



TESTOWANIE OPROGRAMOWANIA | TEST DRIVEN DEVELOPMENT

Autor: Michał Bojanowski
Prawa do korzystania z materiałów posiada Software
Development Academy

JUnit - executables



Czym jest executable?

```
import org.junit.jupiter.api.function.Executable;
```

```
@FunctionalInterface
@API(status = STABLE, since = "5.0")
public interface Executable {
    void execute() throws Throwable;
}
```

```
public class ExecutableExample implements Executable {
    @Override
    public void execute() {
        assertEquals(2, 3);
    }
}
```

JUnit – inny sposób użycia



```
@Test
void testThatDoesNothingButShowsHowToUseExecutables() {
    final Executable checkOneExecutable = () -> assertEquals(1, 1);
    final Executable checkTwoExecutable = () -> assertTrue(4 % 3 == 1);

    assertAll(checkOneExecutable,
              checkTwoExecutable,
              new ExecutableExample());
}
```

Java – bloki statyczne



```
public class StaticBlocks {

    private static List<String> staticList;
    private List<String> notStaticList;

    static {
        // to mogę zrobić bo staticList jest statyczne
        staticList = Arrays.asList("STR1", "STR2", "STR3");

        // tego nie mogę zrobić. Tu mogę zmieniać jedynie zmienne statyczne
        // notStaticList = new ArrayList<>();
        System.out.println("BLOK STATYCZNY");
    }

    public StaticBlocks() {
        // tu mogę edytować zmienne statyczne i niestatyczne
        notStaticList = new ArrayList<>();
        staticList.contains("STR1");
        System.out.println("KONSTRUKTOR");
    }
}
```

BLOK STATYCZNY
KONSTRUKTOR



Czym jest mapa?

- Kontener przechowujący pary. Wszystkie pary to tzw. EntrySet
- Każda para to tzw. Entry
- Każda para, tzn. Entry, składa się z klucza (key) i wartości (value)
- Każdy klucz może pojawić się w mapie tylko raz

Java – mapy, klucze i wartości



```
final Map<Integer, String> indexToMonthName = new HashMap<>();  
indexToMonthName.put(1, "STYCZEN");  
indexToMonthName.put(2, "LUTY");  
indexToMonthName.put(3, "GRUDZIEŃ");
```

```
indexToMonthName.get(1);
```

```
boolean containsKeyTwo = indexToMonthName.containsKey(2);
```

```
boolean containsStyczen = indexToMonthName.containsValue("STYCZEN");
```

Java – mapy, cd.

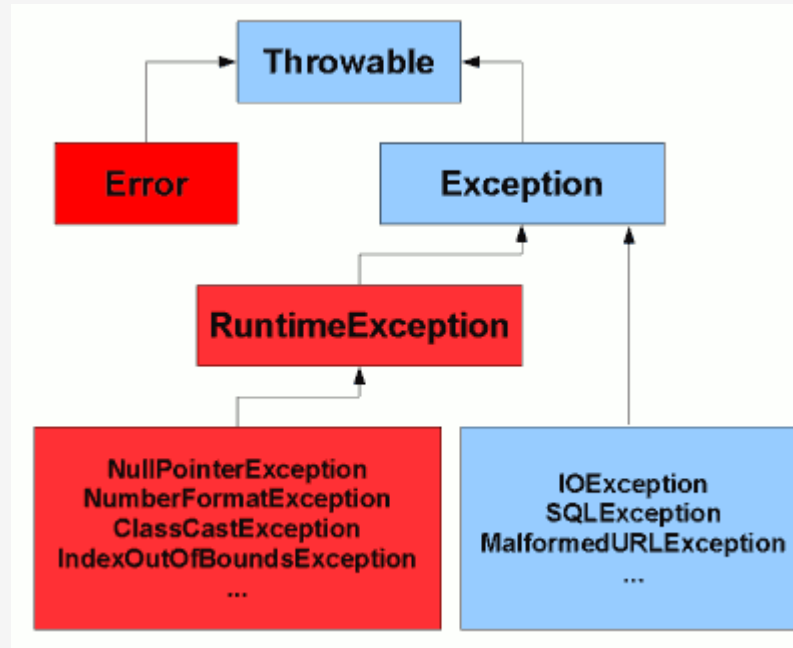


```
indexToMonthName.entrySet().forEach(entry ->  
System.out.println(entry.getKey() + " " + entry.getValue()));
```

```
indexToMonthName.forEach((key, value) -> System.out.println(key + " " + value));
```

```
1 STYCZEN  
2 LUTY  
3 GRUDZIEN
```

Java – wyjątki



Autor: Michał Bojanowski

Prawa do korzystania z materiałów posiada Software Development Academy

Java – wyjątki



Każdy wyjątek posiada:

