

Unit Testing – Our Basic Components Work in Isolation



Cătălin Tudose

PHD IN COMPUTER SCIENCE, JAVA AND WEB TECHNOLOGIES EXPERT

<https://www.linkedin.com/in/catalin-tudose-847667a1>



Overview



Unit testing benefits

- Safer code
- Find bugs early
- Isolate incorrect code
- Easily introduce new functionality
- Document the application

Code coverage

Move the application to unit testing with JUnit5

Add new features that are unit tested





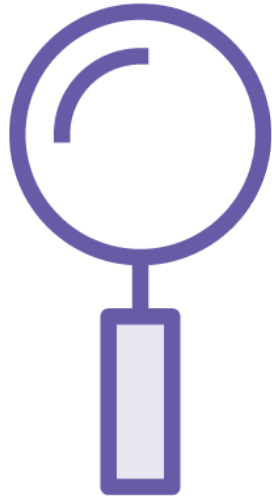
Safer code

Think harder about the code

Exposes the edge cases



Find Bugs Early



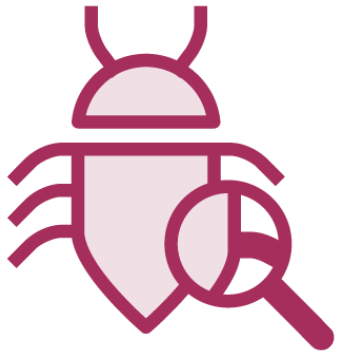
Bugs found at an early stage



Unit testing by developers



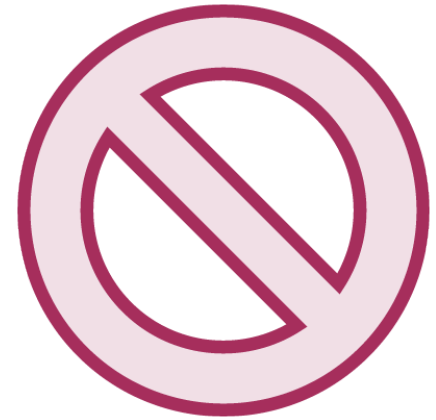
Isolate Incorrect Code



Incorrect code



Attention!



No Entry!



Easily Introduce New Functionality



Developer



Functionality

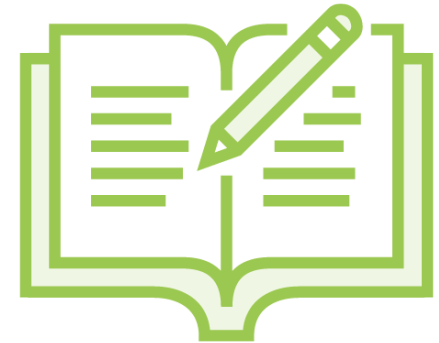
Document the Application



Read Documentation



Read Documentation



Write Documentation



Code Coverage

A measure used to describe the degree to which the source code of a program is executed when a particular test suite runs.



Code Coverage Tools

JCov

OpenClover

EMMA

JaCoCo



What Code Coverage Percentage Is Feasible?



80%





90%










100%



Code Coverage Results

 [test_pyramid_strategy](#) >  com.pluralsight.test_pyramid_strategy.airport

com.pluralsight.test_pyramid_strategy.airport













Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 Flight		0%		n/a	14	14	28	28	14	14	1	1
 Passenger		67%		100%	4	13	6	29	4	9	0	1
 Main		0%		n/a	2	2	10	10	2	2	1	1
Total	181 of 261	30%	0 of 8	100%	20	29	44	67	20	25	2	3



Code Coverage Results

 test_pyramid_strategy >  com.pluralsight.test_pyramid_strategy.airport >  Passenger

Passenger

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxy	Missed	Lines	Missed	Methods
toString()		0%		n/a	1	1	1	1	1	1
recordToSystem()		0%		n/a	1	1	2	2	1	1
setName(String)		0%		n/a	1	1	2	2	1	1
getName()		0%		n/a	1	1	1	1	1	1
Passenger(String, String, String)		100%		100%	0	3	0	12	0	1
setIdentifier(String)		100%		100%	0	2	0	5	0	1
setCountryCode(String)		100%		100%	0	2	0	4	0	1
getIdentifier()		100%		n/a	0	1	0	1	0	1
getCountryCode()		100%		n/a	0	1	0	1	0	1
Total	38 of 118	67%	0 of 8	100%	4	13	6	29	4	9



Code Coverage Results

test_pyramid_strategy > com.pluralsight.test_pyramid_strategy.airport > Passenger.java

Passenger.java

```
1. package com.pluralsight.test_pyramid_strategy.airport;
2.
3. import java.util.Arrays;
4. import java.util.Locale;
5. import java.util.regex.Matcher;
6. import java.util.regex.Pattern;
7.
8. public class Passenger {
9.
10.     private String identifier;
11.     private String name;
12.     private String countryCode;
13.     String regex = "(?!000|666)[0-8][0-9]{2}-(?!00)[0-9]{2}-(?!0000)[0-9]{4}$";
14.     Pattern pattern = Pattern.compile(regex);
15.
16.     public Passenger(String identifier, String name, String countryCode) {
17.         Matcher matcher = pattern.matcher(identifier);
18.         if (!matcher.matches()) {
19.             throw new RuntimeException("Invalid identifier");
20.         }
21.
22.         if (!Arrays.asList(Locale.getISOCountries()).contains(countryCode)) {
23.             throw new RuntimeException("Invalid country code");
24.         }
25.
26.         this.identifier = identifier;
27.         this.name = name;
28.         this.countryCode = countryCode;
29.     }
30.
31.     public String getIdentifier() {
32.         return identifier;
```



100% code coverage does
not mean your code works
perfectly.



Demo



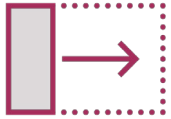
Flights management application

Unit test Passenger with JUnit5

- Creation of a passenger
- Restrictions on identifier and country code
- Methods behavior



Identifier Rules



SSNs: 9-digit numbers, AAA-GG-SSSS



The first three digits cannot be 000, 666, or between 900 and 999



Digits 4 and 5: group number, from 01 to 99



Last 4 digits: serial numbers from 0001 to 9999



Demo



Unit test Flight with JUnit5

- Creation of a flight
- Restrictions on flight number format
- Business logic of a flight



Demo



New business logic

- New identifier format for non-US passengers
- Better manage the status of a flight



Summary



Unit testing benefits

Code coverage

Move the application to unit testing with JUnit5

- Passengers identifiers and country codes restrictions
- Flights numbers restrictions
- Bad input values
- Boundary conditions

Add new features that are unit tested

- Separated the identifier pattern
- Better management of flights statuses

