American Airlines

Acceptance Automation Testing Strategy

From “legacy code” to delivering product value

TNT coaching pod
Yelena Gouralnik, David Hammerslag, Lucas DeSouza, Amber Waller

Tech 2.0 Metrics

Yes, lack of automation testing strategy will impact your metrics.

Automating SOLID checks

Yes, local installation of a linter will ensure all of your SOLID principles are covered.

Developing Safe Software on Different Levels

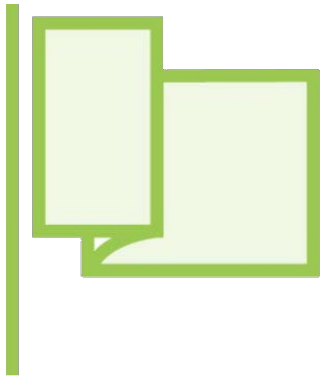
- Software Testing and its purpose
- Types of Testing
- High-Level testing, low-level testing
- Manual Testing
- Test Automation
- What is a test?
- The test “pyramid”
- Tools, framework, environments
- Starting with Legacy Application

Software Testing

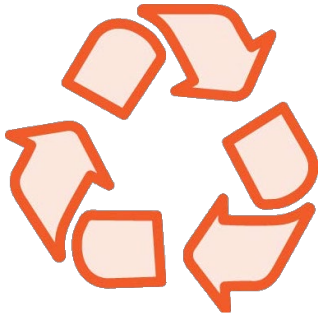
- Making objective judgement regarding the extent to which the system meets, exceeds, or fails to meet stated objectives.

Purpose of Testing

Verifying stated objectives and requirements



Uncover negative impact to customers and maintainability

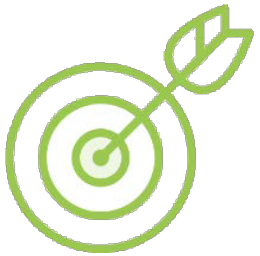


Mitigate Risk

Why Test At All?

Understand Software

Stability of the System



Meets User Expectations

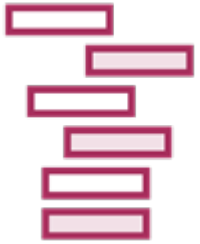
Regression

Quality

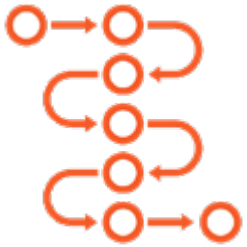
Work

Types of Testing

Unit



Integration



System

Acceptance

High Level vs. Low Level Testing

- High Level
  - More abstract, it describes overall goals and systemic features. Typically more concerned with the interaction with the user through GUI with the system as a whole or larger components of the system.
- Low Level
  - Low-level testing describes individual components, it provides the test rather than overview, rudimentary functions rather than complex, it is typically more concerned with individual components within the system and how they operate.

Manual Testing

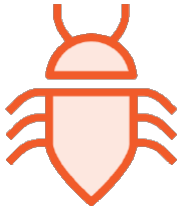
F. C. Baia Mare	– C. S. Tîrgoviște	5–0	(1–0)
Jiul Petroșani	– Steaua	1–0	(0–0)
F. C. Bihor	– Gloria Buzău	2–1	(0–1)
Dinamo	– F. C. Argeș	3–4	(1–2)
A.S.A. Tg. Mureș	– S. C. Bacău	3–1	(2–0)
U. T. Arad	– Politehnica Iași	6–2	(2–2)
Corvinul Hunedoara	– „Poli" Timișoara	3–1	(1–0)
Olimpia Satu Mare	– Sportul studențesc	1–0	(1–0)
Univ. Craiova	– Chimia Rm. Vîlcea	4–1	(1–0)

INPUT

1.	F. C. ARGEȘ	34	20	5	9	54–29	45
2.	Dinamo	34	16	9	9	51–28	41
3.	Steaua	34	18	4	12	57–32	40
4.	Univ. Craiova	34	15	8	11	40–25	38
5.	F. C. Baia Mare	34	17	4	13	42–38	38
6.	Sportul stud.	34	14	7	13	42–41	35
7.	C. S. Tîrgoviște	34	15	5	14	38–38	35
8.	S. C. Bacău	34	14	6	14	37–33	34
9.	A.S.A. Tg. Mureș	34	13	6	15	49–59	32
10.	Olimpia	34	14	4	16	38–52	32
11.	„Poli" Timișoara	34	13	5	16	35–37	31
12.	Politehnica Iași	34	11	9	14	37–44	31
13.	Gloria Buzău	34	13	5	16	34–46	31
14.	Jiul	34	13	5	16	38–51	31
15.	Chimia Rm. V.	34	13	5	16	38–54	31
16.	Corvinul	34	13	4	17	45–50	30
17.	U. T. Arad	34	11	7	16	45–46	29
18.	F. C. Bihor	34	10	8	16	37–49	28

OUTPUT

Automation Testing



Bugs are one of the outputs of testing.

Manual to Automation

Automated Tests may be created, parameterized, and reputedly executed.

What to Test?

Business logic	Unexpected conditions	Boundary conditions
Invariants	Bad input values	Regressions

Testing Pyramid Revealed

Unit Testing

Integration Testing

System Testing

Acceptance Testing

Tools, Frameworks, and Environment

JUnit, TestNG

Mockito, PowerMock

Arquillian

Cucumber, JBehave

Demo

Flight Management Application

No Tests

Business Logic

Classes

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help test_pyramid_strategy - Passenger.java [test_pyramid_strategy]

Before > airport > src > main > java > com > tnttest > test_pyramid_strategy > airport > Passenger

Add Configuration...

Project

- Before D:\training\TDD\implementing-test-pyramid-s
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
        - java
          - com.tnttest.test_pyramid_strategy.airp
            - Flight
            - Main
            - Passenger
      - test
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
  - < 1.8 > C:\Program Files (x86)\Java\jdk1.8.0_241
- Scratches and Consoles

Passenger.java | pom.xml (test_pyramid_strategy)

```java
package com.tnttest.test_pyramid_strategy.airport;

public class Passenger {

    private String identifier;
    private String name;
    private String countryCode;

    public Passenger(String identifier, String name, String countryCode) {
        this.identifier = identifier;
        this.name = name;
        this.countryCode = countryCode;
    }

    public String getIdentifier() { return identifier; }

    public void setIdentifier(String identifier) { this.identifier = identifier; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getCountryCode() { return countryCode; }

    public void setCountryCode(String countryCode) { this.countryCode = countryCode; }
```

TODO | 6: Problems | Terminal | Build | Event Log

48:1 CRLF UTF-8 Tab*

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help test_pyramid_strategy - Passenger.java [test_pyramid_strategy]

Before > airport > src > main > java > com > tnttest > test_pyramid_strategy > airport > Passenger

Add Configuration...

