Modern Web Development for Java Programmers

Unit 6. Overview of the new features of Java SE 8.
Applying bulk operations on the collections.
Java build automation with Gradle.
Test-Driven Development in Java.

Unit 6 Timeline

- Java SE 8. Default methods — 15 min
    - Walkthrough 1 — 5 min
- Java SE 8. Lambdas — 10 min
    - Walkthrough 2 — 10 min
- Java SE 8. Streams API — 10 min
    - Walkthrough 3 — 20 min
- Break — 10 min
- Build Automation with Gradle — 20 min
    - Walkthrough 4 — 25 min
- Break — 5 min
- Unit testing with Spock framework — 30 min
    - Walkthrough 5 — 20 min

Java SE 8. Selected New Features

Can you modify an interface?

1. Yes, but adding/changing method declarations will break the existing classes that implement this interface.
2. Yes, you can and no changes in the existing classes that implement this interface is required.

`Iterable` Interface in Java 7 and 8

- Java 7 `Iterable` declares one method: `iterator()`
  http://bit.ly/1mgVzxl

- Java 8 `Iterable` declares three methods: `iterator()`, `spliterator()`, and `forEach()`
  http://bit.ly/1ekrDMc

`Iterable` Interface in Java 8

All Methods | **Instance Methods** | **Abstract Methods** | **Default Methods**

Modifier and Type	Method and Description
`default void`	**`forEach`**`(`**`Consumer`**`<? super T> action)` Performs the given action for each element of the `Iterable` until all elements have been processed or the action throws an exception.
`Iterator``<T>`	**`iterator`**`()` Returns an iterator over elements of type T.
`default` **`Spliterator`**`<T>`	**`spliterator`**`()` Creates a **`Spliterator`** over the elements described by this `Iterable`.

Default methods also know as *defender* methods.

Default methods introduce multiple inheritance of behavior, but not multiple inheritance of state.

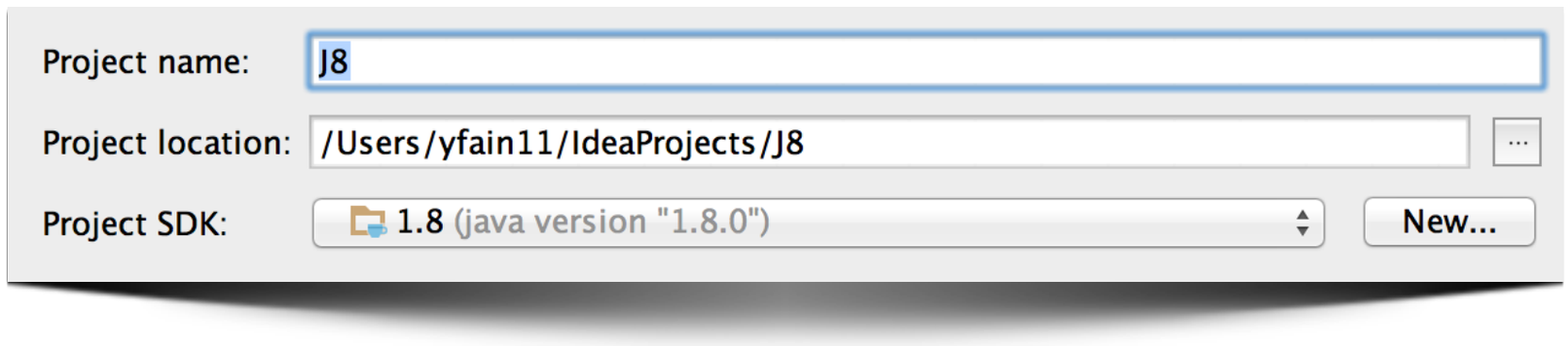
FARATA

Walkthrough 1(start)

- Prerequisite: you should have JDK 8 downloaded from http://bit.ly/1iOZIrD and installed.

