In this course, you will learn how to use Lambda expressions and the Streams API to program in a more functional style using JDK 8. This will enable you to solve common problems in a more concise and more flexible way that can take advantage of multiple cores and CPUs in your machine.
Lesson 1: Lambda Expressions
Simon Ritter
Java Technology Evangelist
Lesson Agenda

- Why does Java need Lambda expressions
- Lambda expression syntax
- Functional interfaces and their definition
- Functional Interfaces in the java.util.function package
- Method and constructor references
- Referencing external variables in Lambdas
- Useful new methods in JDK 8 that can use Lambdas
Lesson 1-1: Why Does Java Need Lambda Expressions?
Concurrency in Java

java.lang.Thread (jsr166)

java.util.concurrent (jsr166)

Phasers, etc (jsr166)

Fork/Join Framework (jsr166y)

Project Lambda

1.0
5.0
6
7
8

The Problem: External Iteration

List<Student> students = ...
double highestScore = 0.0;

for (Student s : students) {
    if (s.getGradYear() == 2011) {
        if (s.getScore() > highestScore)
            highestScore = s.getScore();
    }
}

- Our code controls iteration
- *Inherently serial:* iterate from beginning to end
- Not thread-safe
  - Business logic is stateful
  - Mutable accumulator variable
Internal Iteration With Inner Classes

More Functional

double highestScore = students
  .filter(new Predicate<Student>() {
    public boolean op(Student s) {
      return s.getGradYear() == 2011;
    }
  })
  .map(new Mapper<Student,Double>() {
    public Double extract(Student s) {
      return s.getScore();
    }
  })
  .max();

- Iteration handled by the library
- Not inherently serial – traversal may be done in parallel
- Traversal may be done lazily – so one pass, rather than three
- Thread safe – client logic is stateless
- High barrier to use
  - Syntactically ugly
Internal Iteration With Lambda Expressions

List<Student> students = ...
double highestScore = students
    .filter(Student s -> s.getGradYear() == 2011)
    .map(Student s -> s.getScore())
    .max();

- More readable
- More abstract
- Less error-prone

This slide is intended to be conceptual. A little more work is needed to get this code to compile.
Section 1

Summary

- Need changes to Java to simplify parallel coding
- Lambda expressions simplify how to pass behaviour as a parameter