Project

- Before D:\training\TDD\implementing-test-pyramid-s
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
        - java
          - com.tnttest.test_pyramid_strategy.airp
            - Flight
            - Main
            - Passenger
      - test
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
  - < 1.8 > C:\Program Files (x86)\Java\jdk1.8.0_241
- Scratches and Consoles

Passenger.java | pom.xml (test_pyramid_strategy)

```java
    }

    public String getIdentifier() { return identifier; }

    public void setIdentifier(String identifier) { this.identifier = identifier; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getCountryCode() { return countryCode; }

    public void setCountryCode(String countryCode) { this.countryCode = countryCode; }


    @Override
    public String toString() {
        return "Passenger " + getName() + " with identifier: " + getIdentifier() + " from " + getCountryCode();
    }

    public void recordToSystem() { System.out.println(this + " has been recorded to the company system"); }

}
```

TODO 6: Problems Terminal Build Event Log

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help test_pyramid_strategy - Flight.java [test_pyramid_strategy]

Before 〉 airport 〉 src 〉 main 〉 java 〉 com 〉 tnttest 〉 test_pyramid_strategy 〉 airport 〉 Flight

Add Configuration...

Project

- Before D:\training\TDD\implementing-test-pyramid-s
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
        - java
          - com.tnttest.test_pyramid_strategy.airp
            - Flight
            - Main
            - Passenger
      - test
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
  - < 1.8 > C:\Program Files (x86)\Java\jdk1.8.0_241
- Scratches and Consoles

Passenger.java | Flight.java | pom.xml (test_pyramid_strategy)

```java
package com.tnttest.test_pyramid_strategy.airport;

public class Flight {

    private String flightNumber;
    private int places;
    private int passengers;
    private String origin;
    private String destination;
    private boolean flying;

    public Flight(String flightNumber, int places) {
        this.flightNumber = flightNumber;
        this.places = places;
        this.passengers = 0;
        this.flying = false;
    }

    public String getFlightNumber() { return flightNumber; }

    public int getPlaces() { return places; }

    public void setPlaces(int places) { this.places = places; }

    public int getPassengers() { return passengers; }

    public String getOrigin() { return origin; }

    public void setOrigin(String origin) { this.origin = origin; }
```

TODO 6: Problems Terminal Build Event Log

7:28 CRLF UTF-8 4 spaces

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help test_pyramid_strategy - Main.java [test_pyramid_strategy]

Before > airport > src > main > java > com > tnttest > test_pyramid_strategy > airport > Main

Project

- Before D:\training\TDD\implementing-test-pyramid-s
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
        - java
          - com.tnttest.test_pyramid_strategy.airp
            - Flight
            - Main
            - Passenger
      - test
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
  - < 1.8 > C:\Program Files (x86)\Java\jdk1.8.0_241
- Scratches and Consoles

Passenger.java | Flight.java | Main.java | pom.xml (test_pyramid_strategy)

```java
package com.tnttest.test_pyramid_strategy.airport;

public class Main {

    public static void main(String[] args) {
        Passenger passenger = new Passenger( identifier: "123-45-6789",  name: "John Smith",  countryCode: "UK");
        passenger.recordToSystem();

        Flight flight = new Flight( flightNumber: "BA884",  places: 100);
        flight.setOrigin("London");
        flight.setDestination("Bucharest");
        flight.sellTicket();
        flight.takeOff();
        flight.land();
    }
}
```

Run: Main

```
"C:\Program Files (x86)\Java\jdk1.8.0_241\bin\java.exe" ...
Passenger John Smith with identifier: 123-45-6789 from UK has been recorded to the company system
Ticket for Flight BA884 from London to Bucharest sold
Flight BA884 from London to Bucharest is taking off
Flight BA884 from London to Bucharest is landing

Process finished with exit code 0
```

4: Run | TODO | 6: Problems | Terminal | Build | Event Log

Build completed successfully in 6 s 259 ms (30 minutes ago)

3:14 CRLF UTF-8 4 spaces

Unit Testing – Our Basic Components Work in Isolation

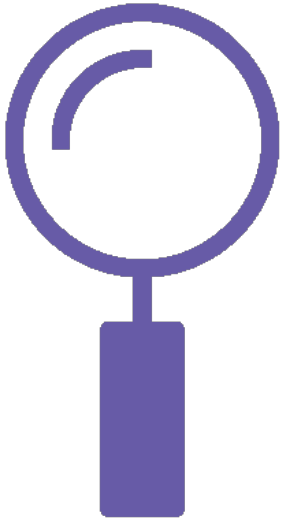
Testing Individual Units and Components

Adding Features Using Unit Tests

Unit Testing Benefits

Safer code
Spend More Time Thinking About The Code
Exposes Edge Cases
Easily introduce new functionality
Documents the application

Unit Testing Benefits



Bugs are found at an early stage

Developers can catch bugs related To bad design and impelemntation

Isolate Incorrect Code

Isolate lines that causing issues

Failed unit test linked to code causing unexpected behavior.

Easily Introduce New Functionality

Writing Test First helps developers confidently add new functionality without causing regression

Writing Test First helps developers understand business value of the code.

Document The Application

Well Written Unit Tests Are Self-Documenting

No need to Reference External Documentation

Execute existing code by running a simple test.

Easy to On-Board

Unit Tests Become Documentation

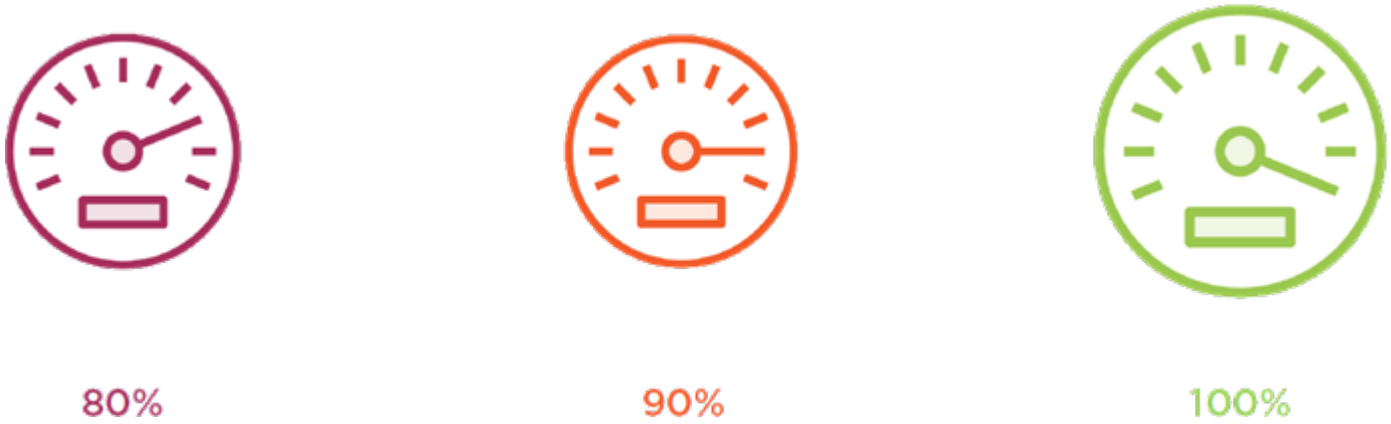
Code Coverage

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

Higher Percentage – more source code executed during testing

Coverage Metrics – percentage of program methods and the percentage of program lines called during execution of the test suite

What Code Coverage Percentage is Feasible?