- Create a new Java project named J8 in IDEA selecting Java 8 as Project SDK. On In Mac OS Java is installed in the folder /Library/Java/JavaVirtualMachines/jdk1.8.0.jdk. On Windows it's in c:\Program Files\Java\jdk1.8.0 unless you changed it.

- Select the language level support to Java 8 in the Project Structure | Modules | Sources.



Walkthrough 1(end)

- Create Payable interface with default method:

```
public interface Payable {
   default void increasePay(){
      System.out.println("Ain't no need your increases");
   }
}
```

- Create an empty class Person its subclass Employee:

```
public class Employee extends Person implements Payable {
    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.increasePay();  // using default increasePay() implementation
    }
 }
```

- Run the Employee program as is - the default method works.

- Add the the following increasePay() to Person and re-run:

```
public void increasePay(){
   System.out.println("Thank you boss!");
}
```

- Change the return value to boolean in Person's increasePay()…

Lambdas

- Lambda is an anonymous function that represents a single-method interface in a concise way.
- Similarly to methods, functions represent a behavior.
- With lambdas you can *pass behavior* via method arguments. For example, lambda is given to collection's forEach() method as argument.
- Anonymous classes allow implement methods from a base class without giving it a name. Lambda expressions allow to implement a single-function interface without even using a class.

Lambda Syntax

ArgumenList -> Body

- Body is either a single expression or a statement block
- If body is a single expression, its result is returned:
  `(int a, int b) -> a+b;`
- If lambda represents a method from an interface, which returns a type, use a `return` statement.
- Lambda expressions are objects. You can store lambda expression in a variable of type `Predicate` for reusability.
- Lambda expression does not create a new scope. Can't declare in lambda expression that has the same name as in enclosing scope.

Functional Interfaces

```
new Thread(new Runnable() {
  public void run() {
    System.out.println("Do something in this thread ");
  }
});
```

```
public interface ActionListener extends EventListener {

    /**
     * Invoked when an action occurs.
     */
    public void actionPerformed(ActionEvent e);

}
```

- Functional interface is an interface with a single abstract method
- Optional annotation `@FunctionalInterface`
- There's a bunch of new functional interfaces in the package `java.util.function`

Method References ::

You can put either a lambda expression of a method reference in place of a functional interface.

```
beer -> System.out.println(beer) // lambda
```

But if you're processing object with inferred type `beer`, you can just simply replace the above with this:

```
System.out::println
```

If your lambda just passes a parameter to a method, you can use :: notation.

```
beers.stream().forEach(System.out::println);
```


Walkthrough 2

- Review the code of **`Lovable`** in the project J8 to see functional interface implementation with lambda expression.

- Add a declaration of a method **`showPassion()`** to **`Lovable`**. Why compiler complains? Try to make a **`showPassion()`** default method. Any complains?

- Hmm…But compiler won't complain if you replace **`showPassion()`** with **`public String toString();`**

- Review the code of **`Love.`** Run it. Love it?

- Modify the method **`showLove()`** in **`Lovable`** to return **`String`** value. Modify the `Love` class accordingly.

Streams API

- Stream is an abstraction that represents zero or more values. This is not a collection. It's rather a fancy iterator.
- External vs internal processing of collections.
- Streams support various operations like filter, map, reduce, find, sort, match.
- An operation on a stream produces the result, but doesn't change the original stream.
- Streams can be chained into a pipeline because *each intermediate operation returns a stream.*

The javadoc of Streams API is here:
java.util.stream

Collections vs Streams

- A collection is a fixed data structure, where every element was evaluated before placing into a collection (*eager* evaluation).

- A stream is *lazily* constructed on the fly when by the user's request, e.g show me inexpensive American beers.

A pipeline consists of

- The data source turned into a stream
- Zero or more intermediate operations (each returns a reference to the stream).
- A terminal operation (ends the stream processing)

Pass lambda expressions to these operations to tell them what to do with the stream.

