# Programowanie w Java Wprowadzenie

Łukasz Rybka · Gdańsk 2016/17

# O mnie

# O mnie

## Z WYKSZTAŁCENIA:

# O mnie

## Z WYKSZTAŁCENIA:

**fizyk**

# O mnie

## Z WYKSZTAŁCENIA:

fizyk

## Z ZAWODU:

# O mnie

## Z WYKSZTAŁCENIA:

fizyk

## Z ZAWODU:

**Development Team Leader w firmie Solwit S.A. /**
**kontrybutor Open Source / freelancer**

# O mnie

## Z WYKSZTAŁCENIA:

fizyk

## Z ZAWODU:

Development Team Leader w firmie Solwit S.A. /
kontrybutor Open Source / freelancer

## Z ZAMIŁOWANIA:

# O mnie

## Z WYKSZTAŁCENIA:

fizyk

## Z ZAWODU:

Development Team Leader w firmie Solwit S.A. /
kontrybutor Open Source / freelancer

## Z ZAMIŁOWANIA:

**wykładowca / prelegent / szkoleniowiec**

# Wykład - materiały

## Literatura:

http://dragonia.org.pl/courses/itj

# Wykład - materiały

## Literatura:

**Bentrand Meyer "Programowanie zorientowane obiektowo"**

http://dragonia.org.pl/courses/itj

# Wykład - materiały

## Literatura:

Bentrand Meyer "Programowanie zorientowane obiektowo"

Bruce Eckel "Thinking in Java"

http://dragonia.org.pl/courses/itj

# Wykład - materiały

## Literatura:

Bentrand Meyer "Programowanie zorientowane obiektowo"

Bruce Eckel "Thinking in Java"

Katy Sierra, Bert Bates "OCA/OCP Java SE 7 Programmer I & II Study Guide"

http://dragonia.org.pl/courses/itj

# Ustalenia

# Ustalenia

## Pytania mile widziane!

# Ustalenia

Pytania mile widziane!

**Slajdy to tak naprawdę ściągawka dla wykładowcy ;)**

# Ustalenia

Pytania mile widziane!

Slajdy to tak naprawdę ściągawka dla wykładowcy ;)

**Nie toleruję przeszkadzania…**

# Ustalenia

Pytania mile widziane!

Slajdy to tak naprawdę ściągawka dla wykładowcy ;)

Nie toleruję przeszkadzania…

… a wykład nie jest obowiązkowy!