80% is the minimum percentage that is required.

Included into definition of done

Unit Testing alone will not get you to 100% of code coverage

Sometimes 100% is not possible, but should always be a target.

JaCoCo

- JaCoCo Agent runs when test are run
  - Agent collects the coverage information.

Code Coverage Results – Class Level

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

JaCoCo

Code Coverage Results on a method level

test_pyramid_strategy > com.pluralsight.test_pyramid_strategy.airport > Passenger

Passenger

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	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

JaCoCo

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

Passenger.java

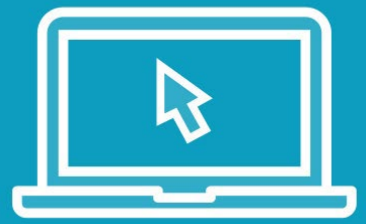
```
1.  package com.pluralsight.test_pyramid_strategy.airport;
2.
3.  import java.util.Arrays;
4.  import java.util.Locale;
5.  import java.util.regex.Matcher;
6.  import java.util.regex.Pattern;
7.
8.  public class Passenger {
9.
10.     private String identifier;
11.     private String name;
12.     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?

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

- Test Scripts do not test anything substantial
- Unit Testing only covers classes and methods in isolation and does not constitute full scope of automated testing.
- Unit Tests do not cover interaction between different classes nor do they exhaust all the possible use cases.

Demo



Flight Management Application

Unit Test Passenger with JUnit

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

Flight Management Application – No Unit Tests

Flight Management Application

Follow along at:

https://ghe.aa.com/00919842/AutomationFrameworkStrategy.git

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help test_pyramid_strategy - Passenger.java [test_pyramid_strategy]

Before > airport > src > main > java > com > tnttest > test_pyramid_strategy > airport > Passenger

Add Configuration...

Project

- Before D:\training\TDD\implementing-test-pyramid-s
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
        - java
          - com.tnttest.test_pyramid_strategy.airp
            - Flight
            - Main
            - Passenger
      - test
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
  - < 1.8 > C:\Program Files (x86)\Java\jdk1.8.0_241
- Scratches and Consoles

Passenger.java | pom.xml (test_pyramid_strategy)

```java
package com.tnttest.test_pyramid_strategy.airport;

public class Passenger {

    private String identifier;
    private String name;
    private String countryCode;

    public Passenger(String identifier, String name, String countryCode) {
        this.identifier = identifier;
        this.name = name;
        this.countryCode = countryCode;
    }

    public String getIdentifier() { return identifier; }

    public void setIdentifier(String identifier) { this.identifier = identifier; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getCountryCode() { return countryCode; }

    public void setCountryCode(String countryCode) { this.countryCode = countryCode; }
```

TODO | 6: Problems | Terminal | Build | Event Log

48:1 CRLF UTF-8 Tab*

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help test_pyramid_strategy - Flight.java [test_pyramid_strategy]

Before 〉 airport 〉 src 〉 main 〉 java 〉 com 〉 tnttest 〉 test_pyramid_strategy 〉 airport 〉 Flight

Add Configuration...

Project

- Before D:\training\TDD\implementing-test-pyramid-s
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
        - java
          - com.tnttest.test_pyramid_strategy.airp
            - Flight
            - Main
            - Passenger
      - test
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
  - < 1.8 > C:\Program Files (x86)\Java\jdk1.8.0_241
- Scratches and Consoles

Passenger.java | Flight.java | pom.xml (test_pyramid_strategy)

```java
package com.tnttest.test_pyramid_strategy.airport;

public class Flight {

    private String flightNumber;
    private int places;
    private int passengers;
    private String origin;
    private String destination;
    private boolean flying;

    public Flight(String flightNumber, int places) {
        this.flightNumber = flightNumber;
        this.places = places;
        this.passengers = 0;
        this.flying = false;
    }

    public String getFlightNumber() { return flightNumber; }

    public int getPlaces() { return places; }

    public void setPlaces(int places) { this.places = places; }

    public int getPassengers() { return passengers; }

    public String getOrigin() { return origin; }

    public void setOrigin(String origin) { this.origin = origin; }
```

TODO 6: Problems Terminal Build Event Log

7:28 CRLF UTF-8 4 spaces

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help test_pyramid_strategy - Main.java [test_pyramid_strategy]

Before > airport > src > main > java > com > tnttest > test_pyramid_strategy > airport > Main

Project

- Before D:\training\TDD\implementing-test-pyramid-s
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
        - java
          - com.tnttest.test_pyramid_strategy.airp
            - Flight
            - Main
            - Passenger
      - test
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
  - < 1.8 > C:\Program Files (x86)\Java\jdk1.8.0_241
- Scratches and Consoles

Passenger.java | Flight.java | Main.java | pom.xml (test_pyramid_strategy)

```java
package com.tnttest.test_pyramid_strategy.airport;

public class Main {

    public static void main(String[] args) {
        Passenger passenger = new Passenger( identifier: "123-45-6789",  name: "John Smith",  countryCode: "UK");
        passenger.recordToSystem();

        Flight flight = new Flight( flightNumber: "BA884",  places: 100);
        flight.setOrigin("London");
        flight.setDestination("Bucharest");
        flight.sellTicket();
        flight.takeOff();
        flight.land();
    }
}
```

Run: Main

```
"C:\Program Files (x86)\Java\jdk1.8.0_241\bin\java.exe" ...
Passenger John Smith with identifier: 123-45-6789 from UK has been recorded to the company system
Ticket for Flight BA884 from London to Bucharest sold
Flight BA884 from London to Bucharest is taking off
Flight BA884 from London to Bucharest is landing

Process finished with exit code 0
```

4: Run | TODO | 6: Problems | Terminal | Build | Event Log

Build completed successfully in 6 s 259 ms (30 minutes ago)

3:14 CRLF UTF-8 4 spaces

Project

- Phase1 C:\Users\919842\source\TDD
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
      - test
        - java
          - com.delextest.test_p
            - PassengerTest
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
- Scratches and Consoles

pom.xml (test_pyramid_strategy)

