TDD with JUnit 5

INTRODUCING TDD AND JUNIT5

Cătălin Tudose
PHD IN COMPUTER SCIENCE, JAVA AND WEB TECHNOLOGIES EXPERT
https://www.linkedin.com/in/catalin-tudose-847667a1

Overview

TDD benefits

- Driven by clear goals
- Safer code
- Isolate incorrect code
- Easily introduce new functionality
- Document the application

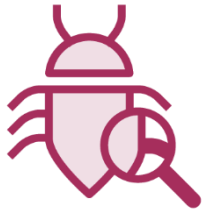
Move an application to TDD using JUnit5

Is your application doing what it is supposed to do?

Code Production Lifecycle

Write code

Test



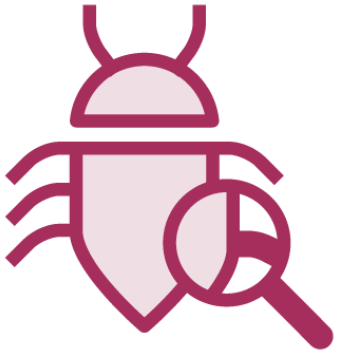
Debug

Write code

Test

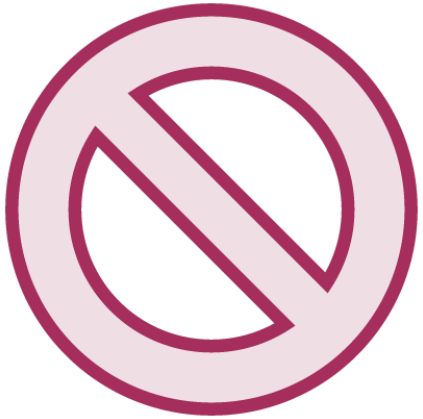
Debug

Isolate Incorrect Code



Incorrect code

Attention!



No Entry!

Easily Introduce New Functionality

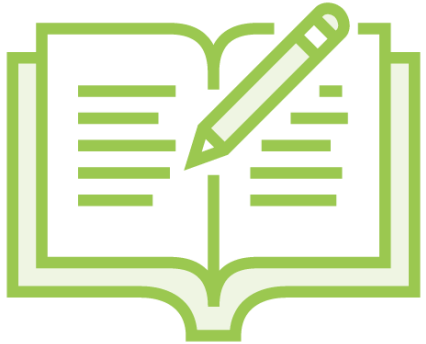
Developer

Functionality

Document the application

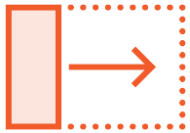
Read Documentation

Read Documentation



Write Documentation

Move an application to TDD using JUnit5



Non-TDD working application as input

Write unit tests for the existing code

Working cycle: write tests for the new functionality, then implement it

Code Coverage

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

Code Coverage Results

TDD > com.pluralsight.tddjunit5.airport

com.pluralsight.tddjunit5.airport

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Flight		70%		66%	11	20	5	20	3	6	0	1
Passenger		80%		n/a	1	3	1	6	1	3	0	1
Total	39 of 138	71%	8 of 24	66%	12	23	6	26	4	9	0	2

Code Coverage Results

TDD > com.pluralsight.tddjunit5.airport > Flight

Flight

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
addPassenger(Passenger)		74%		66%	4	8	1	6	0	1
removePassenger(Passenger)		72%		66%	4	8	1	6	0	1
getPassengersList()		0%		n/a	1	1	1	1	1	1
getId()		0%		n/a	1	1	1	1	1	1
getFlightType()		0%		n/a	1	1	1	1	1	1
Flight(String, String)		100%		n/a	0	1	0	5	0	1
Total	36 of 123	70%	8 of 24	66%	11	20	5	20	3	6

Code Coverage Results

TDD > com.pluralsight.tddjunit5.airport > Flight.java

Flight.java

```
1.  package com.pluralsight.tddjunit5.airport;
2.
3.  import java.util.ArrayList;
4.  import java.util.Collections;
5.  import java.util.List;
6.
7.  public class Flight {
8.
9.      private String id;
10.     List<Passenger> passengersList = new ArrayList<Passenger>();
11.     private String flightType;
12.
13.     public Flight(String id, String flightType) {
14.         this.id = id;
15.         this.flightType = flightType;
16.     }
17.
18.     public String getId() {
19.         return id;
20.     }
21.
22.     public List<Passenger> getPassengersList() {
23.         return Collections.unmodifiableList(passengersList);
24.     }
```

What Code Coverage Percentage Is Feasible?