# Organizacja zajęć

# Organizacja zajęć

## 15 godzin wykładu

# Organizacja zajęć

15 godzin wykładu

15 godzin laboratoriów

# Organizacja zajęć

15 godzin wykładu

15 godzin laboratoriów

**Zaliczenie przedmiotu: informacja na kolejnych zajęciach**

# Konsultacje

# Konsultacje

## Po wcześniejszej informacji (np. mailowo)

# Konsultacje

## Po wcześniejszej informacji (np. mailowo)

## Sala 415 Gmachu B

# Zakres wykładów

# Zakres wykładów

## Wprowadzenie do ekosystemu Java/JVM

# Zakres wykładów

## Wprowadzenie do ekosystemu Java/JVM

## Apache Maven i struktura projektu

# Zakres wykładów

Wprowadzenie do ekosystemu Java/JVM

Apache Maven i struktura projektu

**Klasy i obiekty**

# Zakres wykładów

Wprowadzenie do ekosystemu Java/JVM

Apache Maven i struktura projektu

Klasy i obiekty

Dziedziczenie

# Zakres wykładów

Wprowadzenie do ekosystemu Java/JVM

Apache Maven i struktura projektu

Klasy i obiekty

Dziedziczenie

**Polimorfizm**

# Zakres wykładów

Wprowadzenie do ekosystemu Java/JVM

Apache Maven i struktura projektu

Klasy i obiekty

Dziedziczenie

Polimorfizm

**Kolekcje**

# Zakres wykładów

Wprowadzenie do ekosystemu Java/JVM

Apache Maven i struktura projektu

Klasy i obiekty

Dziedziczenie

Polimorfizm

Kolekcje

**Obsługa wyjątków**

# Czego będziemy używać?

# Czego będziemy używać?

## Java Standard Edition Development Kit (Java SE JDK) - 8u60

# Czego będziemy używać?

## Java Standard Edition Development Kit (Java SE JDK) - 8u60

## Eclipse Neon for Java Developers

# Po co to wszystko?

| Sep 2015 | Sep 2014 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 2 | ↑ | Java | 19.565% | +5.43% |
| 2 | 1 | ↓ | C | 15.621% | -1.10% |
| 3 | 4 | ↑ | C++ | 6.782% | +2.11% |
| 4 | 5 | ↑ | C# | 4.909% | +0.56% |
| 5 | 8 | ↑ | Python | 3.664% | +0.88% |
| 6 | 7 | ↑ | PHP | 2.530% | -0.59% |
| 7 | 9 | ↑ | JavaScript | 2.342% | -0.11% |
| 8 | 11 | ↑ | Visual Basic .NET | 2.062% | +0.53% |
| 9 | 12 | ↑ | Perl | 1.899% | +0.53% |
| 10 | 3 | ⇊ | Objective-C | 1.821% | -8.11% |

# Co to jest JVM?

# Co to jest JVM?

## JAVA VIRTUAL MACHINE:

# Co to jest JVM?

## JAVA VIRTUAL MACHINE:

**środowisko uruchomieniowe**

# Co to jest JVM?

## JAVA VIRTUAL MACHINE:

środowisko uruchomieniowe

## JAVA BYTECODE:

# Co to jest JVM?

## JAVA VIRTUAL MACHINE:
środowisko uruchomieniowe

## JAVA BYTECODE:
wiele języków kompilowanych do jednego kodu "maszynowego"

# Co to jest JVM?

## JAVA VIRTUAL MACHINE:
środowisko uruchomieniowe

## JAVA BYTECODE:
wiele języków kompilowanych do jednego kodu "maszynowego"

## GARBAGE COLLECTOR:

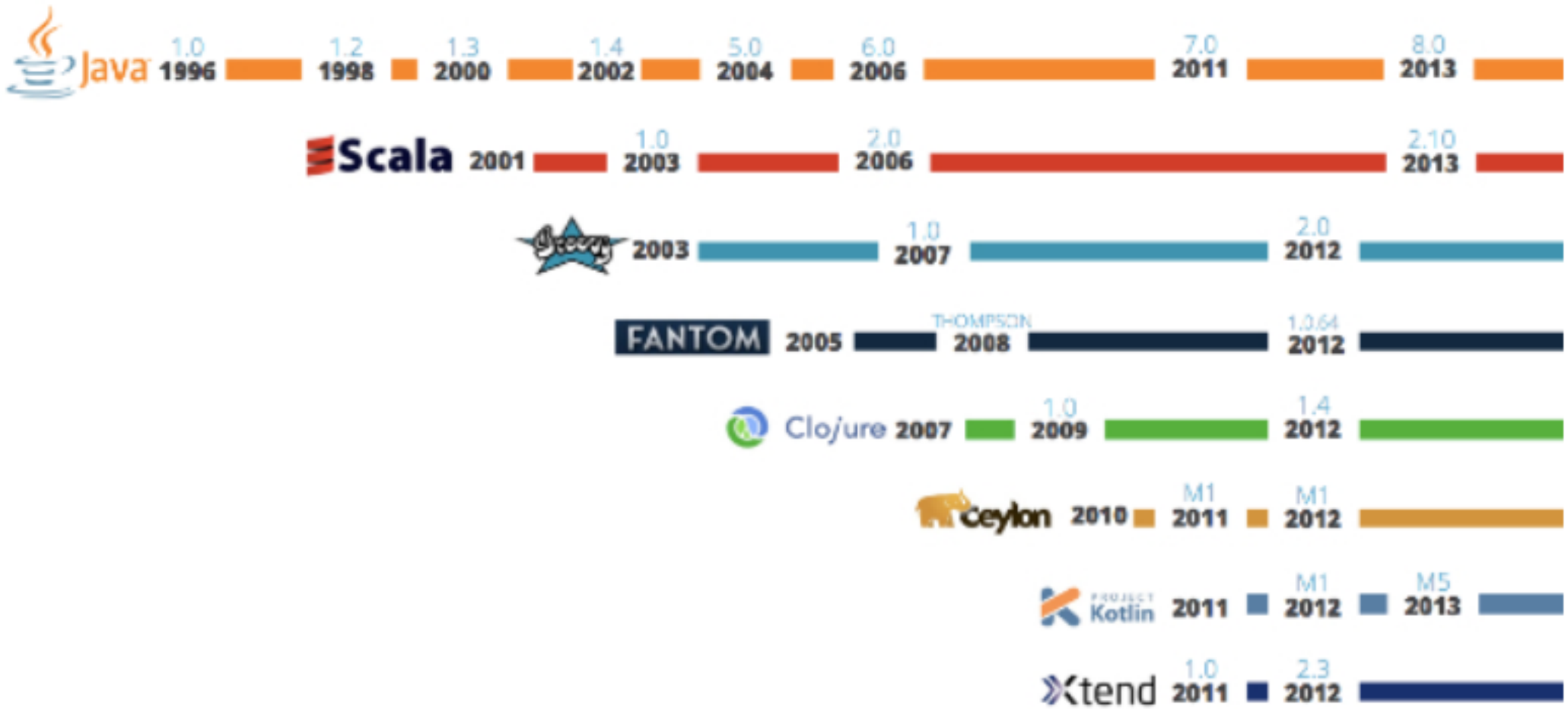
# Co to jest JVM?