```
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>

<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.0.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.0.1</version>
        <scope>test</scope>
    </dependency>
</dependencies>

</project>
```

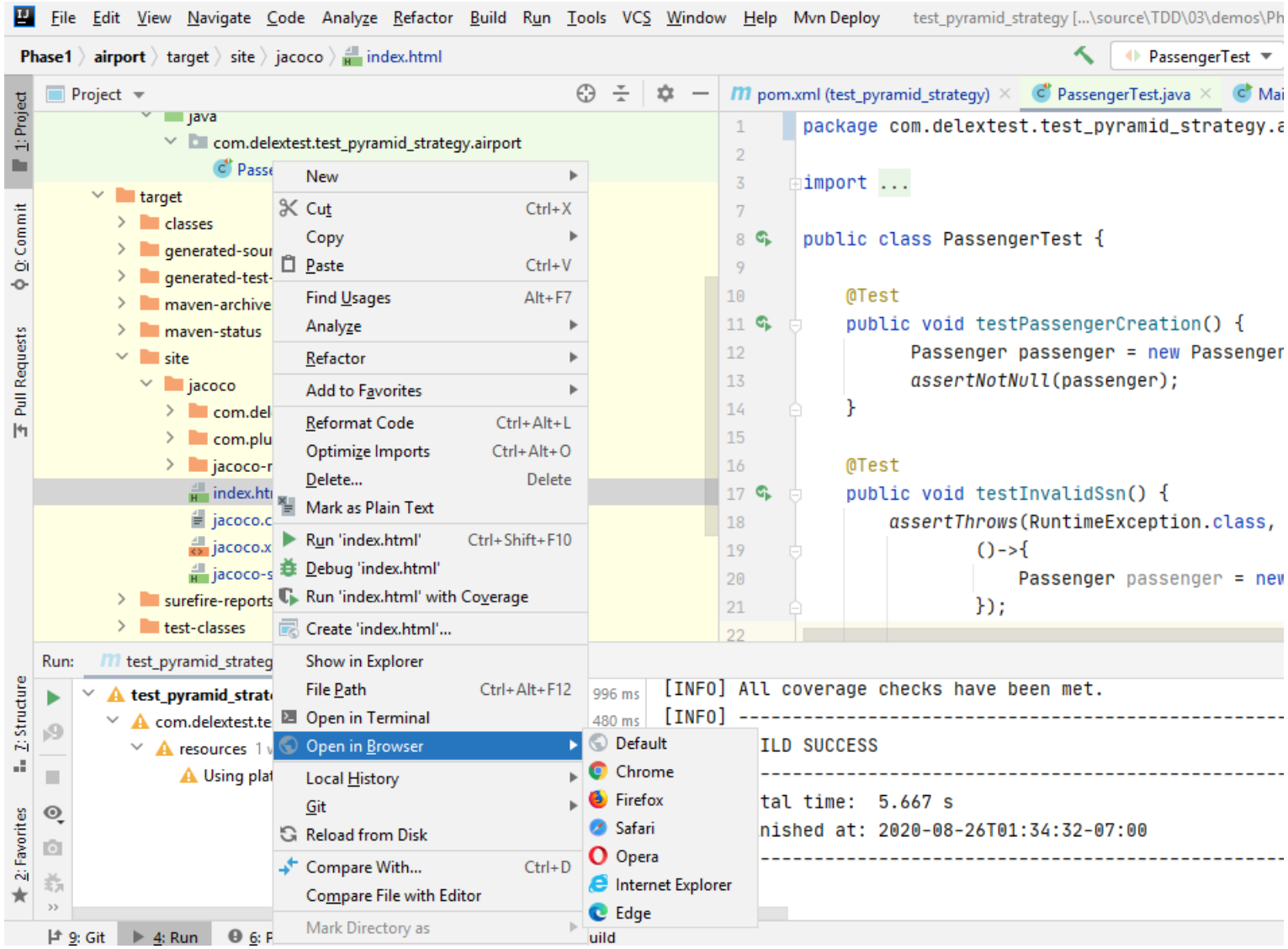
Add JaCoCo to your POM XML

Adding First Test Class

1. Inside the test folder, and a package
2. Inside the package add a test class and give it a name that is descriptive of the test.

Checking Test Coverage

- Open command line and run mvn test from the command line
- Running mvn test will run the jacoco plugin to measure your test coverage.
- Navigate to target > site > jacoco > index.html to view your test coverage



JaCoCo Coverage Report

- After running the tests that we have added, we get following coverage reports.

localhost:63342/test_pyramid_strategy/target/site/jacoco/index.html

test_pyramid_strategy

test_pyramid_strategy

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.delextest.test_pyramid_strategy.airport		14%		100%	24	26	50	60	24	25	2	3
Total	195 of 228	14%	0 of 2	100%	24	26	50	60	24	25	2	3

test_pyramid_strategy > com.delextest.test_pyramid_strategy.airport

com.delextest.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		38%		100%	8	10	12	22	8	9	0	1
Main		0%		n/a	2	2	10	10	2	2	1	1
Total	195 of 228	14%	0 of 2	100%	24	26	50	60	24	25	2	3

JaCoCo Method Coverage Report

Since we only added tests for a passenger class, we get 100% coverage for that class

test_pyramid_strategy > com.delextest.test_pyramid_strategy.airport > Passenger

Passenger

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
toString()		0%		n/a	1	1	1	1	1	1
recordToSystem()		0%		n/a	1	1	2	2	1	1
setIdentifier(String)		0%		n/a	1	1	2	2	1	1
setName(String)		0%		n/a	1	1	2	2	1	1
setCountryCode(String)		0%		n/a	1	1	2	2	1	1
getIdentifier()		0%		n/a	1	1	1	1	1	1
getName()		0%		n/a	1	1	1	1	1	1
getCountryCode()		0%		n/a	1	1	1	1	1	1
Passenger(String, String, String)		100%		100%	0	2	0	10	0	1
Total	52 of 85	38%	0 of 2	100%	8	10	12	22	8	9

Adding more tests and checking coverage

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

Test Invalid ID

Start by writing a test that checks and ID for a US passenger:

```
generated
generated
maven-ar
maven-st
site
jacoco
co
co
jac
inc
16      @Test
17      public void testInvalidSsn() {
18          assertThrows(RuntimeException.class,
19                  ()->{
20                      Passenger passenger = new Passenger( identifier: "123-456-789",  name: "John Smith",  countryCode: "US");
21                  });
22
23
24      }
```

Add the new business rules to the passenger class

Test Invalid ID

Start by writing a test that checks and ID for a US passenger:

Add the new business rules to the passenger class