Streaming beer delivery

You can ask a pipeline to be executed in parallel.

Don’t tell this girl to bring 10 beers one at a time (external operations). Just ask for 10 beers (internal operations).

1. Pour the beer from 10 dispensers (a.k.a. fork)
2. Grab all 10 glasses (a.k.a. join) and run to the table

FARATA

Intermediate Operations compared to SQL

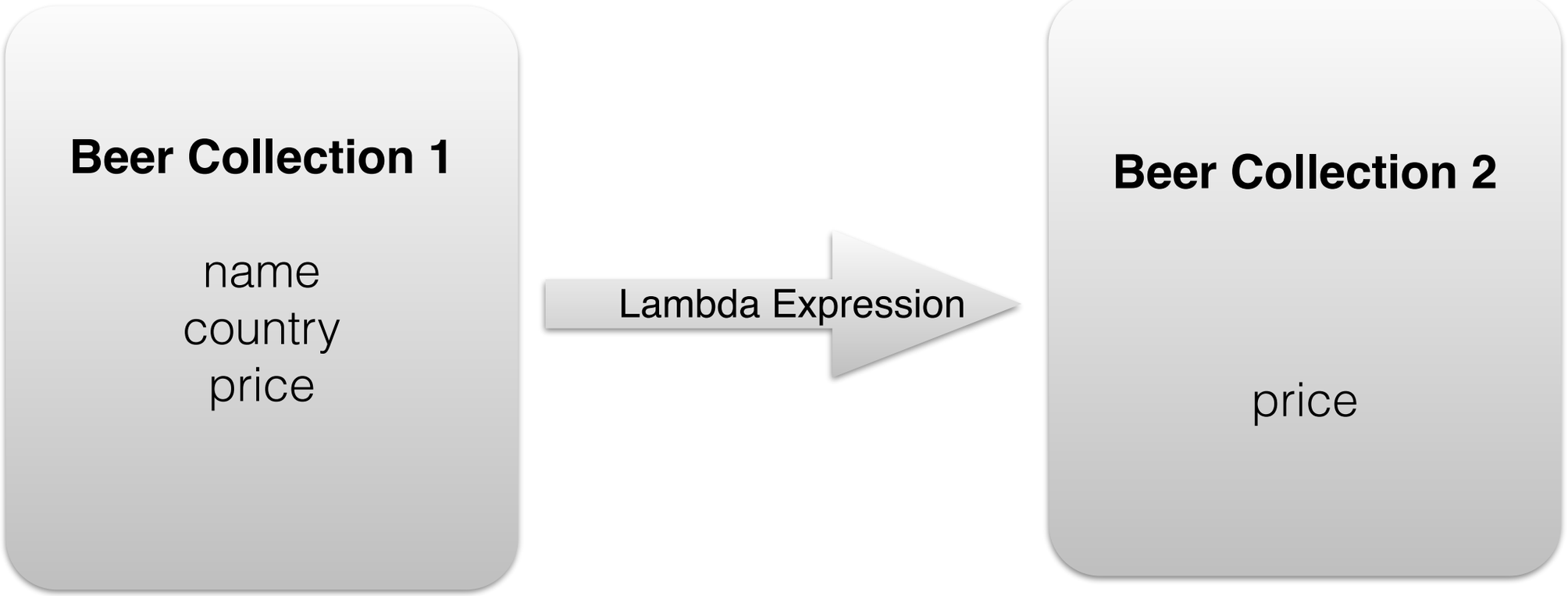
- filter() - select the object that meet certain criteria,
  e.g. using the `where` clause in SQL).
  The size of resulting collection can be smaller than original.

- map() - select only a subset of properties of the objects,
  e.g select the column list in SQL.
  The size of resulting collection is the same as original.

