface. On the left, the 'Project' view displays the file structure of 'BeyondJava8Feature1', including 'src' and 'out' directories. The 'src' directory contains 'Iterator1', 'Main', and 'OptionalToStream'. The 'out' directory is highlighted. On the right, the 'OptionalToStream.java' file is open, showing the following code:

```
1 import java.util.Optional;
2 import java.util.stream.Stream;
3
4 public class OptionalToStream {
5     public static void main(String[] args) {
6         Optional<String> optional = Optional.of(value: "Hello");
7
8         Stream<String> stream = optional.stream();
9         stream.forEach(System.out::println);
10    }
11 }
12
```

### Output:

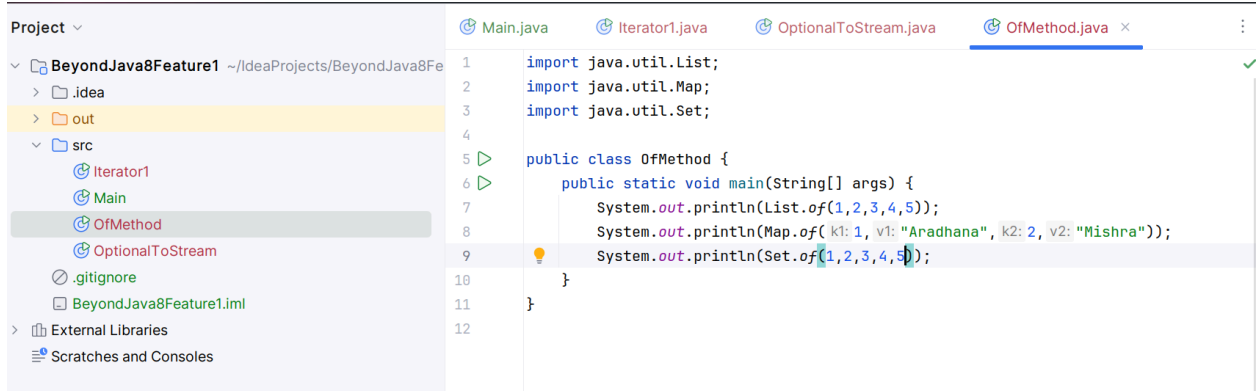


The screenshot shows the IntelliJ IDEA output console. The output is as follows:

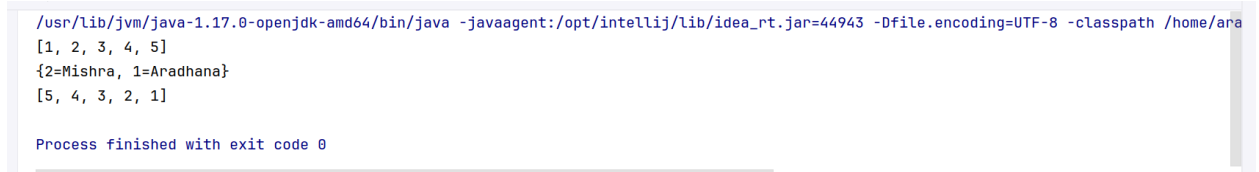
```
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=33755 -Dfile.encoding=UTF-8 -classpath /home/
Hello

Process finished with exit code 0
```

