Finish him

#10 Practice

Lexer Factory

- Want to create two different Lexer instances
    - Old Lexer
    - New Lexer with State Machine

Lexer Factory

Two factories

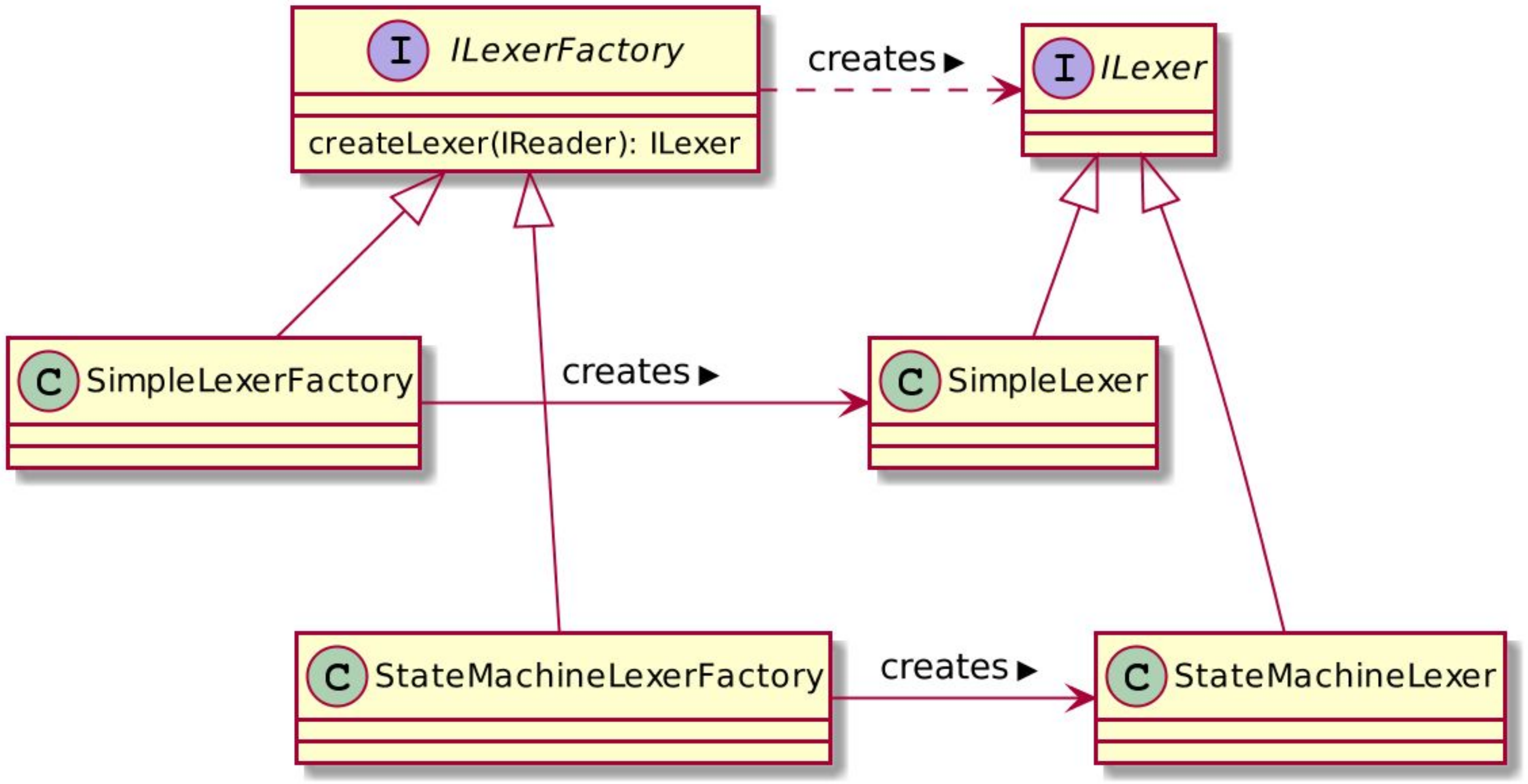
```
public interface ILexerFactory {
   ILexer createLexer(IReader reader);
}

public interface IStateMachineLexerFactory {
   IStateMachineLexer createLexer(IReader reader);
}
```