80%

90%

100%

100% code coverage does not mean your code works perfectly

JUnit 5 Architecture

JUnit 4 Architecture

junit.jar

A single JAR file

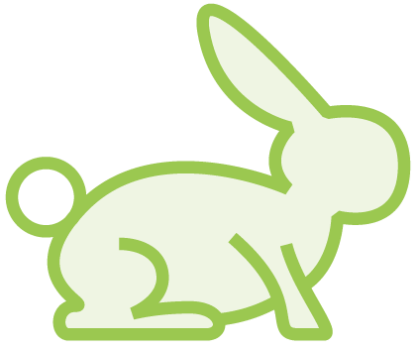
No flexible API

Used by everyone

Modular Approach

Write tests

Discover and run the tests



Run tests from the IDEs and tools

JUnit 5 Modules

JUnit Platform

JUnit Jupiter

JUnit Vintage

JUnit Platform

junit-platform-commons

junit-platform-console

junit-platform-console-standalone

JUnit Platform

junit-platform-engine

junit-platform-launcher

junit-platform-runner

JUnit Platform

junit-platform-suite-api

junit-platform-surefire-provider

junit-platform-gradle-plugin

JUnit Jupiter

junit-jupiter-api

junit-jupiter-engine

junit-jupiter-params

junit-jupiter-migrationsupport

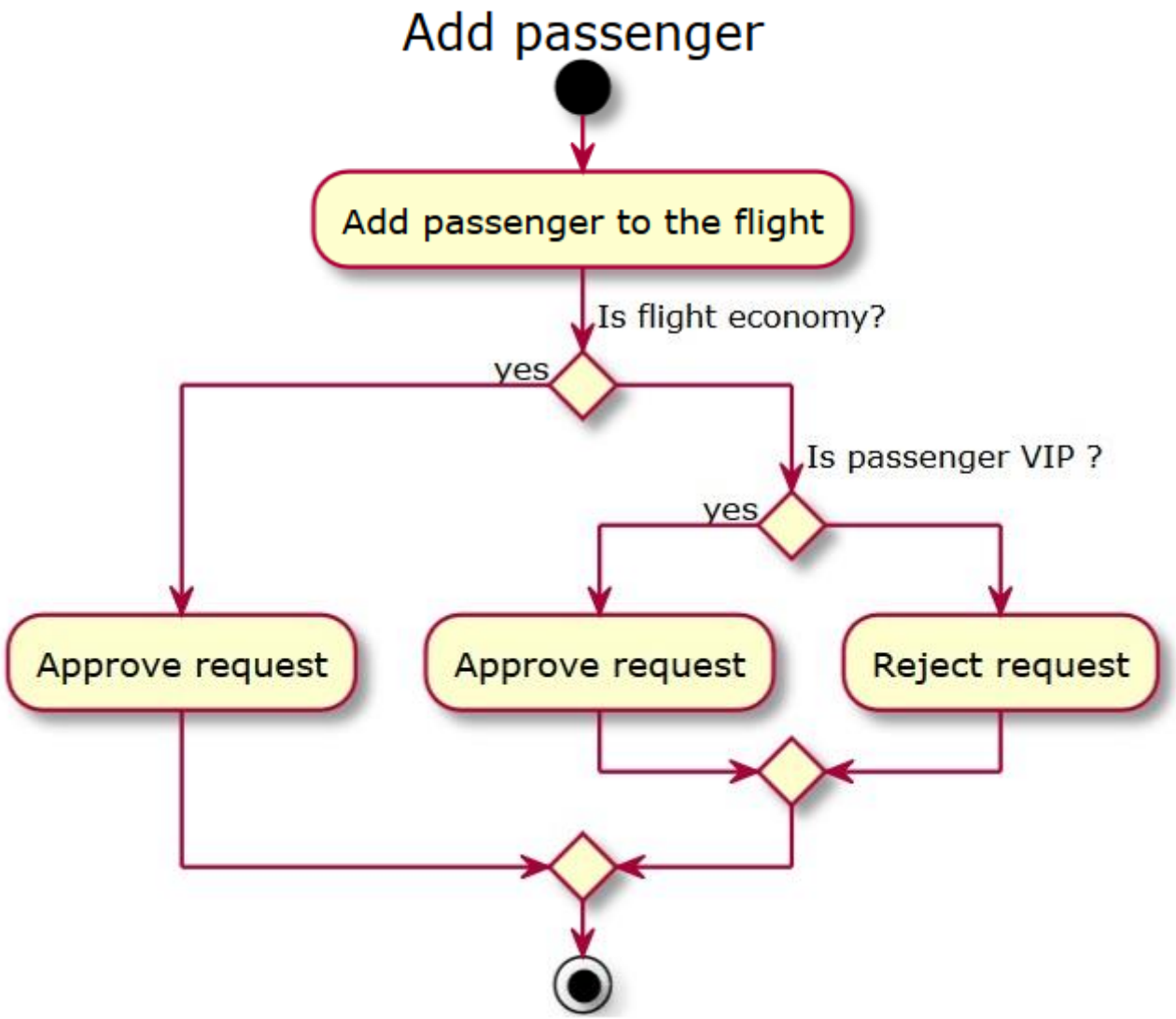
JUnit Vintage

junit-vintage-engine

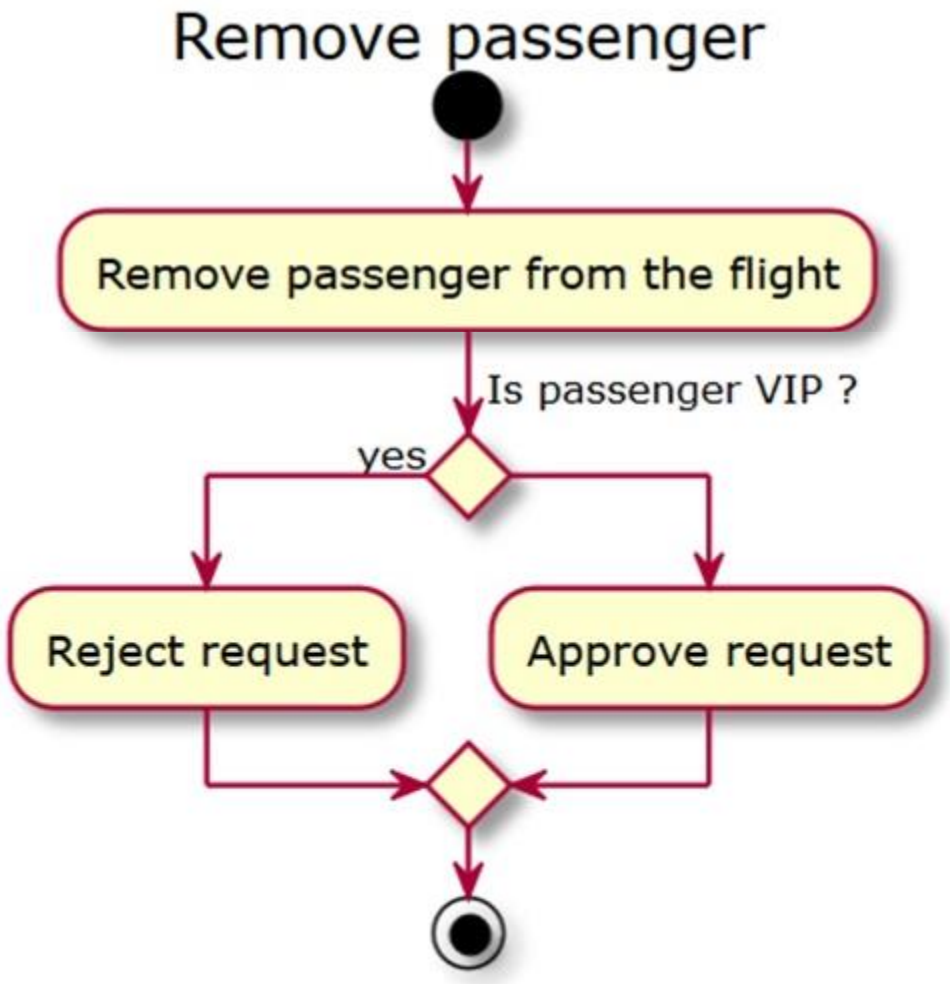
JUnit 3 or 4 JARs

The Flight Management Application

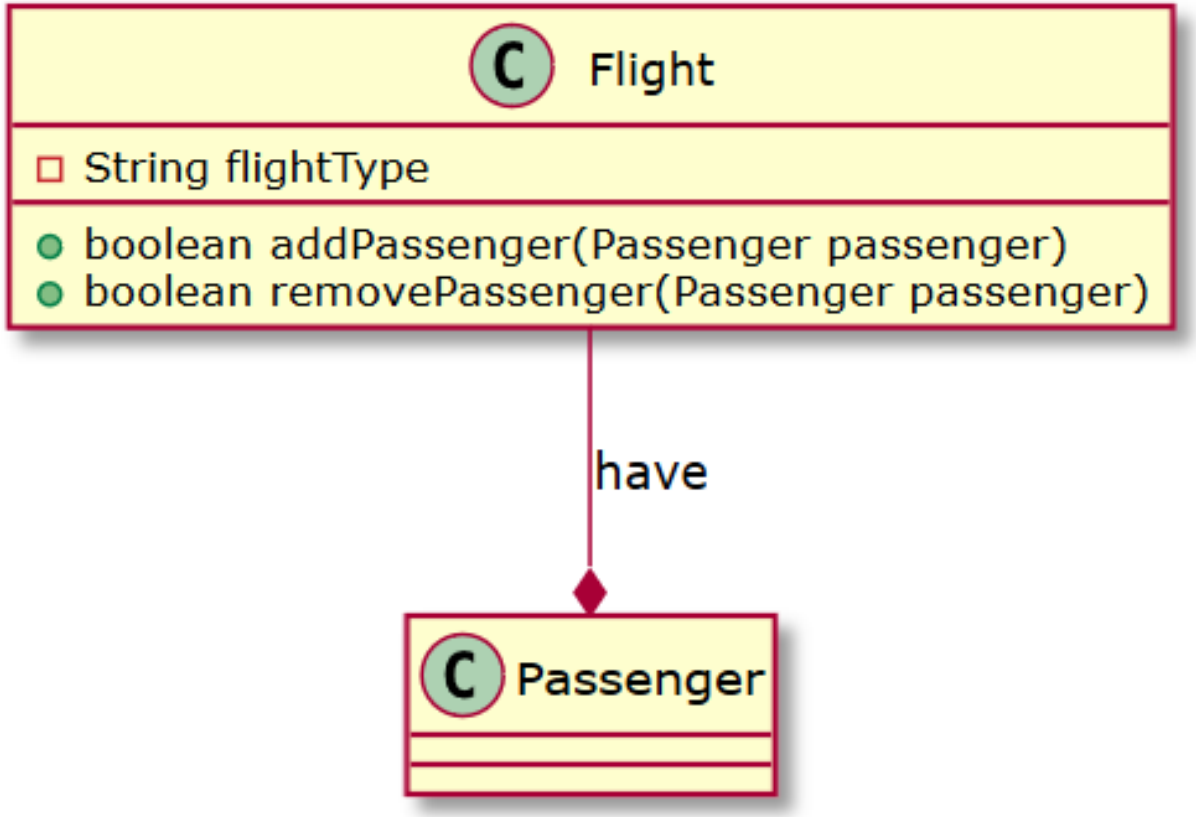
Adding Passenger Business Logic



Removing Passenger Business Logic