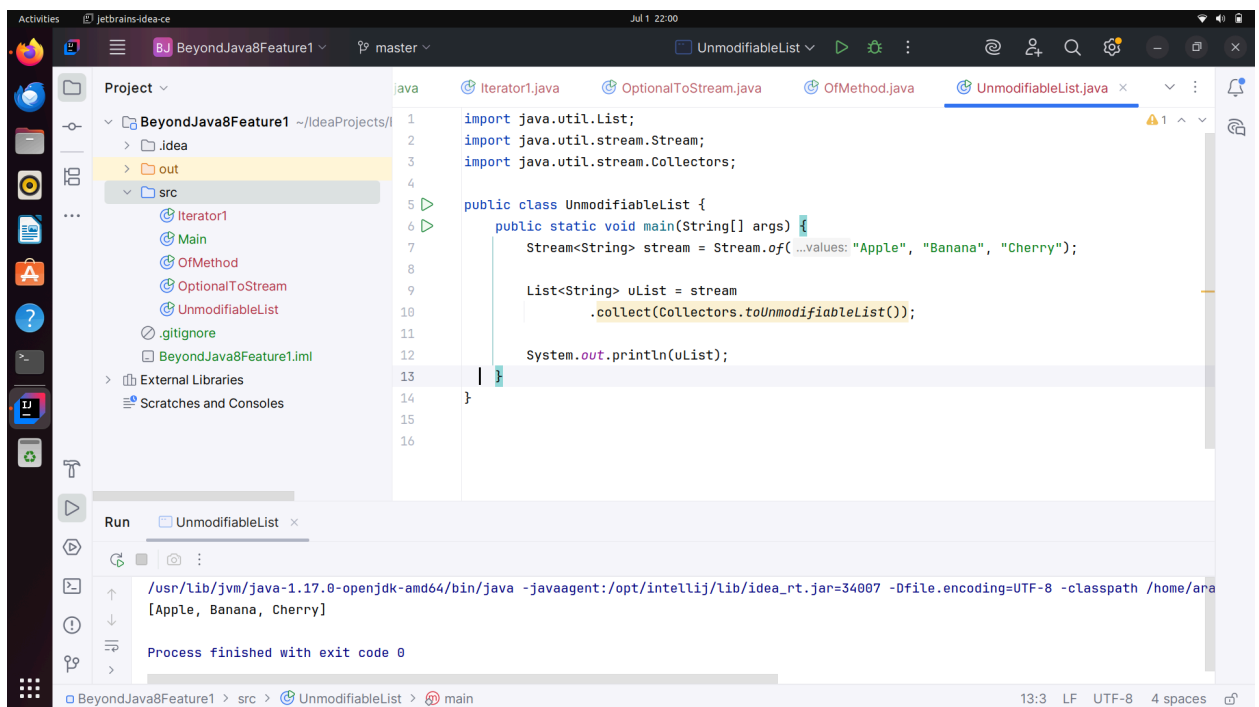
## 3. Use Of method to create List, Set and Map