Lexer Factory

Two factory methods

```
public interface ILexerFactory {
    ILexer createLexer(IReader reader);
    IStateMachineLexer createStateMachineLexer(IReader
reader);
}
```

Lexer Factory



Config

- Externalize commands map
- Externalize state transitions map
- To never modify the code
- In glory of SOLID

JSON

```json
[
 {
   "state": "default",
   "actions": [
     {
       "input": null,
       "command": "CharLexeme",
       "state": "default"
     },
     {
       "input": " ",
       "command": "SpaceLexeme",
       "state": "spacing"
     }
   ]
 }
]
```

GSON

```xml
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
</dependency>
```

Gson

```
InputStream file = LexerStateMachineLoader.class.
      getResourceAsStream("/lexer.json");
Gson gson = new Gson();
JsonArray states = gson.fromJson(
      new InputStreamReader(file), JsonArray.class);
for (JsonElement state : states) {
   JsonObject stateObject = state.getAsJsonObject();
   String stateName =
stateObject.get("state").getAsString();
   JsonArray actions =
stateObject.getAsJsonArray("actions");
   for (JsonElement action: actions) {
      //...
   }
```

YAML

Yet Another Markup Language

```yaml
- state: 'default'
  actions:
    - input: null
      command: 'CharLexeme'
      state: 'default'
    - input: ' '
      command: 'SpaceLexeme'
      state: 'spacing'
```

SnakeYAML

```xml
<dependency>
    <groupId>org.yaml</groupId>
    <artifactId>snakeyaml</artifactId>
    <version>1.17</version>
</dependency>
```

ShakeYAML

```
InputStream file = LexerStateMachineLoader.class.
      getResourceAsStream("/lexer.yaml");
Yaml yaml = new Yaml();
List statesDefs = (List) yaml.load(file);
for (Object stateDefObject : statesDefs) {
   Map stateDef = (Map) stateDefObject;
   String stateName = stateDef.get("state").toString();
   List actionsDefs = (List) stateDef.get("actions");
   for (Object actionDef : actionsDefs) {
      //...
   }
}
```

Command Definition

```
command: 'CharLexeme'
command: 'SpaceLexeme'
```

Command Definition

- Script
  - Need to add a script language interpreter
- Name
  - Need to find implementation by name
    - Again map or switch
- Class Name
  - Need to instantiate class

Reflection

```
static final String COMMAND_PACKAGE =
        "formatter.lexer.commands";

public ICommand createCommand(final String className)
        throws ClassNotFoundException,
        IllegalAccessException, InstantiationException {
    String fullName = COMMAND_PACKAGE + "." + className;
    return (ICommand) Class.forName(fullName).newInstance();
}
```

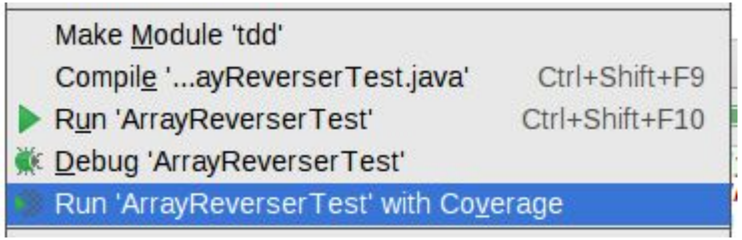
Tests Refactoring?