## JAVA VIRTUAL MACHINE:
środowisko uruchomieniowe

## JAVA BYTECODE:
wiele języków kompilowanych do jednego kodu "maszynowego"

## GARBAGE COLLECTOR:
zarządzanie pamięcią

# JVM languages



# Najprostszy program Java (Main.java)

```
public class Main {(1)

    public static void main(String[] args) {(2)

        System.out.println("Hello World!");(3)

    }
}
```

(1) Definicja głównej (i jedynej) klasy w aplikacji

(2) Definicja metody main, do której zostanie przekazane sterowanie aplikacją przy jej uruchomieniu

(3) Instrukcja wyświetlenia na standardowe wyjście systemowe (konsolę) napisu

```
javac Main.java(1)

java Main(2)
```

(1) Kompilacja klasy (plik .java) do bytecode (plik .class)

(2) Uruchomienie skompilowanej klasy (programu)

# Bytecode

```
Compiled from "Main.java"❶
public class Main {
  public Main();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."
<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2                  // Field
java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #3                  // String Hello World!
       5: invokevirtual #4                  // Method
java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

❶ javap -c Main

# Kilka podstawowych informacji

# Kilka podstawowych informacji

## Plik ma tę samą nazwę co klasa publiczna

# Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa publiczna

**Uruchamiamy program przez uruchomienie klasy z funkcją main(String[] args)**

# Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa publiczna

Uruchamiamy program przez uruchomienie klasy z funkcją main(String[] args)

# Pakiety - "nie wszystko w jednym worku"!

# Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa publiczna

Uruchamiamy program przez uruchomienie klasy z funkcją main(String[] args)

Pakiety - "nie wszystko w jednym worku"!

**Dokumentujemy kod w kodzie (javadoc)**

# Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa publiczna

Uruchamiamy program przez uruchomienie klasy z funkcją main(String[] args)

Pakiety - "nie wszystko w jednym worku"!

Dokumentujemy kod w kodzie (javadoc)

**Podział programu na klasy według (ich) odpowiedzialności**

# Klasa pomocnicza (DateUtils.java)

```
package pl.org.dragonia.helloapp.utils;


import java.util.Calendar;

public class DateUtils {
    private Calendar calendar;

    public DateUtils() {
        this.calendar = Calendar.getInstance();
    }

    public String sayHello() {
        String message = "Hello, today is ";

        message += calendar.get(Calendar.DAY_OF_YEAR);
        message += " day of year!";

        return message;
    }
}
```

# Klasa pomocnicza (Main.java)