- reduce() - aggregate the data,
  e.g. select `count(*)` or select `sum(price)` in SQL

The class DoubleSummaryStatistics has a bunch of methods to collect statistics on streams (min, max, count, sum, and average)

The map() Method



Walkthrough 3 (start)

- Run the StreamsAndBeer class. It'll filter the Belgium beers without stream (external processing of a collection).

- What's the difference between
  `if ("Belgium".equals(beer.country))` and
  `if (beer.country.equals("Belgium"))`?

- Using Shift-Alt-Down move the beginning of block comment from line 50 to line 56. Re-run. Observe the use of the new method `forEach()`.

- Move the beginning of block comment from line 56 to 66. Re-run. Observe the use the lambda with return.

Walkthrough 3 (end)

- Move the beginning of block comment from line 66 to 74. Re-run. We've been adding "Octoberfest Special" only to `preferredBeers` collection, but why the `beers` collection has been modified too?

- Move the beginning of block comment from line 74 to 84. Re-run. Observe the use of `stream()`, `filter()`, and `collect()`.

- Move the beginning of block comment from line 84 to 94. Re-run.
  Observe that collection is sorted.

- In the sorting block replace the stream() with parallelStream(). Re-run.
  Sorting broken. Replace forEach() with forEachOrdered(). Re-run. Sorting fixed, but parallel execution might be lost.

- Move the beginning of block comment from line 94 to 105.
  Observe the use of `mapToDouble()` and `average()`. Replace the `stream()` invocation with `parallelStream()`. Re-run. The output is the same, but the processing was done in parallel.

- Move the beginning of block comment from line 105 to 118. Re-run.
  Note the use of the Predicate<Beer> to store a lambda expression in the variable madeInUSA.

Additional Reading

- Lambda Quick Start: http://bit.ly/1eR0we0
- Streams package summary: http://bit.ly/1eh4aZo
- JDK8 Tutorials: http://bit.ly/OALLA4
- Parallel Streams http://bit.ly/1kyElaK
- New Date Time API: https://www.youtube.com/watch?v=OIg9lNpMJew
- Support of Java 8 in Eclipse Kepler: http://www.eclipse.org/downloads/java8/

Gradle

Build automation for Java platform

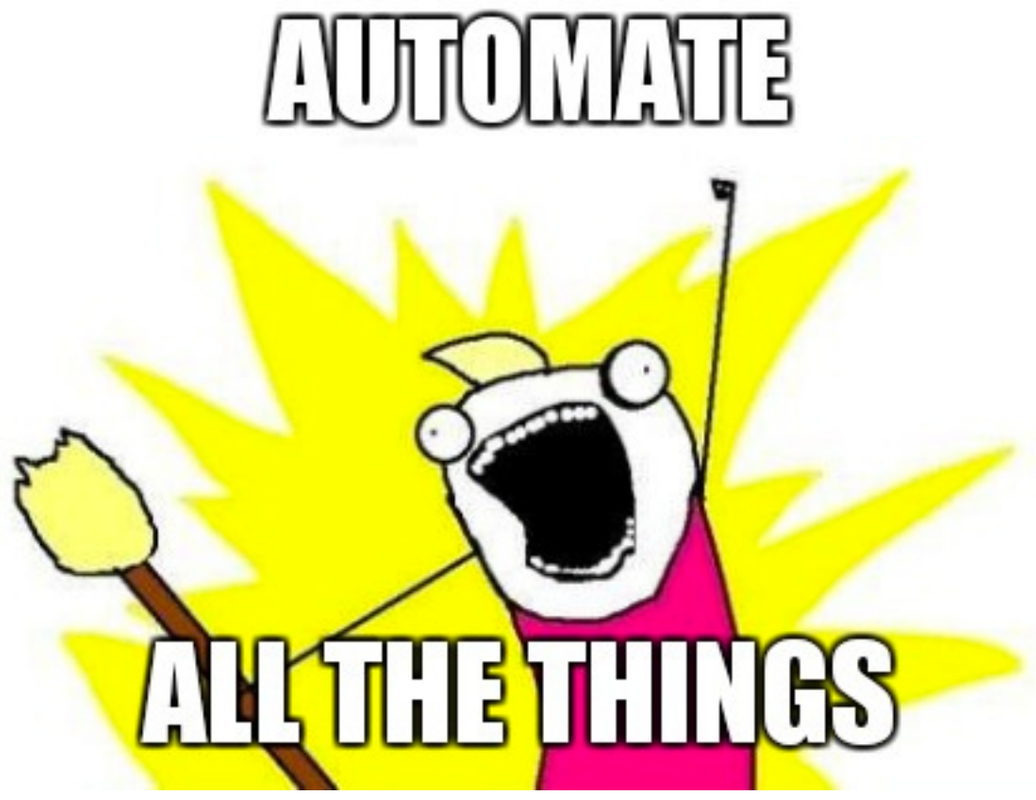
Why to automate you build?