- Wiadomość wyjątku
- Przyczynę wyjątku (tzn. poprzedni wyjątek, który go wyrzucił)
- StackTrace: stos wywołań metod, który przyczynił się do wyrzucenia wyjątku

```
new Throwable().getMessage();  
new Throwable().getCause();  
new Throwable().getStackTrace();
```

Java – wyjątki



```
public static void showingExceptionWithCause() {  
    try {  
        new FibonacciSeries().compute(-1);  
    } catch (final IllegalArgumentException exp) {  
        System.err.println("Failed to compute fibonacci serie value");  
        throw new FibonacciSeriesException(exp.getMessage(), exp);  
    }  
}
```

Oryginalny exp (IllegalArgumentException) jest POWODEM (tzw. cause) wyrzucenia wyjątku FibonacciSeriesException

Java – cause



```
public static void showingExceptionWithCause() {  
    try {  
        new FibonacciSeries().compute(-1);  
    } catch (final IllegalArgumentException exp) {  
        System.err.println("Failed to compute fibonacci serie value");  
        throw new FibonacciSeriesException(exp.getMessage(), exp);  
    }  
}
```

```
"C:\Program Files (x86)\Java\jdk1.8.0_131\bin\java.exe" ...
```

```
Failed to compute fibonacci serie value
```

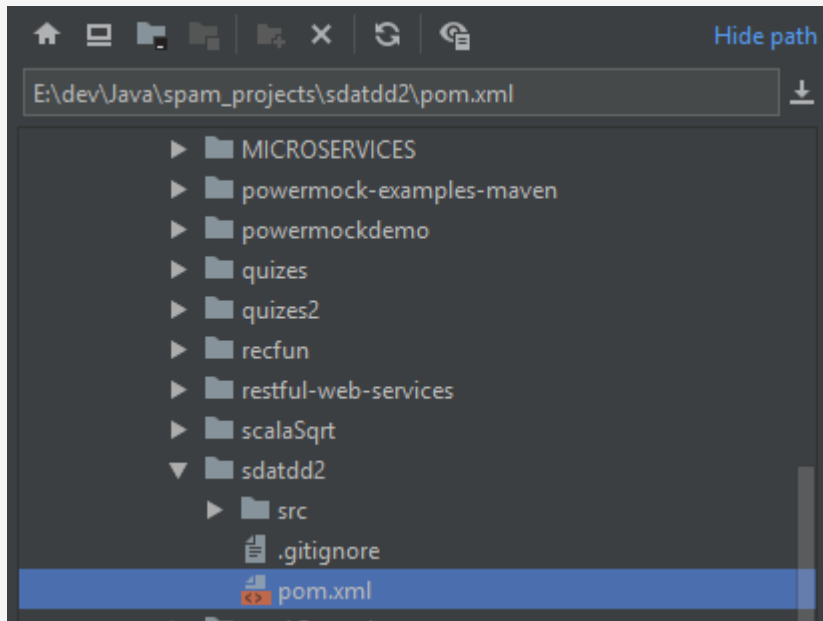
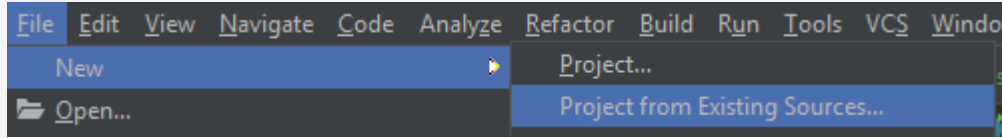
```
pl.sdacademy.exceptions.FibonacciSeriesException: Index has to be positive
```

```
    at pl.sdacademy.ExceptionsExample.showingExceptionWithCause(ExceptionsExample.java:13)  
    at pl.sdacademy.ExceptionsExampleTest.test(ExceptionsExampleTest.java:11) <15 internal calls>  
    at java.util.ArrayList.forEach(ArrayList.java:1249) <5 internal calls>  
    at java.util.ArrayList.forEach(ArrayList.java:1249) <17 internal calls>
```

```
Caused by: java.lang.IllegalArgumentException: Index has to be positive
```

```
    at pl.sdacademy.calculations.FibonacciSeries.compute(FibonacciSeries.java:7)  
    at pl.sdacademy.ExceptionsExample.showingExceptionWithCause(ExceptionsExample.java:10)  
    ... 40 more
```

IDEA – poprawne importowanie projektów



IDEA – poprawne importowanie projektów



Import Project from Maven

Root directory: E:\dev\Java\spam_projects\sdatdd2

☐ Search for projects recursively

Project format: .idea (directory based)

☐ Synchronize Maven project model and IDEA project model each time when pom.xml is changed

☒ Import Maven projects automatically

☒ Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)

☐ Create module groups for multi-module Maven projects

☒ Keep source and test folders on reimport

☒ Exclude build directory (%PROJECT_ROOT%/target)

☒ Use Maven output directories

Generated sources folders: Detect automatically

Phase to be used for folders update: process-resources

IDEA needs to execute one of the listed phases in order to discover all source folders that are configured via Maven plugins.
Note that all test-* phases firstly generate and compile production sources.

