

A background image of four students in a library setting. A young man in a grey t-shirt is smiling and looking towards the right. A young woman with glasses and a dark polka-dot top is looking at a laptop. Another young woman is partially visible on the right, looking at the laptop. A young man is on the left, looking towards the center. They are all sitting at a table with books and a laptop. The background is filled with bookshelves.

Working with Streams and Lambda Expressions

Method References

Method References

- Lambda expressions help in making your code more concise (fewer keystrokes).
- Method references, can, in certain situations, help in making your lambda expressions even more concise.
- If all your lambda expression does is call one method, then that is an opportunity to use a method reference.
- If a lambda parameter is simply passed to another method, then the redundancy of specifying the variable twice can be removed.



Interface

Consumer<T>

Functional method

void accept (T t)

```
List<String> names = Arrays.asList("Sean", "Mary", "John");  
names.forEach(name -> System.out.println(name)); // lambda  
names.forEach(System.out::println); // method reference
```

Method References

- There are four different styles/types:
 1. Bound – instance known at compile-time
 2. Unbound – instance provided at runtime
 3. Static
 4. Constructor
- With method references, **context** is key!
 - the functional interface type being created is hugely important in determining context.
- The compiler turns your method references into lambdas in the background.