Why to automate you build?

- Be lazy and don’t repeat yourself (DRY)

Why to automate you build?

- Be lazy and don’t repeat yourself (DRY)
- Automate all the things


AUTOMATE
ALL THE THINGS

Why to automate you build?

- Be lazy and don’t repeat yourself (DRY)
- Automate all the things
- Build in IDE doesn’t work
- Remember the “Hit by a Bus” Rule

Why Gradle?

Why Gradle?

- Ruby uses build scripts written in Ruby — Rake

Why Gradle?

- Ruby uses build scripts written in Ruby — Rake
- Scala uses build scripts written in Scala — SBT

Why Gradle?

- Ruby uses build scripts written in Ruby — Rake
- Scala uses build scripts written in Scala — SBT
- Java uses build scripts written in XML (???) — Maven, Ant

Gradle

- Written in Java
- Uses Groovy DLS
- Supported by major IDEs
- «Dream team» is behind it

Gradle vs Ant

Gradle vs Ant

```xml
<project>

    <target name="clean">
        <delete dir="build"/>
    </target>

    <target name="compile">
        <mkdir dir="build/classes"/>
        <javac srcdir="src/main/java" destdir="build/classes"/>
    </target>

    <target name="jar" depends="compile">
        <mkdir dir="build/libs"/>
        <jar destfile="build/libs/gradle_spock_ant.jar" basedir="build/classes">
            <manifest>
                <attribute name="Main-Class"
                           value="com.farata.course.mwd.java8.SwingActionListener"/>
            </manifest>
        </jar>
    </target>

    <target name="run" depends="jar">
        <java jar="build/libs/gradle_spock_ant.jar" fork="true"/>
    </target>

    <target name="build" depends="compile, jar"/>

</project>
```


FARATA

Gradle vs Ant

Gradle vs Ant

```
apply plugin: 'java'
apply plugin: 'application'

mainClassName = "com.farata.course.mwd.java8.SwingActionListener"
```


Gradle vs Maven

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.farata.course.mwd</groupId>
    <artifactId>gradle_spock</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>Example Maven pom</name>
    <url>https://github.com/yfain/WebDevForJavaProgrammers</url>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>1.2.1</version>
                <configuration>
                    <mainClass>com.farata.course.mwd.java8.SwingActionListener</mainClass>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.codehaus.gmaven</groupId>
                <artifactId>gmaven-plugin</artifactId>
                <version>1.5</version>
                <configuration>
                    <providerSelection>2.0</providerSelection>
                </configuration>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
                    </execution>
                </executions>
                <dependencies>
                    <dependency>
                        <groupId>org.codehaus.gmaven.runtime</groupId>
                        <artifactId>gmaven-runtime-2.0</artifactId>
                        <version>1.5</version>
                        <exclusions>
                            <exclusion>
                                <groupId>org.codehaus.groovy</groupId>
                                <artifactId>groovy-all</artifactId>
                            </exclusion>
                        </exclusions>
                    </dependency>
                    <dependency>
                        <groupId>org.codehaus.groovy</groupId>
                        <artifactId>groovy-all</artifactId>
                        <version>2.3.0-beta-1</version>
                    </dependency>
                </dependencies>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.spockframework</groupId>
            <artifactId>spock-core</artifactId>
            <version>0.7-groovy-2.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Gradle vs Maven