Initial application design



Demo

Flights management application

- Simulate real life scenario
- Start from a non-TDD application

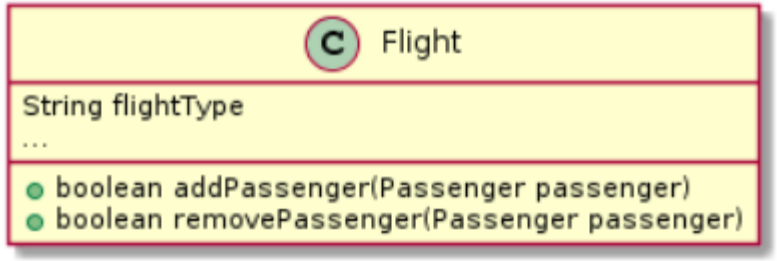
Check its functionality

Write unit tests for the initial code

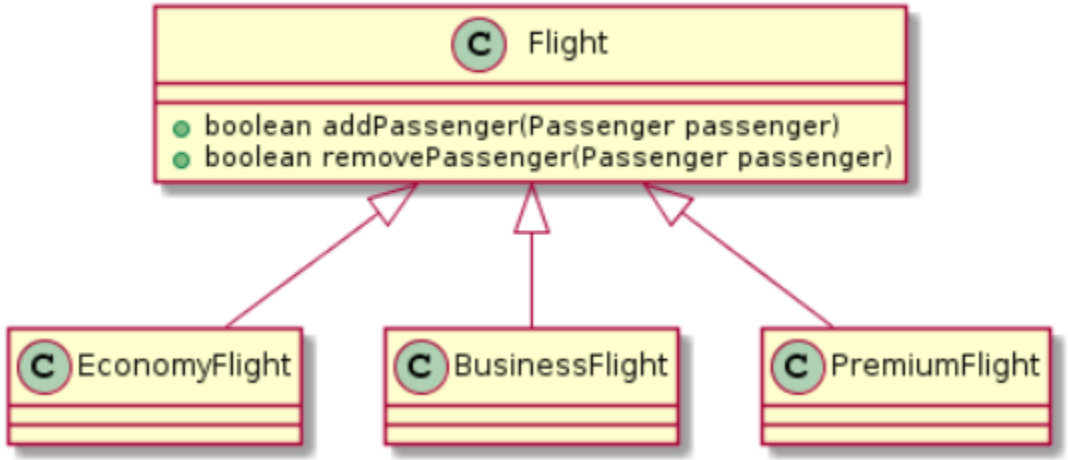
Measure code coverage and improve it

Replacing conditional with polymorphism

Using conditional



Using polymorphism



Summary

TDD and its benefits

Code Coverage

JUnit 5

Flight management application

Add features TDD style using JUnit5

- Real life simulation