Functional Programming in Scala

Graeme Ludwig

Code samples available at:

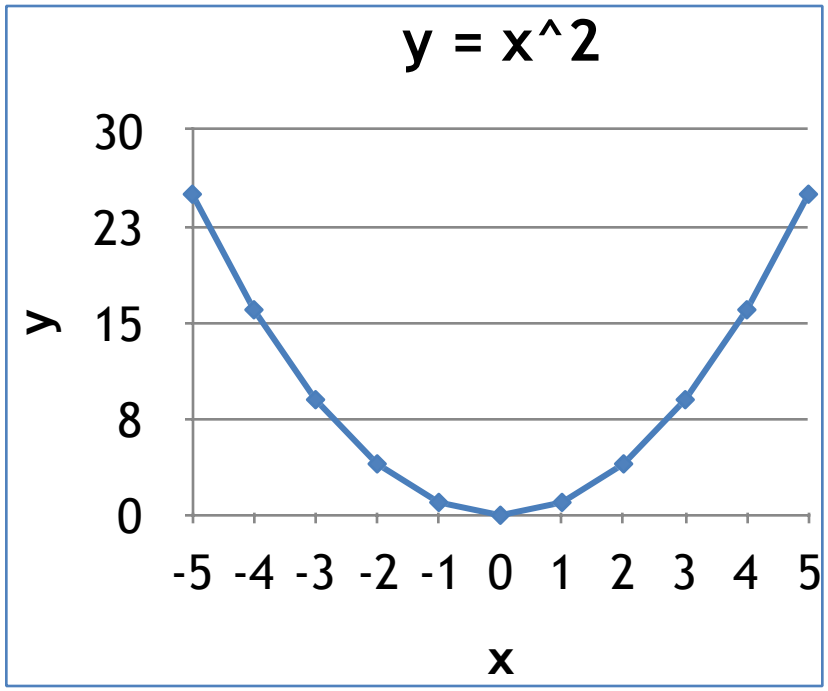
https://github.com/ludwiggj/devCon7

Code presented using Scala IDE 3.0.2-final (http://scala-ide.org/)

Functional Programming

- A style of programming
- Compute things by evaluating expressions
- Compose functions to create more useful (powerful) functions

Functional Programming



- y depends on x ONLY
- There is NO state
- There is NO mutable data
- There are NO side effects
- Same input gives same output
  e.g. 3 * 3 is always 9

Programs composed of such functions are easier to understand and predict.

Scala

- Arrived in 2003
- A hybrid language…
  - Functional
  - Object-oriented
  - Supports imperative & declarative styles
  - Feature-rich; concurrency, lazy evaluation etc.
- Runs on the JVM
- Interworks with Java libraries
- A replacement for Java?

Java 8

- Due in 2014. Biggest update since Java 5.
- Java embraces functional programming to support multi-core architectures.
- Lambdas… functions as 1st class citizens.
- Streams and parallel streams.

Increasing the “Beef to Bun” ratio

```
Collection<Person> people = …;
Collection<Person> electoralRoll = people;

Iterator<Person> ip = electoralRoll.iterator();

while (ip.hasNext()) {
      Person p = ip.next();
      if (p.getAge() < 18) {
          ip.remove();
      }
}
```

More beef…

```
Collection<Person> people = …;
Collection<Person> electoralRoll = people;

Collections.removeAll(
    electoralRoll,
    new Predicate<Person>() {
        public boolean test(Person p) {
            return p.getAge() < 18;
        }
    });
```

Extra beef with lamb(das)

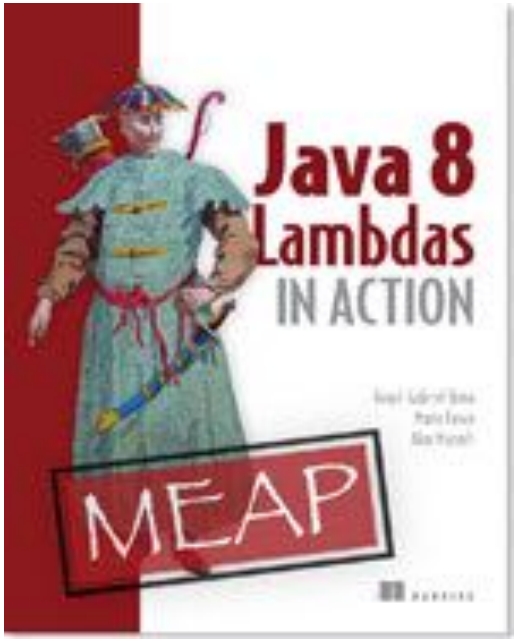
```
Collection<Person> people = …;
Collection<Person> electoralRoll = people;


Collections.removeAll(
    electoralRoll,
    p -> p.getAge() < 18
);
```

Streams

- Similar to collections.
- Collection is in memory. Holds all values. Iterated externally.
- Stream is conceptually fixed. Lazily evaluated. Iterated internally.
- Can be parallelised (ParallelStream) e.g. UK electoral roll.
- BUT methods passed in must not interact with each other e.g. mutable shared objects representing house occupancy rules.

“Java 8 Lambdas In Action”



Now in MEAP (Manning Early Access Program).

First chapter available for free.

See http://www.manning.com/urma/