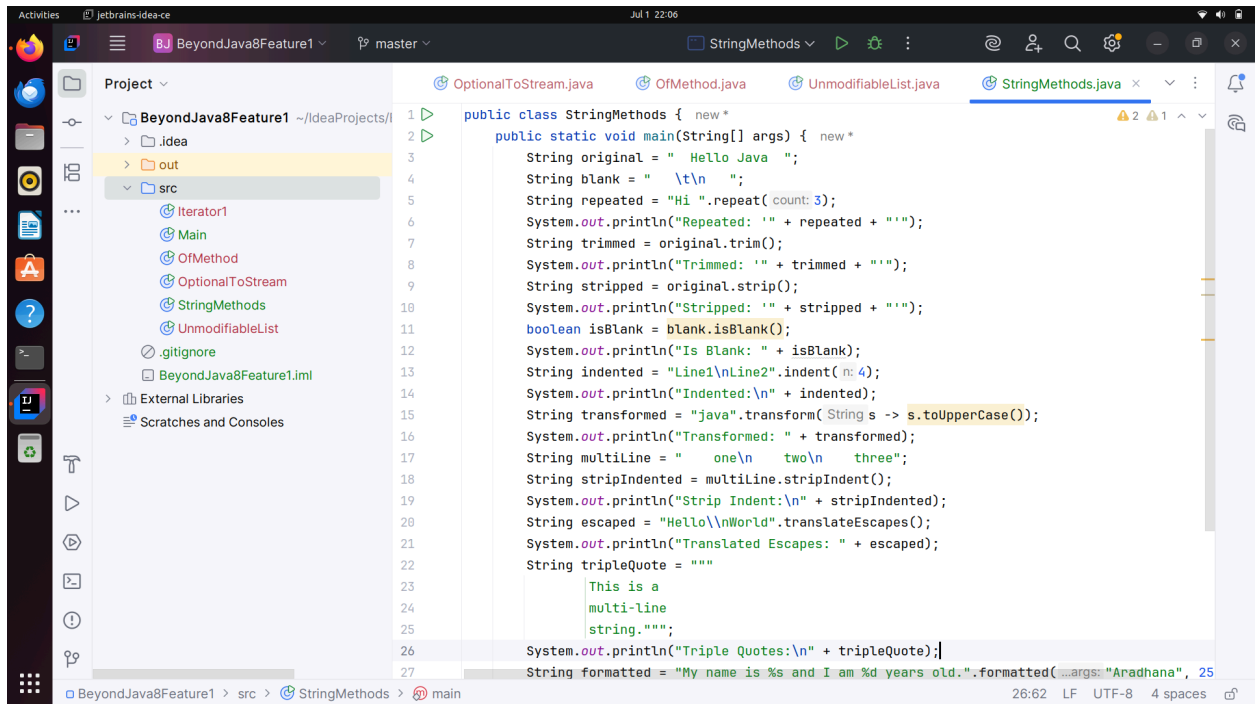
## Output



## 4. Create Unmodifiable List from a Steam

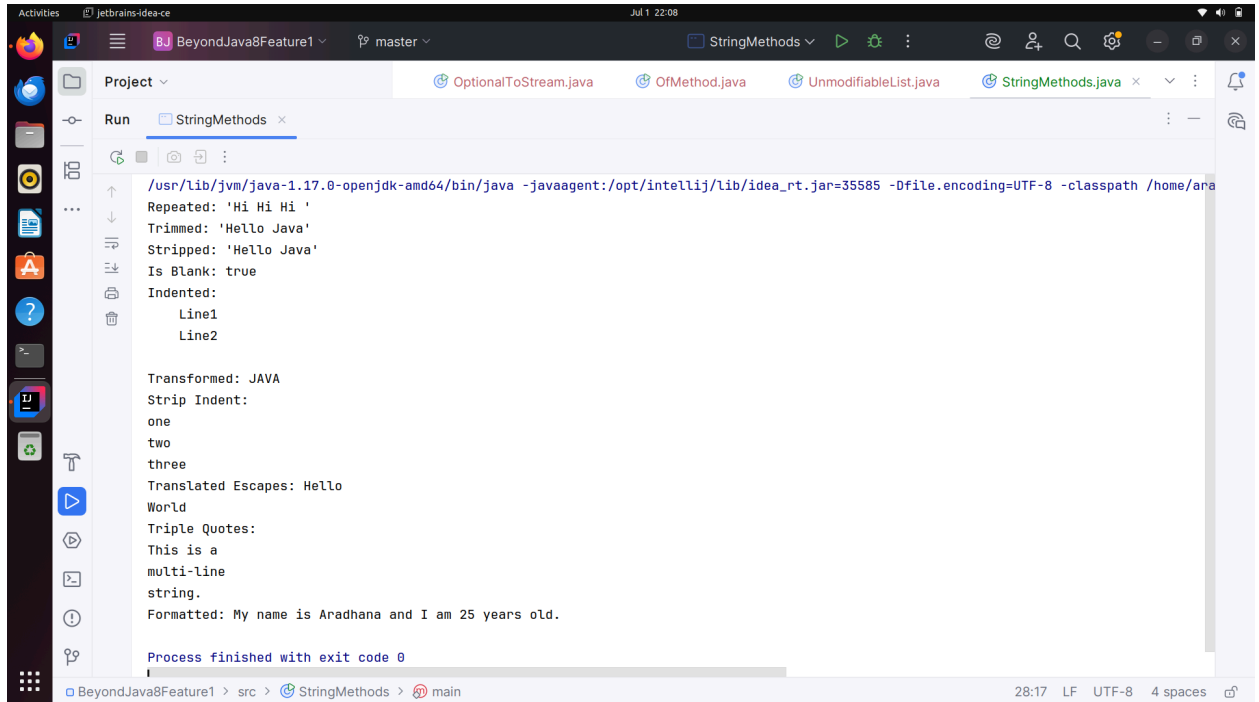


## 5. Demonstrate the use of repeat(), strip(), trim(), isBlank(), indent(), transform(), stripIndent(), translateEscapes(), tripleQuotes and formatted() methods.



```
System.out.println("Translated Escapes: " + escaped);
String tripleQuote = ""
    This is a
    multi-line
    string.""
System.out.println("Triple Quotes:\n" + tripleQuote);
String formatted = "My name is %s and I am %d years old."
    .formatted(...args: "Aradhana", 25);
System.out.println("Formatted: " + formatted);
}
```

Output:



```
Run StringMethods x
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=35585 -Dfile.encoding=UTF-8 -classpath /home/ara
Repeated: 'Hi Hi Hi '
Trimmed: 'Hello Java'
Stripped: 'Hello Java'
Is Blank: true
Indented:
    Line1
    Line2

Transformed: JAVA
Strip Indent:
one
two
three
Translated Escapes: Hello
World
Triple Quotes:
This is a
multi-line
string.
Formatted: My name is Aradhana and I am 25 years old.

Process finished with exit code 0
```