```
public class Passenger {

    private String identifier;
    private String name;
    private String countryCode;
    private String ssnRegex = "^(?!000|666)[0-8][0-9]{2}-(?!00)[0-9]{2}-(?!0000)[0-9]{4}$";
    private Pattern pattern = Pattern.compile(ssnRegex);

    public Passenger(String identifier, String name, String countryCode) {
        Matcher matcher = pattern.matcher(identifier);
        if(!matcher.matches()) {
            throw new RuntimeException("Invalid identifier");
        }
        this.identifier = identifier;
        this.name = name;
        this.countryCode = countryCode;
    }
```

Run JaCoCo to check the coverage

test_pyramid_strategy

test_pyramid_strategy

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.pluralsight.test_pyramid_strategy.airport		45%		100%	16	29	38	67	16	25	2	3
Total	143 of 261	45%	0 of 8	100%	16	29	38	67	16	25	2	3

Created with Ja

The coverage has increased to 41% on the class level

Increasing the coverage by adding a country code check

```java
@Test
public void testInvalidCountryCode() {
    assertThrows(RuntimeException.class,
            ()->{
                Passenger passenger = new Passenger( identifier: "123-45-6789",  name: "John Smith",  countryCode: "GJ");
            });
}
```

Add and run the test. The test should fail.

```java
public Passenger(String identifier, String name, String countryCode) {
    Matcher matcher = pattern.matcher(identifier);
    if(!matcher.matches()) {
        throw new RuntimeException("Invalid identifier");
    }

    if(!Arrays.asList(Locale.getISOCountries()).contains(countryCode)) {
        throw new RuntimeException("Invalid country code");
    }
```

Unit Testing for the Flight Class

- Add a new test class to test the Passenger class
- Write your first test testFlightCreateion()
- Run the test
- Check the Coverage Report

Integration Testing

After testing each component separately, in order to achieve better code coverage, and ensure confidence, we need to create some unit tests.

Integration testing is a level of software testing where individual units are combined and tested as a group.

Check the interaction between integrated units.

The fact that unit tests work fine in isolation does not necessarily mean that they also work fine together.

Integration testing is performed to expose effects in interfaces and in interactions between components.

Type Of Integration Testing

Black Box Testing

- also known as behavioral testing,
- Internal Implementation of an item being tested is not known
- Therefore the name

Type Of Integration Testing

White Box Testing

- White box testing, also known as open box testing,
- Code-based testing or structural testing
- Choose inputs, execution path, and outputs
- Programming know how is required

Type Of Integration Testing

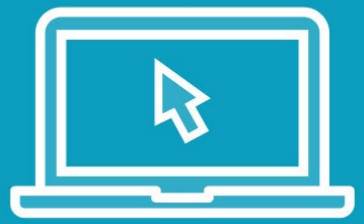
Grey Box Testing

software testing method which is a combination of black box testing method and white box testing method

System Testing

- System testing means to test the complete and integrated software.
- In order to evaluate the system's compliance with the specified requirements. Objective is
  to detect inconsistencies between units that are integrated together.
- Common cases of using a test double/mocking/virtualizaiton are when you are communicating with an external service or with an internal one, but which is n ot yet available. This kind of service may be maintained by the different team. T hey may be slow or difficult to access or it may take some time until becoming f ully available.

Acceptance Testing – Compliance with Business



Did we satisfy customer requests?

What gives value to the software being developed?

What adds business value to the application?

Motivation of Automated Acceptance Testing

- A system is tested for acceptability
- Evaluate the system's compliance with the business requirements.
- Confirm that the software is ready to be made available to the end users.

What Gives Software Business Value

- Reducing Uncertainty in business requirements
- Working Features
- Feature is a tangible deliverable piece of functionality that helps the business achieve their business goals.

- One of the business goals may be to satisfy customers by providing a simple and convenient way for them to manage their flights
- Some features that might help achieve this goal could be choose a flight, change a flight, or choose a seat on the plane.

Communication within a Product Team

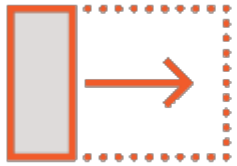
Customer

Business Analyst

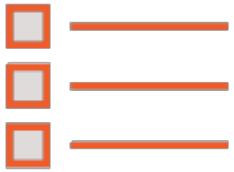
Developer

Tester

Introducing a New Feature



Features – high level requirements



Break the feature into stories

Acceptance criteria

Formulated User Story Example

Given the flights operated by company X

When I want to find the quickest route from Bucharest to New York on May 15 20...

Then I will be provided the route Bucharest - Frankfurt - New York, with a duration of...

From User Story Analysis to Acceptance Criteria

Clear understanding about what the project will need to deliver

Main goals of the application

Example: Increase sales by providing higher quality overall flight services

Requirements:

- Choose Flights
- Change Flights
- Find an optimal route

Describing Requirements

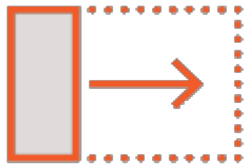
As a passenger

- I want to know the flights for a given destination within a given period of time
- So that I can choose the flight(s) that suit(s) my needs

As a passenger

- I want to be able to change my initial flight(s) to different one(s)
- So that I can follow the changes of my schedule

Breaking Features down into Stories



Each story - a different facet of the problem

Feedback on the functionality

Too large to be implemented at once

Breaking Features down into Stories

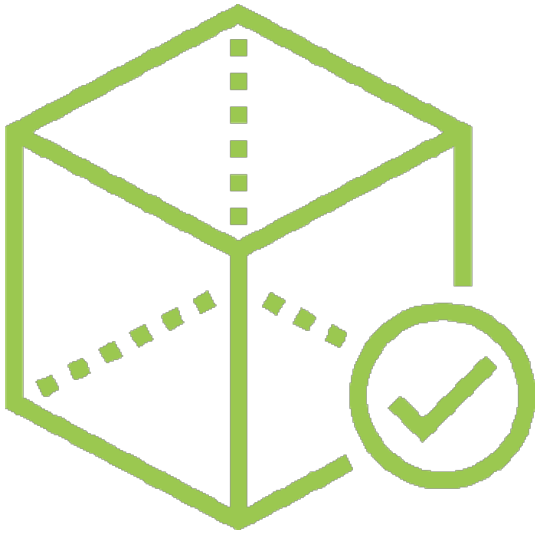
Find direct flights

Find alternative routes with connections

Find One Way Flights

Find Return Flights

Acceptance Criteria



Definition: Given/When/Then

- Given <a context>
- When <something happens>
- Then <some result is expected>

Given the flights operated by the company

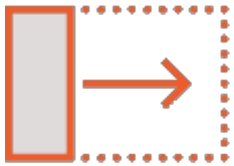
When I want to travel from Bucharest to London next Wednesday

Then I should be provided 2 possible flights: 10:35 and 16:20

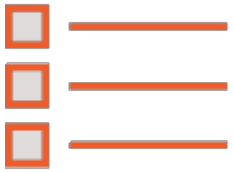
Cucumber, Acceptance Testing Framework

What is cucumber and what is it good for?

Automated Acceptance Tests



Originates from XP and TDD



Different from unit tests, focus on behavior

Express what the software needs to do

Automated Acceptance Test Example

Feature: Removing and Adding a Passenger

Process of adding and removing passengers depends on type of reservation (passenger type)

Scenario:

Given A flight is available

When we have an economy reservation

Then you can add remove a passenger from the flight

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Mvn Deploy test_pyramid_strategy [...\source\TDD\06\demos\Phase1] - pom.xml (test_pyramid_strategy)

Phase1 > airport > pom.xml

PassengerTest

Project

- annotations
- producers
- BonusPolicy
- CucumberTest
- FlightBuilderUtil
- FlightTest
- FlightWithPassengersTest
- PassengerTest
- resources
  - features
    - bonus_policy.feature
  - flights_information.csv
  - flights_information2.csv
  - flights_information3.csv
- target
  - classes
  - generated-sources
    - annotations
  - generated-test-sources
    - test-annotations
  - maven-status
  - test-classes
  - jacoco.exec
- pom.xml
- test_pyramid_strategy.iml
- test_pyramid_strategy.ipr
- test_pyramid_strategy.iws
- External Libraries
- Scratches and Consoles

Policy.java | PassengerTest.java | CucumberTest.java | FlightTest.java | FlightWithPassengersTest.java | pom.xml (test_pyramid_strategy)