```
package pl.org.dragonia.helloapp;

import pl.org.dragonia.helloapp.utils.DateUtils;

public class Main {

    public static void main(String[] args) {
        DateUtils dateUtils = new DateUtils();

        System.out.println(dateUtils.sayHello());
    }
}
```

# Dokumentacja kodu (Javadoc)

```java
/**
 * Method saying hello and which day of year is today.
 *
 * @return {String} string telling which day of year is today
 */
public String sayHello() {
    String message = "Hello, today is ";

    int dayOfYear = calendar.get(Calendar.DAY_OF_YEAR);
    switch (dayOfYear) {
        case 1:
            message += dayOfYear + "st";
        break;
        // ...
        default:
            message += dayOfYear + "th";
        break;
    }

    message += " day of year!";

    return message;
}
```

# Apache Maven

# Apache Maven

## Project Management Tool

# Apache Maven

## Project Management Tool

**Wspomaga budowanie (kompilację, linkowanie, …), testowanie, raportowanie, dokumentację i wiele więcej**

# Apache Maven

## Project Management Tool

## Wspomaga budowanie (kompilację, linkowanie, …), testowanie, raportowanie, dokumentację i wiele więcej

## Zarządza zależnościami projektu

# Apache Maven

## Project Management Tool

## Wspomaga budowanie (kompilację, linkowanie, …), testowanie, raportowanie, dokumentację i wiele więcej

## Zarządza zależnościami projektu

## Pozwala na zarządzanie projektami o skomplikowanej strukturze

# Konfiguracja projektu

# Konfiguracja projektu

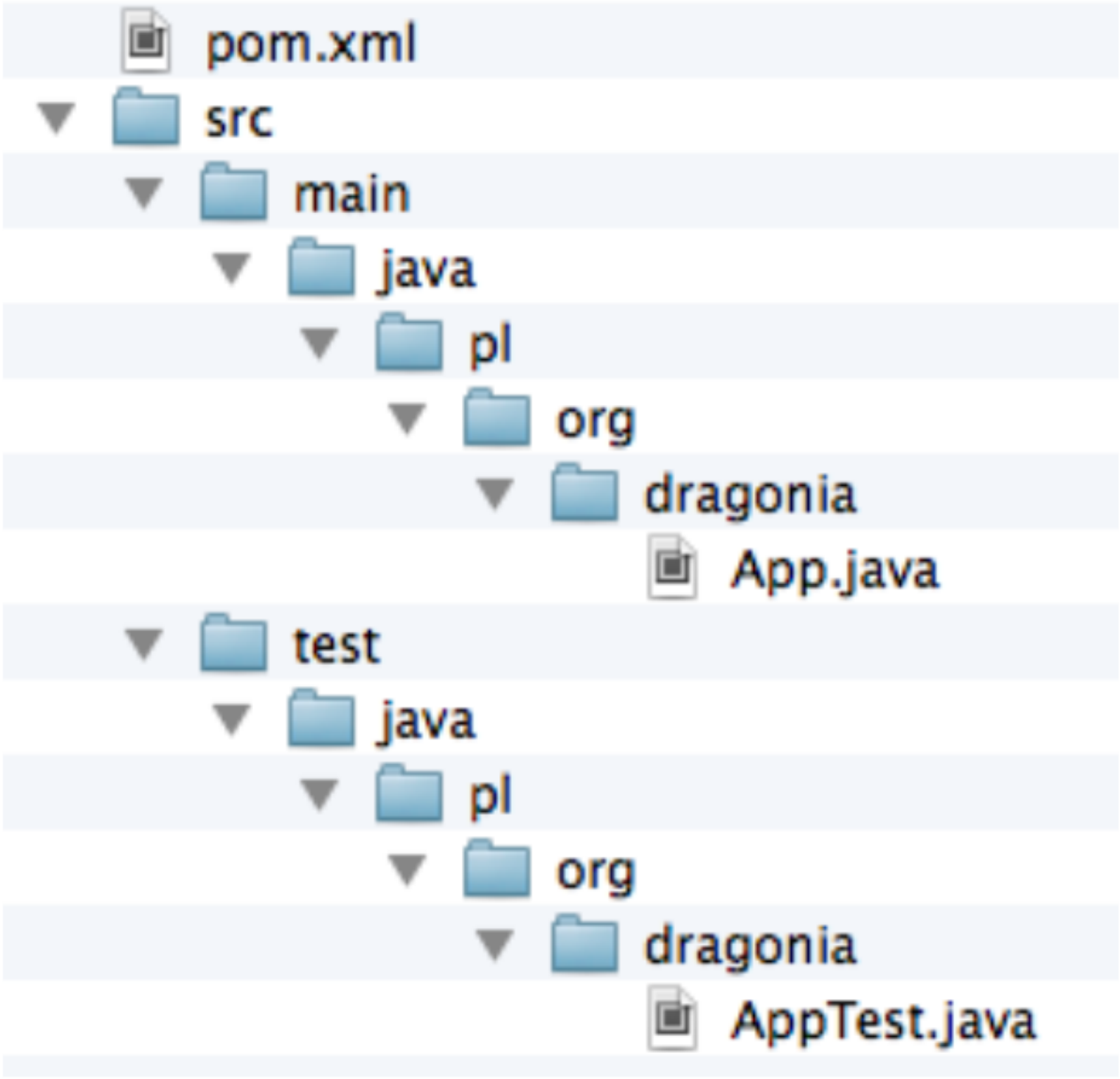
## Centralne miejsce informacji i konfiguracji projektu - plik POM (pom.xml)

# Konfiguracja projektu

## Centralne miejsce informacji i konfiguracji projektu - plik POM (pom.xml)

## Zarządzanie przez konwencję - odpowiednia struktura katalogów i plików

# Struktura katalogów i plików



# Plik pom.xml

# Plik pom.xml

## POM - Project Object Model

# Plik pom.xml

## POM - Project Object Model

**Zawiera podstawowe informacje o projekcie i jego konfiguracji**

# Plik pom.xml

## POM - Project Object Model

## Zawiera podstawowe informacje o projekcie i jego konfiguracji

## Wymagane informacje:

**modelVersion** (4.0.0 – oznacza build zgodny z Maven 2)

**groupId** – ID grupy, do którego należy projekt (zazwyczaj pokrywa się z pakietem)

**artifactId** – ID artefaktu (projektu)

**version** - bieżąca wersja projektu

# Minimalistyczny plik pom.xml