6. You are tasked with writing a processOrderStatus method that takes an OrderStatus enum as input and returns a descriptive string based on the order status. Here's the OrderStatus enum:

```
public enum OrderStatus {
    PENDING,
    PROCESSING,
    SHIPPED,
    DELIVERED,
    CANCELLED,
    REFUNDED
}
```

Your processOrderStatus method should adhere to the following rules: For PENDING orders, return: "Order is awaiting confirmation."

For PROCESSING orders, return: "Order is being prepared."

For SHIPPED orders, return: "Order has been dispatched."

For DELIVERED orders, return: "Order has been successfully delivered."

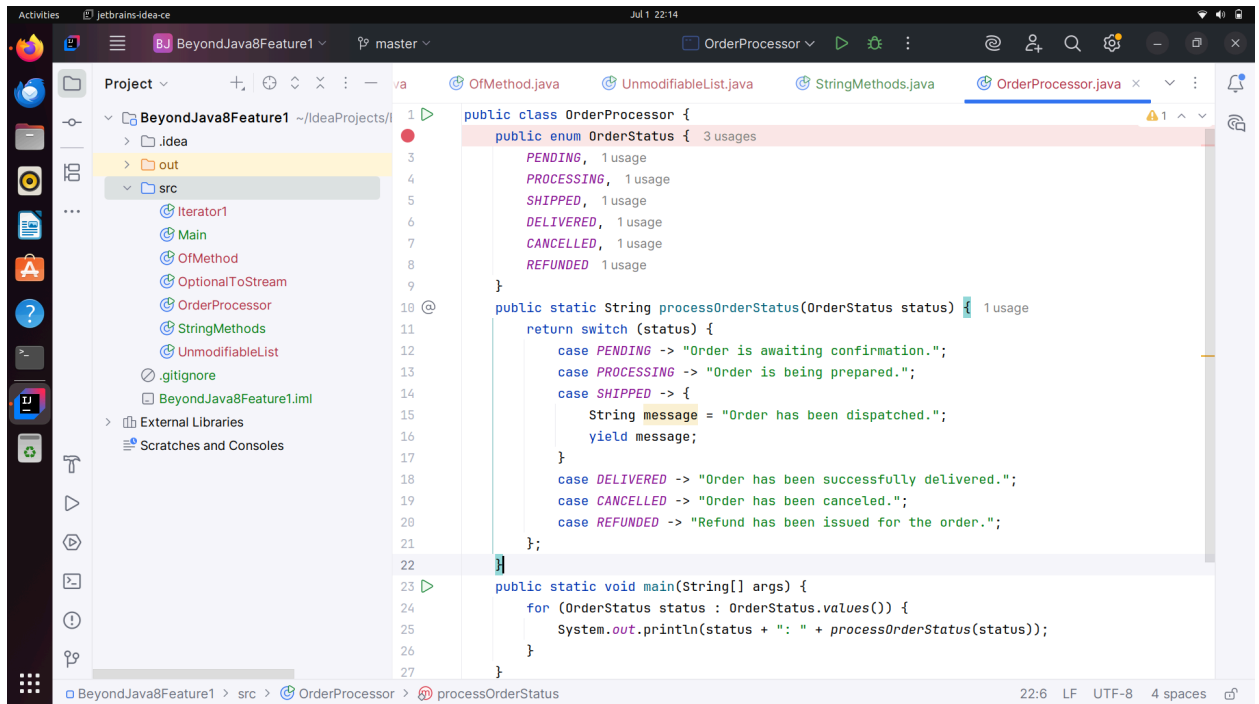
For CANCELLED orders, return: "Order has been canceled."

For REFUNDED orders, return: "Refund has been issued for the order."

Use a single switch expression to achieve this.