```
107            <artifactId>arquillian-weld-ee-embedded-1.1</artifactId>
108            <version>1.0.0.CR9</version>
109            <scope>test</scope>
110        </dependency>
111        <dependency>
112            <groupId>org.jboss.weld</groupId>
113            <artifactId>weld-core</artifactId>
114            <version>2.3.5.Final</version>
115            <scope>test</scope>
116        </dependency>
117        <dependency>
118            <groupId>info.cukes</groupId>
119            <artifactId>cucumber-java</artifactId>
120            <version>1.2.5</version>
121            <scope>test</scope>
122        </dependency>
123        <dependency>
124            <groupId>info.cukes</groupId>
125            <artifactId>cucumber-junit</artifactId>
126            <version>1.2.5</version>
127            <scope>test</scope>
128        </dependency>
129    </dependencies>
130
131 </project>
```

project > build > plugins

Text | Dependency Analyzer

IntelliJ IDEA 2020.2.1 available
Update...

6: Problems | TODO | Terminal | Java Enterprise | Build

IntelliJ IDEA 2020.2.1 available // Update... (a minute ago)

13:18 LF UTF-8 Tab*

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Mvn Deploy test_pyramid_strategy [...\source\TDD\06\demos\Phase1] - pom.xml (test_pyramid_strategy)

Phase1 airport PassengerTest

Project

- Phase1 C:\Users\919842\source\TDD\06\demos\Phase1
  - .idea
  - airport [test_pyramid_strategy]
    - .idea
    - src
      - main
      - test
        - java
        - resources
          - features
            - flights_information.csv
            - flights_information2.csv
            - flights_information3.csv
    - target
    - pom.xml
    - test_pyramid_strategy.iml
    - test_pyramid_strategy.ipr
    - test_pyramid_strategy.iws
- External Libraries
- Scratches and Consoles

est.java | CucumberTest.java | FlightTest.java | FlightWithPassengersTest.java | pom.xml (test_pyramid_strategy)

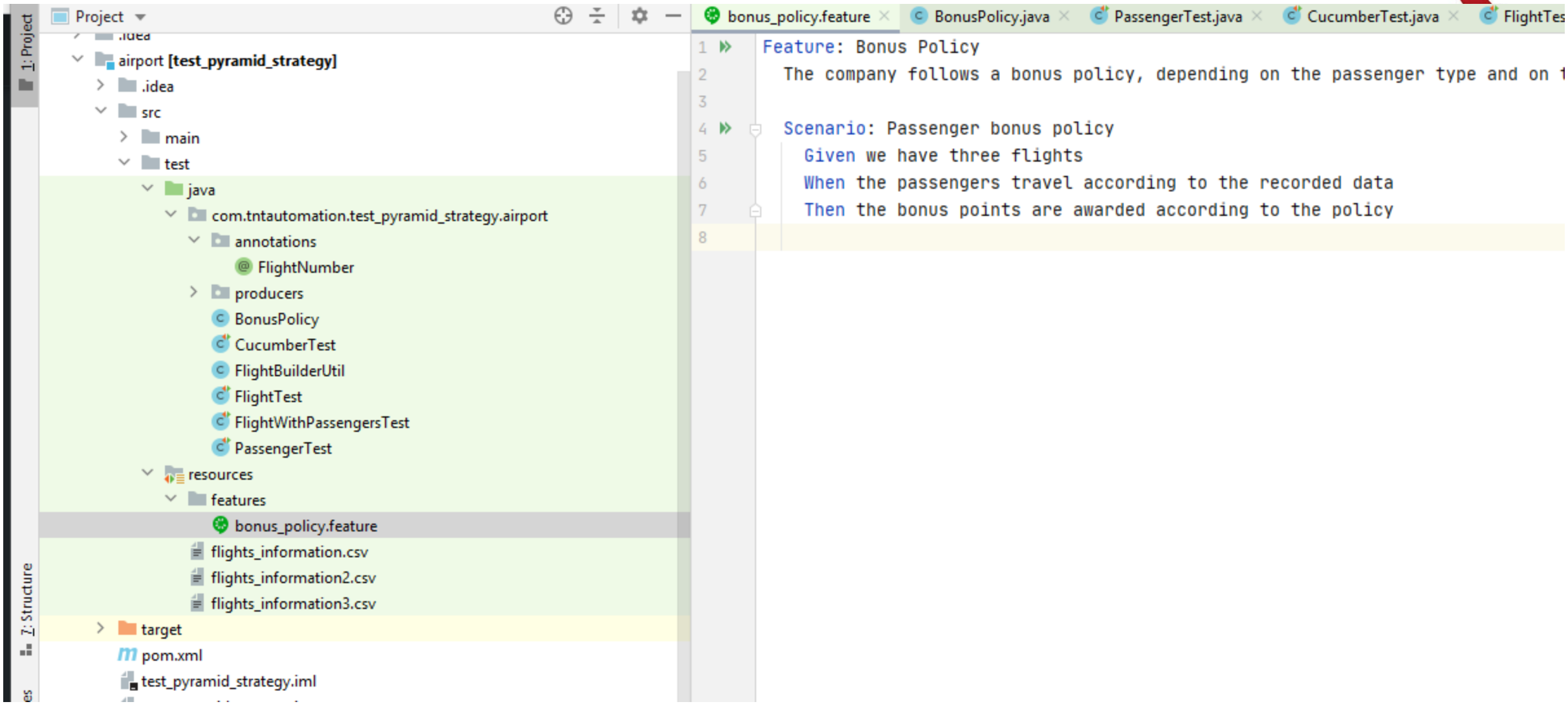