Automatically download: ☐ Sources ☐ Documentation

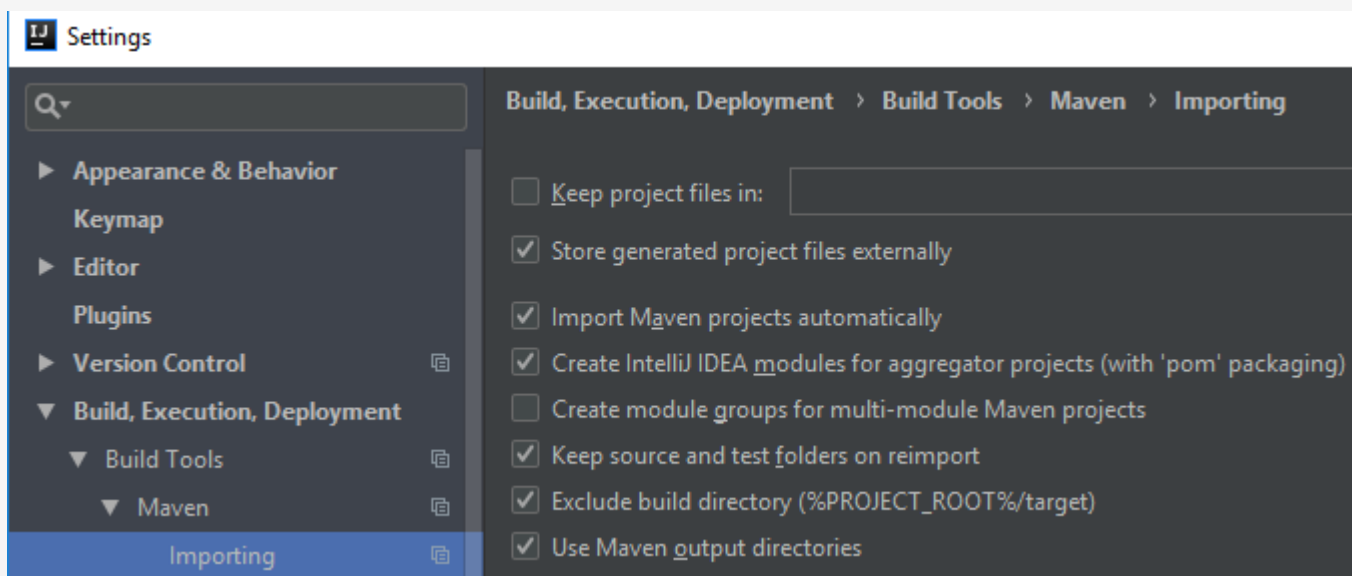
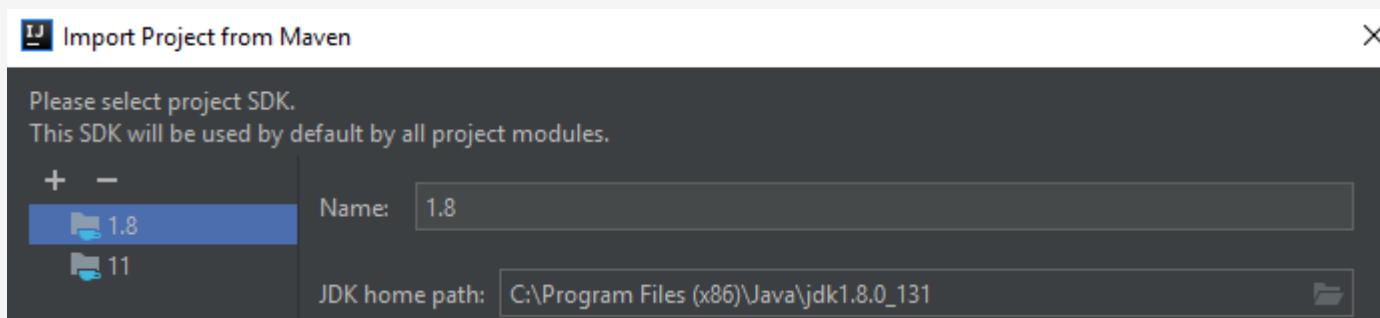
Dependency types: jar, test-jar, maven-plugin, ejb, ejb-client, jboss-har, jboss-sar, war, ear, bundle

Comma separated list of dependency types that should be imported

Environment settings...

Previous Next Cancel Help

IDEA – poprawne importowanie projektów



IDEA – wymuszenie reimportu



CTRL + SHIFT + A

