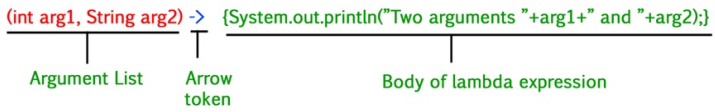
----------day 2.22
What are the new features released in Java 8?
359.What are the main benefits of new features introduced in Java 8?

- forEach() method in Iterable interface.
- default and static methods in Interfaces.
- Functional Interfaces and Lambda Expressions.
- Java Stream API.
- Java Time API.
- Collection API improvements.
- Concurrency API improvements.
- Java IO improvements.

360.What is a Lambda expression in Java 8?

```
(parameter1, parameter2) -> expression
(parameter1, parameter2) -> { code block }
```

361.What are the three main parts of a Lambda expression in Java?
three parts: **a parenthesized set of parameters, an arrow, and then a body**



362.What is the data type of a Lambda expression?
Interface
363.What is the meaning of following lambda expression?
A lambda expression is **a short block of code which takes in parameters and returns a value**. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

364.Why did Oracle release a new version of Java like Java 8?
releases provide only stability, security, and performance improvements—not new features.
365.What are the advantages of a lambda expression?
Lambda expressions **improve code readability and do not require interpretation**. Lambdas allow you to write concise code.
366.What is a Functional interface in Java 8?
A functional interface is **an interface that contains only one abstract method**. lambda expressions can be used to represent the **instance of a functional interface**. A functional interface can have any number of default methods.
367. What is a Single Abstract Method (SAM) interface in Java 8?
An interface having only one abstract method is known as a functional interface and named as Single Abstract Method Interfaces (SAM Interfaces). One abstract method means that either a default method or an abstract method whose implementation is available by default is allowed.
368.How can we define a Functional interface in Java 8?

Functional Interface in Java is also called **Single Abstract Method (SAM) interface**. From Java 8 onwards, to represent the instance of functional interfaces, lambda expressions are used. Lambda expressions are anonymous (don't have names) shortcode blocks that can take parameters and return a value just like methods

369.Why do we need Functional interface in Java?
SE 8 with Lambda expressions and Method references in order **to make code more readable, clean, and straightforward**. Functional interfaces are interfaces that ensure that they include precisely only one abstract method.
370.Is it mandatory to use @FunctionalInterface annotation to define a Functional interface in Java 8?

It's not mandatory to mark the functional interface with @FunctionalInterface annotation, the compiler doesn't throw any error. But it's good practice to use.

371.What are the differences between Collection and Stream API in Java 8?
Collection----group of objs
Stream API--- help to convert collection to stream of obj and process individual item in create stream form collection
A stream does not store data. An operation on a stream does not modify its source, but simply produces a result. Collections have a finite size, but streams do not
372.What are the main uses of Stream API in Java 8?
process a sequence of elements, the **Stream API** can help to make code more concise, as well as increase its readability similar to SQL statements.
373.What are the differences between Intermediate and Terminal

Intermediate operations are those operations that return *Stream* itself, allowing for further operations on a stream.---intermediate operations are *filter*, *map*.

Operations in Java 8 Streams?

374.What is a Spliterator in Java 8?

like Iterator s, **are for traversing the elements of a source.**

375.What are the differences between Iterator and Spliterator in Java 8?

Iterator performs only iteration over a set of elements.

Spliterator splits as well as iterates over a set of elements --- `use to` parallel processing of elements.

An Iterator is a simple representation of a series of elements that can be iterated over. A Spliterator can be used to split given element set into multiple sets so that we can perform some kind of operations/calculations on each set in different threads independently, possibly taking advantage of parallelism.