Gradle vs Maven

```
apply plugin: 'java'
apply plugin: 'groovy'

apply plugin: 'application'

group = "com.farata.course.mwd"

sourceCompatibility = 1.8
targetCompatibility = 1.8

mainClassName = "com.farata.course.mwd.java8.SwingActionListener"

repositories {
    jcenter()
    mavenCentral()
}

dependencies {
   testCompile 'org.codehaus.groovy:groovy-all:2.3.0-beta-1'
   testCompile "org.spockframework:spock-core:0.7-groovy-2.0"
   testRuntime 'com.h2database:h2:1.3.176'
   testCompile "junit:junit:4.11"
}
```


The unique Gradle's features

The unique Gradle's features

- Build scripts considerably smaller than Maven or Ant one. XML is more verbose than Gradle DSL

The unique Gradle's features

- Build scripts considerably smaller than Maven or Ant one. XML is more verbose than Gradle DSL

- Incremental builds

The unique Gradle's features

- Build scripts considerably smaller than Maven or Ant one. XML is more verbose than Gradle DSL

- Incremental builds

- The Gradle Daemon (or build daemon)

The unique Gradle's features

- Build scripts considerably smaller than Maven or Ant one. XML is more verbose than Gradle DSL

- Incremental builds

- The Gradle Daemon (or build daemon)

- The Gradle Wrapper

The unique Gradle's features

- Build scripts considerably smaller than Maven or Ant one. XML is more verbose than Gradle DSL
- Incremental builds
- The Gradle Daemon (or build daemon)
- The Gradle Wrapper
- Independent tasks can be run in parallel

The unique Gradle's features

- Build scripts considerably smaller than Maven or Ant one. XML is more verbose than Gradle DSL
- Incremental builds
- The Gradle Daemon (or build daemon)
- The Gradle Wrapper
- Independent tasks can be run in parallel
- IDE project generation

They’re using Gradle

They're using Gradle

- Gradle, Groovy, Griffon, many Spring IO platform projects

They're using Gradle

- Gradle, Groovy, Griffon, many Spring IO platform projects
- Hibernate

They're using Gradle

- Gradle, Groovy, Griffon, many Spring IO platform projects
- Hibernate
- Official build tool in Android project

They're using Gradle

- Gradle, Groovy, Griffon, many Spring IO platform projects
- Hibernate
- Official build tool in Android project
- LinkedIn, all Netflix's OpenSource projects, Orbitz

They're using Gradle

- Gradle, Groovy, Griffon, many Spring IO platform projects
- Hibernate
- Official build tool in Android project
- LinkedIn, all Netflix's OpenSource projects, Orbitz
- Goldman Sachs has custom build system based on Gradle

Walkthrough 4

- Checking Gradle installation
    - Gradle UI (*`gradle -ui`*)
- Creating Gradle build file
    - demonstration of *`gradle init`*
    - *`./gradlew`*
- Running Gradle tasks
    - *`gradle tasks (–all)`*
    - *`gradle test`*
    - *`gradle run`*
- Gradle support for IDEs
    - *`gradle idea`*
    - *`gradle eclipse`*
- Gradle support in IntelliJ IDEA
    - Import Gradle module/project in IDEA
    - Gradle tasks view

Additional reading

- Video from Gradle creator and lead explaining difference between build tools http://www.infoq.com/presentations/compare-build-tools

- Pretty good info from Hibernate project member https://community.jboss.org/wiki/Gradlewhy

- Gradle Effective Implementation Guide http://www.amazon.com/Gradle-Effective-Implementation-Hubert-Ikkink-ebook/dp/B009X5KIFK

- Gradle in Action http://www.amazon.com/Gradle-Action-Benjamin-Muschko/dp/1617291307/

Spock Framework

Live long, prosper and test!


