Lambda Expressions

By Hatef Palizgar

Task 1

Write the following anonymous class as a lambda expression:

```
Runnable runnable = new Runnable() {
   @Override
   public void run() {
       String myString = "Let's split this up into an array";
       String[] parts = myString.split(" ");
       for (String part : parts) {
           System.out.println(part);
       }
   }
 };
```

Task 2

Write the following method as a lambda expression and store it in a variable `Function<String, String> lambdaFunction`.

This method prints ***every second character*** in the string.

For example: "1234567890" -> "24680"

```
public static String everySecondChar(String source) {
  StringBuilder returnVal = new StringBuilder();
  for (int i = 0; i < s.length(); i++) {
    if (i % 2 == 1) {
      returnVal.append(s.charAt(i));
    }
  }

  return returnVal.toString();
}
```

Task 3

The lambda expression you wrote on **Task 2** doesn't do anything. Write a single line of code that will execute it with an argument "1234567890" and print it out to the console.

Task 4

Instead of executing the function of **Task 3** directly, suppose we want to pass it to a method.

Write a method called `everySecondCharacter` that accepts the function as a parameter and executes it with the argument "1234567890".

Below is how `everySecondCharacter` method will be used:

```
String result = everySecondCharacter(lambdaFunction);
System.out.println(result);
```

Task 5

Write a lambda expression that can be stored as `java.util.Supplier` interface.

This lambda should return the string "I love Java!"

Assign it to a variable called `iLoveJava`

Task 6

As with the Function you wrote for **Task 3**, the Supplier for **Task 5** won't do anything until we use it. Use this supplier to assign the string "I love Java!" to a variable called `supplierResult`.

Then print the variable to the console.

Task 7

There are many interfaces in Java SDK, and sometimes we can use a lambda expression instead of creating an instance that implements the interface we want to use.

Question 1:

Given a specific interface, how can we tell whether we can map a lambda expression to it or not?

What's the criteria that has to be met?

Question 2:

With that in mind, can we use a lambda expression to represent an instance of `java.util.concurrent.Callable`[1] interface?

[1] The `callable` documentation can be found here:
https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/Callable.html

Question 3:

Is `java.util.Comparator`[2] interface a functional interface?

Task 8

Suppose we have the following list of the top 5 male and female names:

```
List<String> topNames = Arrays.asList(
                "Amelia",
                "Olivia",
                "emily",
                "Isla",
                "Ava",
                "oliver",
                "Jack",
                "Charlie",
                "harry",
                "Jacob"
        );
```

Write a code to print the items in the list in sorted order, and with the first letter in each name upper-cased.

The name 'harry' should be printed as 'Harry' and should be printed after 'Emily' and before 'Isla'.

Use Lambda expressions wherever possible.

Task 9

Change the code you have written for **Task 8** such that it uses method references.

Remember that a method reference looks like `Class::methodName`

[2] The `Comparator` documentation can be found here: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Comparator.html