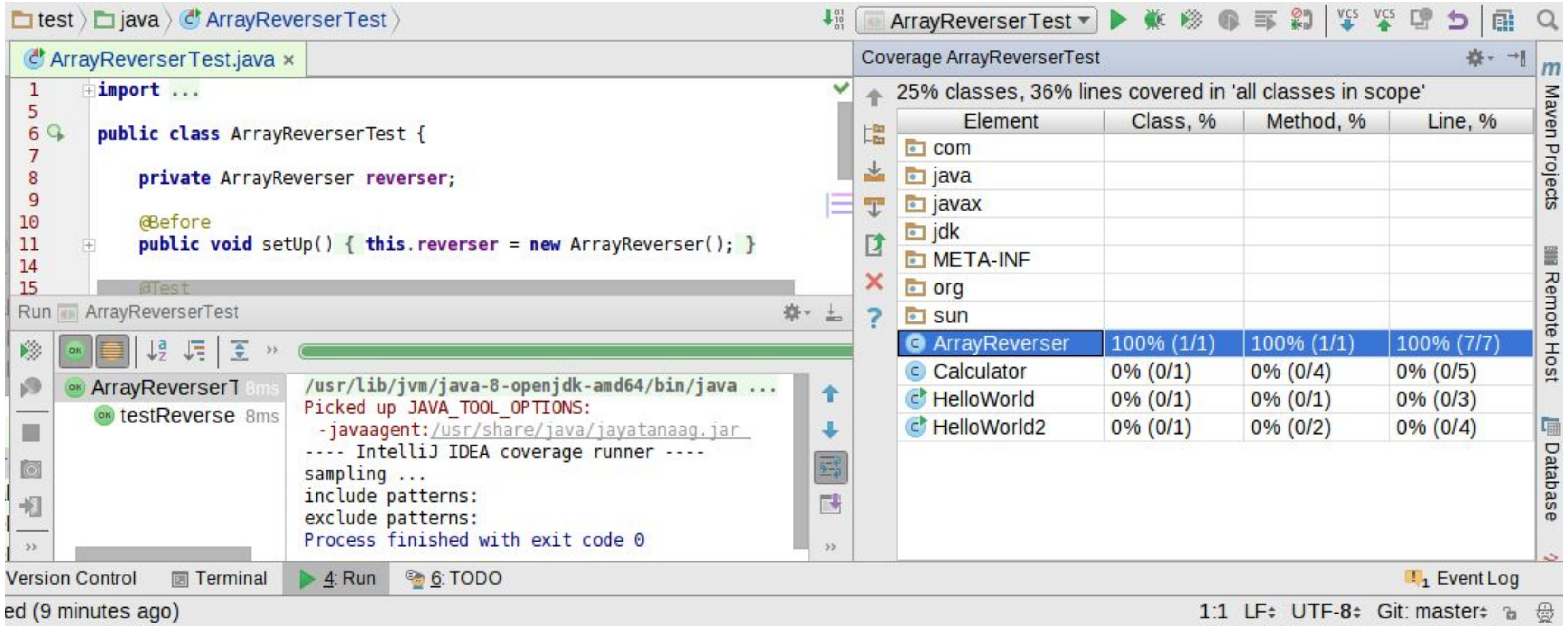
- Two or three Lexer implementation
- Do they need specific tests?
- Do they need the same tests?

Test Coverage

- Classes
- Methods
- Lines of code
- Branches

Coverage in IDEA


Make Module 'tdd'
Compile '...ayReverserTest.java' Ctrl+Shift+F9
Run 'ArrayReverserTest' Ctrl+Shift+F10
Debug 'ArrayReverserTest'
Run 'ArrayReverserTest' with Coverage



```
import ...

public class ArrayReverserTest {

    private ArrayReverser reverser;

    @Before
    public void setUp() { this.reverser = new ArrayReverser(); }

    @Test
```

ArrayReverserT 8ms
testReverse 8ms

```
/usr/lib/jvm/java-8-openjdk-amd64/bin/java ...
Picked up JAVA_TOOL_OPTIONS:
 -javaagent:/usr/share/java/jayatanaag.jar
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
exclude patterns:
Process finished with exit code 0
```

25% classes, 36% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
com			
java			
javax			
jdk			
META-INF			
org			
sun			
ArrayReverser	100% (1/1)	100% (1/1)	100% (7/7)
Calculator	0% (0/1)	0% (0/4)	0% (0/5)
HelloWorld	0% (0/1)	0% (0/1)	0% (0/3)
HelloWorld2	0% (0/1)	0% (0/2)	0% (0/4)

Coverage with Maven

- JaCoCo
- http://www.jacoco.org/jacoco/trunk/doc/maven.html

Use Cases

1. As a User I want to format a code containing semicolons and curly brackets
2. As a User I want to format a code containing string literals
3. As a User I want to format a code containing single-line and multi-line comments
4. As a User I want to format a code containing "for" loops