I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO, I DO IT IN PRODUCTION

What is Spock?

What is Spock?

- **Sp**ecification and M**ock**

What is Spock?

- **Sp**ecification and M**ock**
- Groovy DSL for Testing

FARATA

What is Spock?

- **Sp**ecification and M**ock**
- Groovy DSL for Testing
- Compatible on JUnit4

What is Spock?

- **Sp**ecification and M**ock**
- Groovy DSL for Testing
- Compatible on JUnit4
- Goal: Simple, Concise, Maintainable tests

FARATA

What is Spock?

- **Sp**ecification and M**ock**
- Groovy DSL for Testing
- Compatible on JUnit4
- Goal: Simple, Concise, Maintainable tests
- BDD: Given/When/Then

FARATA

Spock elements

Spock elements

- Blocks

Spock elements

- Blocks
  - `setup: cleanup: expect: given: when: then: where: and:`

Spock elements

- Blocks
  - `setup: cleanup: expect: given: when: then: where: and:`
- Fixture Methods

Spock elements

- Blocks
  - `setup: cleanup: expect: given: when: then: where: and:`
- Fixture Methods
  - `setup(), cleanup(), setupSpec(), cleanupSpec()`

Spock elements

- Blocks
  - `setup: cleanup: expect: given: when: then: where: and:`
- Fixture Methods
  - `setup(), cleanup(), setupSpec(), cleanupSpec()`
- Instance and `@Shared` fields

Spock elements

- Blocks
  - `setup: cleanup: expect: given: when: then: where: and:`
- Fixture Methods
  - `setup(), cleanup(), setupSpec(), cleanupSpec()`
- Instance and `@Shared` fields
- `old()` and `thrown()`

old()

```
def "push"() {
    when:
    stack.push("elem")

    then:
    stack.size() == old(stack.size()) + 1
}
```


Data Driven Tests

- `where:` block
- Data tables
- `@Unroll`
- External data sources

Data Driven Test

```
def "maximum of two numbers"() {
    expect:
    Math.max(a, b) == c

    where:
    a << [3, 5, 9]
    b << [7, 4, 9]
    c << [7, 5, 9]
}
```

```
@Unroll("minimum of #a and #b is #c")
def "minimum of two numbers"() {
    expect:
    Math.min(a, b) == c

    where:
    [a, b, c] << [[3, 7, 3], [5, 4, 4], [9, 9, 9]]
}
```


Interaction testing

- Mocks
- Stubs
- Spies

Walkthrough 5

- Spock spec elements
- Data driven tests
    - Unrolled report
        - using external datasource
- Interaction test
- Running the tests in IntelliJ IDEA
    - run particular test

Additional Reading

- Spock home https://code.google.com/p/spock/
- Spock documentation site http://docs.spockframework.org/en/latest/index.html
- Spock WebConsole http://meetspock.appspot.com/

Java 8 Homework

- Using the provided classes in the package `streams.auction` write a Java program that will implement the auction bidding rules below. Use Streams API with the collection of `Bid` objects. Use `Comparator` and sort the stream.

- Multiple people can bid on a `Product`. Just create a couple of `Product` instances with hardcoded data. Use `TimerTask` and random number generator to emulate bidding process.

- A collection of `Bid` objects represents current bids

- When a “user” places a new bid, add it to the collection of bids, and “send an email” (just print on the console) to all bidders who opted for receiving overbid emails.

- If a bid is greater or equal to the `Product` reserved price, send the bidder a winning email.

- If a bid is less that a min `Product` price, send a bidder a sorry email.

- Implement the test suite for auction prototype using Spock Framework