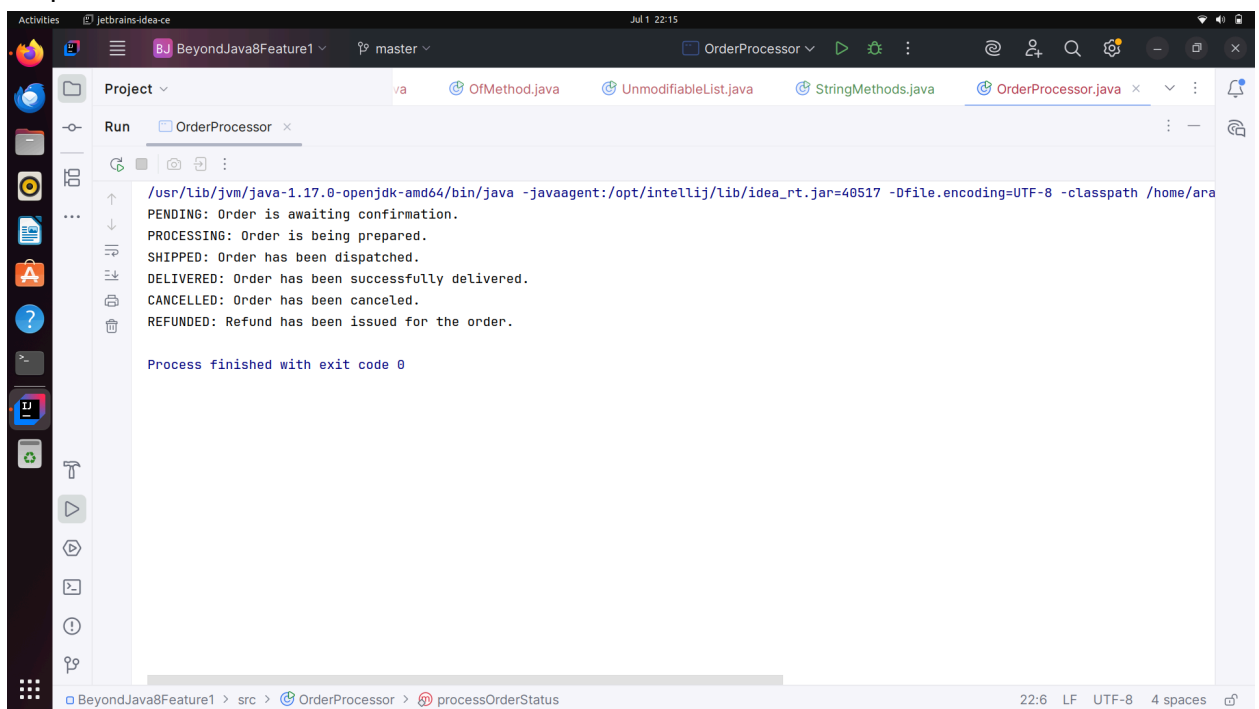
Enhancements:

yield Keyword: If your logic requires more complex processing within a case, demonstrate the use of the yield keyword to return a value from the switch expression.



```
1 public class OrderProcessor {
2     public enum OrderStatus { 3 usages
3         PENDING, 1 usage
4         PROCESSING, 1 usage
5         SHIPPED, 1 usage
6         DELIVERED, 1 usage
7         CANCELLED, 1 usage
8         REFUNDED 1 usage
9     }
10    @
11    public static String processOrderStatus(OrderStatus status) { 1 usage
12        return switch (status) {
13            case PENDING -> "Order is awaiting confirmation.";
14            case PROCESSING -> "Order is being prepared.";
15            case SHIPPED -> {
16                String message = "Order has been dispatched.";
17                yield message;
18            }
19            case DELIVERED -> "Order has been successfully delivered.";
20            case CANCELLED -> "Order has been canceled.";
21            case REFUNDED -> "Refund has been issued for the order.";
22        };
23    }
24    public static void main(String[] args) {
25        for (OrderStatus status : OrderStatus.values()) {
26            System.out.println(status + ": " + processOrderStatus(status));
27        }
28    }
29 }
```

## Output:



```
Run OrderProcessor x
/usr/lib/jvm/java-1.17.0-openjdk-amd64/bin/java -javaagent:/opt/intellij/lib/idea_rt.jar=48517 -Dfile.encoding=UTF-8 -classpath /home/ara
PENDING: Order is awaiting confirmation.
PROCESSING: Order is being prepared.
SHIPPED: Order has been dispatched.
DELIVERED: Order has been successfully delivered.
CANCELLED: Order has been canceled.
REFUNDED: Refund has been issued for the order.

Process finished with exit code 0
```