```
107            <artifactId>arquillian-weld-ee-embedded-1.1</artifactId>
108            <version>1.0.0.CR9</version>
109            <scope>test</scope>
110        </dependency>
111        <dependency>
112            <groupId>org.jboss.weld</groupId>
113            <artifactId>weld-core</artifactId>
114            <version>2.3.5.Final</version>
115            <scope>test</scope>
116        </dependency>
117        <dependency>
118            <groupId>info.cukes</groupId>
119            <artifactId>cucumber-java</artifactId>
120            <version>1.2.5</version>
121            <scope>test</scope>
122        </dependency>
123        <dependency>
124            <groupId>info.cukes</groupId>
125            <artifactId>cucumber-junit</artifactId>
126            <version>1.2.5</version>
127            <scope>test</scope>
128        </dependency>
129    </dependencies>
130
131 </project>
```

project › build › plugins

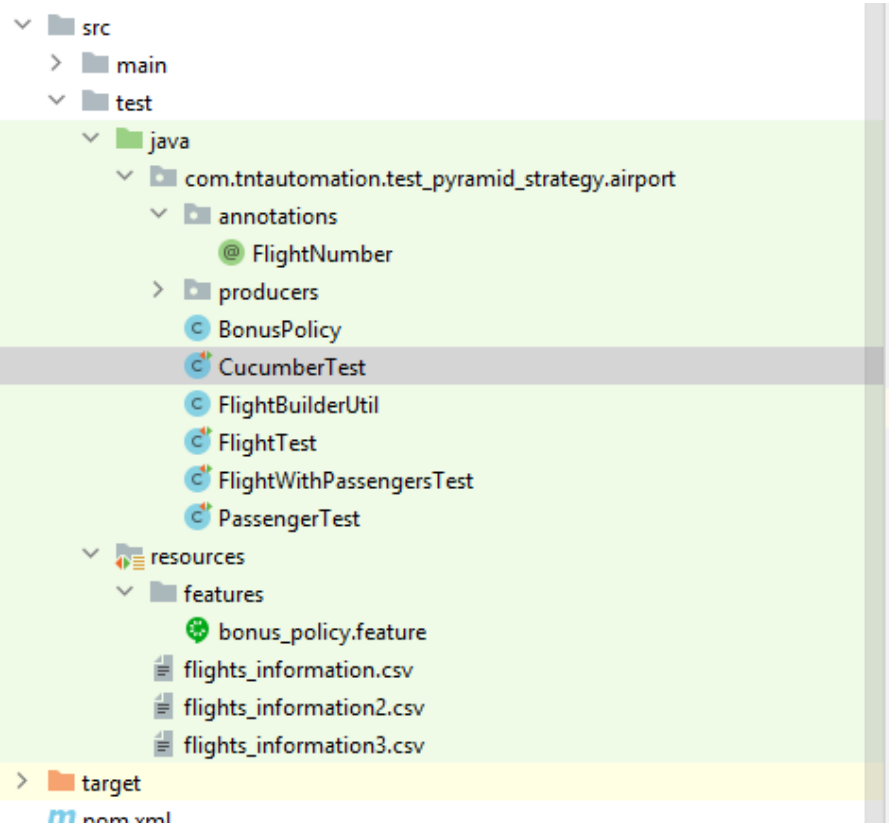
Text Dependency Analyzer

IntelliJ IDEA 2020.2.1 available
Update...

6: Problems TODO Terminal Java Enterprise Build

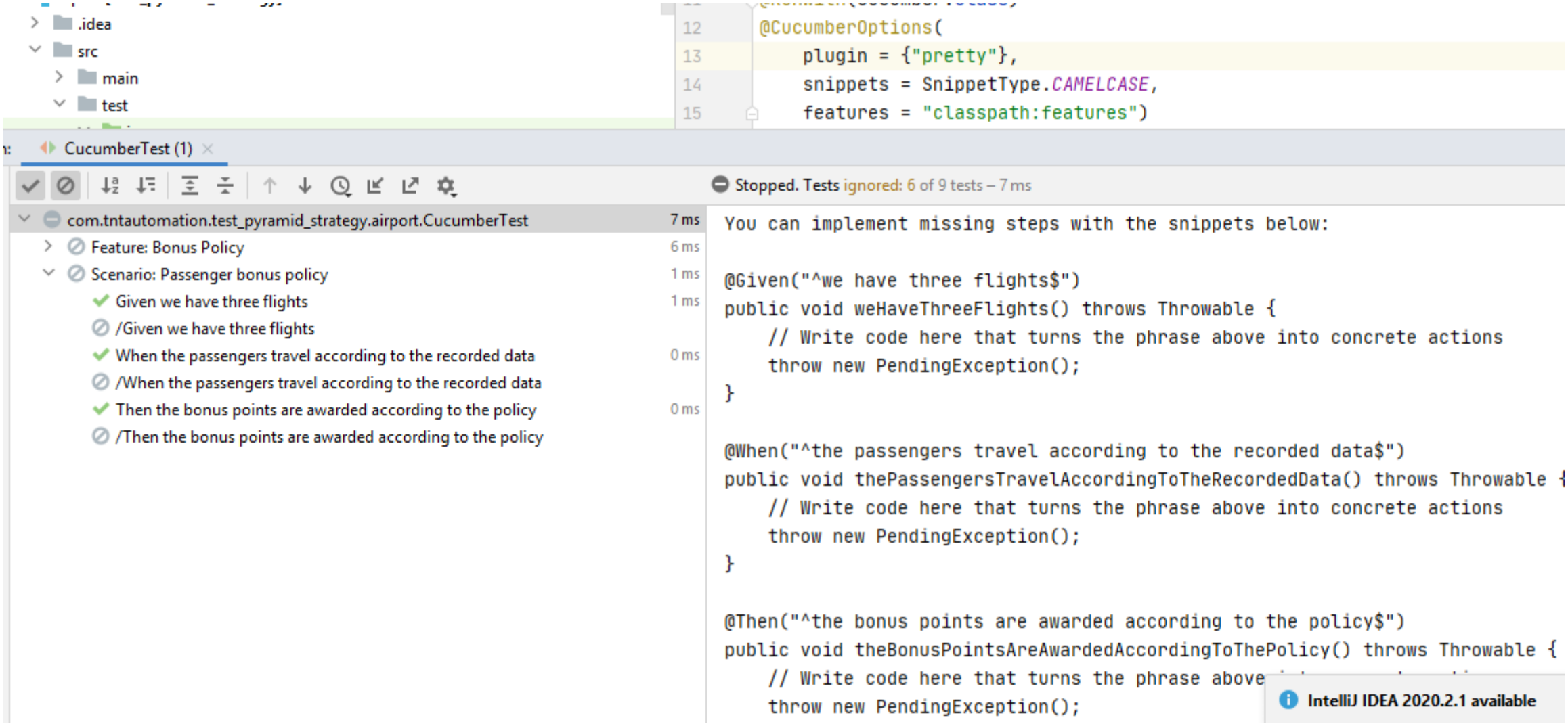

Project
airport [test_pyramid_strategy]
.idea
src
main
test
java
com.tntautomation.test_pyramid_strategy.airport
annotations
FlightNumber
producers
BonusPolicy
CucumberTest
FlightBuilderUtil
FlightTest
FlightWithPassengersTest
PassengerTest
resources
features
bonus_policy.feature
flights_information.csv
flights_information2.csv
flights_information3.csv
target
pom.xml
test_pyramid_strategy.iml
bonus_policy.feature
BonusPolicy.java
PassengerTest.java
CucumberTest.java
Feature: Bonus Policy
The company follows a bonus policy, depending on the passenger type and on
Scenario: Passenger bonus policy
Given we have three flights
When the passengers travel according to the recorded data
Then the bonus points are awarded according to the policy

Adding a test runner

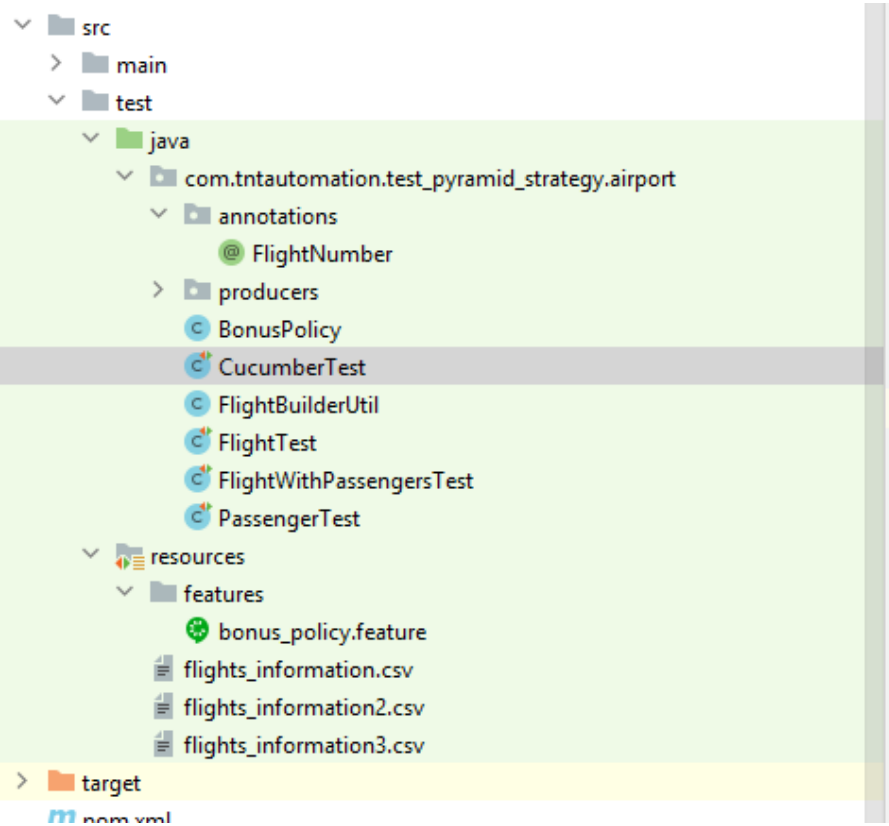



