Predicate not() method:

A static not() method has been added to the Predicate interface in Java 11. As the name suggests, this method is used to negate a Predicate. The not() method can also be used with method references.

Code:

```
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;
import java.util.stream.Collectors;

public class PredicateNot {
 public static void main(String[] args) {
 Predicate<String> startWithZ = s -> s.charAt(0) == 'z';
 Predicate<String> doesNotStartWithZ = Predicate.not(startWithZ);

 List<String> list = Arrays.asList("za", "zq", "az", "aq", "zz");
 List<String> strsStartingWithZ = list.stream()
 .filter(startWithZ)
 .collect(Collectors.toList());
 List<String> strsNotStartingWithZ = list.stream()
 .filter(doesNotStartWithZ)
 .collect(Collectors.toList());

 System.out.println(strsStartingWithZ);
 System.out.println(strsNotStartingWithZ);
 }
}
```

Output:

```
[za, zq, zz]

[az, aq]
```

Collection to an Array:

The new default toArray() method is used to easily convert a collection to an array of the correct type.

Code:

```
import java.util.List;

public class CollToArray {
 public static void main(String[] args) {
 List<String> list = List.of("foo", "bar", "baz");

String[] strings2a = list.toArray(new String[list.size()]);
 String[] strings2b = list.toArray(new String[0]);

 for (String s : strings2a) {
 System.out.println("Full Size " + s);
 }

 for (String s : strings2b) {
 System.out.println("Single String " + s);
 }
 }
}
```

Output:

```
Full Size foo

Full Size bar

Full Size baz

Single String foo

Single String bar

Single String baz
```

New File Methods:

Java 11 makes it a lot easier to read and write strings. The readString() and writeString() static methods are added to the Files class for this purpose.

Code:

```
import java.io.IOException;
import java.net.URI;
import java.nio.file.Files;
import java.nio.file.Path;

public class FilesReadWrite {
 public static void main(String[] args) throws IOException {
 Path path = Files.createTempFile("testFile", ".txt");
 System.out.println(path);

 // Not Mandatory
 String tempFolder = System.getProperty("java.io.tmpdir");
 String uriFullPath = "file:///" + tempFolder.replace("\\", "/") + "test.txt";
 URI uri = URI.create(uriFullPath);
 System.out.println(uri);
 // Create Path from a URI
 Path path2 = Path.of(uri);

 //Writing to the file
 Files.writeString(path, "Hello World");
 //Reading from the file
 String s = Files.readString(path);
 System.out.print(s);
 }
}
```

Output:

```
C:\Users\Wissen\AppData\Local\Temp\testFile4119949823874205369.txt

file:///C:/Users/Wissen/AppData/Local/Temp/test.txt

Hello World
```

New String Methods:

Java 11 introduced a few new methods to the String class. The following list explains these new methods.

a.) isBlank() - this method is used to check whether a string is blank or not. Empty strings and strings with just whitespace are considered blank.

b.) lines() - this method splits a string using line terminators and returns a stream.

c.) repeat() - this method is used to duplicate or repeat a string.

d.) strip(), stripLeading(), stripTrailing() - These methods are used to remove whitespace from the strings. They are very similar to the existing trim() method, but provide Unicode support.

Code:

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

public class StringNewMethods {
 public static void main(String[] args) {
 System.out.println(" ".isBlank()); //true

 String s = "Anupam";
 System.out.println(s.isBlank()); //false
 String s1 = "";
 System.out.println(s1.isBlank()); //true

 String str = "A\nB\nC";
 System.out.println(str);
 System.out.println(str.lines().collect(Collectors.toList()));

 String str2 = " Do ";
 System.out.print("Start");
 System.out.print(str2.strip() );
 System.out.println("End");

 System.out.print("Start");
 System.out.print(str2.stripLeading());
 System.out.println("End");

 System.out.print("Start");
 System.out.print(str2.stripTrailing());
 System.out.println("End");

 String str3 = "=".repeat(2);
 System.out.println(str3); //prints ==
 }
}
```

Output:

```
true

false

true

A

B

C

[A, B, C]

StartDoEnd

StartDo End

Start DoEnd

==
```

Optional.isEmpty():

Java 11 introduced new method to Optional class as isEmpty() to check if value is present. isEmpty() returns false if value is present otherwise true.

It can be used as an alternative of isPresent() method which often needs to negate to check if value is not present.

Code:

```
import java.util.Optional;

class OptionalIsEmpty {
 public static void main(String[] args) {
 String test = "hello-educative";
 Optional<String> stringOptional = Optional.ofNullable(test);
 System.out.println("The isEmpty() method returns " + stringOptional.isEmpty() +
" when the Optional object has a string value");

 test = "";
 stringOptional = Optional.ofNullable(test);
 System.out.println("The isEmpty() method returns " + stringOptional.isEmpty() +
" when the Optional object has an empty string value");

 test = null;
 stringOptional = Optional.ofNullable(test);
 System.out.println("The isEmpty() method returns " + stringOptional.isEmpty() +
" when the Optional object has no value");

 String name = null;
 System.out.println(!Optional.ofNullable(name).isPresent());
 System.out.println(Optional.ofNullable(name).isEmpty());

 name = "Joe";
 System.out.println(!Optional.ofNullable(name).isPresent());
 System.out.println(Optional.ofNullable(name).isEmpty());
 }
}
```

Output:

```
The isEmpty() method returns false when the Optional object has a string value
The isEmpty() method returns false when the Optional object has an empty string value
The isEmpty() method returns true when the Optional object has no value
true
true
false
false
```