Purpose	Practice using Lambda Expressions and Streams
What to Submit	Commit a revised coinpurse.MoneyUtil with test code to your Purse project .

(not complete yet)

Streams

We can create a Stream from any collection by calling stream(). The main Stream methods are described in the PDF in the week14 folder. Here is an example that filters Strings have length less than or equal to 4.

```
List<String> words =
     Arrays.asList("Dog", "elephant", "Bird", "Zebra", "snake");
// Define a Predicate that is true fo the objects we want:
Predicate<String> shortWords = (s) -> (s.length() <= 4);
// Print the whole list
words.stream().forEach( System.out::println );
// Filter and print sublist
words.stream().filter( shortWords ).forEach(System.out::println);
```

In the above example we *consume* the stream using forEach. You can also *collect* the stream into an object using a Collector. The Java Collectors class provides many useful predefined collectors, such as Collectors.asList(). Collectiors.asList() puts the stream into a List and returns it.

```
// Create a list of words with length <= 4
List<String> result =
words.stream().filter( shortWords ).collect( Colletors.asList() );
```

X. Modify the MoneyUtil.filterByCurrency method to use a stream and a Predicate as filter. You can write Predicate as a lambda or anonymous class. When you finish, the method should not have any loops. It should have 3 statements: 1) check that currency param is not null, 2) define a Predicate as filter, 3) use a Stream to filter the list and return the result.

Java 8 New Interfaces and Lambdas

Java 8 has several new interfaces in the package java.util.function. They are called *functional interfaces* because they have only one abstract method. Most of them are special cases of one of these interfaces:

Interface	Abstract Method	Purpose
`Consumer<T>`	`void accept(T t)`	A function of one variable that doesn't return anything.
`Function<T, R>`	`R apply(T t)`	A function of one variable that returns a result. The result may be a different type.
`BiFunction<T,U,R>`	`R apply(T a, U b)`	A function of two variables that returns a result.
`Predicate<T>`	`boolean test(T t)`	Perform a test on the argument and return true or false.
`Supplier<T>`	`T get( )`	"Supplies" objects of type T, one object per call.