```java
import ...

/**
 * Entry point for running the Cucumber tests in JUnit.
 */
@RunWith(Cucumber.class)
@CucumberOptions(
    plugin = {"pretty"},
    snippets = SnippetType.CAMELCASE,
    features = "classpath:features")
public class CucumberTest {

    /**
     * This class should be empty, step definitions should be in separate classes.
     */

}
```

Running feature files – not implemented.



Test Runner Overview




```
3   import ...
7
8   /**
9    * Entry point for running the Cucumber tests in JUnit.
10   */
11  @RunWith(Cucumber.class)
12  @CucumberOptions(
13      plugin = {"pretty"},
14      snippets = SnippetType.CAMELCASE,
15      features = "classpath:features")
16  public class CucumberTest {
17
18      /**
19       * This class should be empty, step definitions should be in separate classes.
20       */
21
22  }
23
```

Discover more Cucumber capabilities

```
Feature: Bonus Policy
  The company follows a bonus policy, depending on the passenger type and on the distance that has been traveled

  Scenario Outline: Passenger bonus policy
    Given we have the flights defined into "<file1>" and "<file2>" and "<file3>"
    When the passengers travel according to the data recorded into these files
    Then passenger with identifier "<identifier>" name "<name>" and countryCode "<countryCode>" has been awarded <poin

    Examples:
      | file1                    | file2                    | file3                    | identifier
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 900-45-6809
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 900-45-6797
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 123-45-6799
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 123-45-6789
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 900-45-6789
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 123-45-6790
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 900-45-6790
```

Discover more Cucumber capabilities

```
Feature: Bonus Policy
  The company follows a bonus policy, depending on the passenger type and on the distance that has been traveled

  Scenario Outline: Passenger bonus policy
    Given we have the flights defined into "<file1>" and "<file2>" and "<file3>"
    When the passengers travel according to the data recorded into these files
    Then passenger with identifier "<identifier>" name "<name>" and countryCode "<countryCode>" has been awarded <poin

    Examples:
      | file1                    | file2                    | file3                    | identifier
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 900-45-6809
      | flights_information.csv  | flights_information2.csv | flights_information3.csv | 900-45-6797
      | flights_information.csv  | flights_information2.csv |
      | flights_information.csv  | flights_information2.csv |
      | flights_information.csv  | flights_information2.csv |
      | flights_information.csv  | flights_information2.csv |
      | flights_information.csv  | flights_information2.csv |
      | flights_information.csv  | flights_information2.csv |
```

name	countryCode	points
Susan Todd	GB	210
Harry Christensen	GB	420
Bethany King	US	630
John Smith	US	420
Jane Underwood	GB	420
James Perkins	US	630
Mary Calderon	GB	630