```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>pl.org.dragonia</groupId>
    <artifactId>sample-app</artifactId>
    <version>1.0-SNAPSHOT</version>
</project>
```

# Archetypy

# Archetypy

## Gotowe do użycia projekty

# Archetypy

## Gotowe do użycia projekty

## Automatyczne generowanie i podstawowa konfiguracja

# Archetypy

Gotowe do użycia projekty

Automatyczne generowanie i podstawowa konfiguracja

**Ogromna ilość (setki!) gotowych i dostępnych archetypów**

# Najpopularniejsze archetypy

# Najpopularniejsze archetypy

maven-archetype-quickstart - prosty i podstawowy szkielet projektu z odpowiednią strukturą katalogów

# Najpopularniejsze archetypy

maven-archetype-quickstart - prosty i podstawowy szkielet projektu z odpowiednią strukturą katalogów

maven-archetype-webapp - aplikacja webowa z podstawową konfiguracją (w plikach XML)

# Najpopularniejsze archetypy

**maven-archetype-quickstart** - prosty i podstawowy szkielet projektu z odpowiednią strukturą katalogów

**maven-archetype-webapp** - aplikacja webowa z podstawową konfiguracją (w plikach XML)

**maven-archetype-j2ee-simple** - aplikacja JEE z podziałem na projekty i komponenty (w tym EJB w starej konfiguracji)

# Najprostszy program Java (Main.java)

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -
DgroupId=pl.edu.pg.ftims -DartifactId=oopl(1)

mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -
DarchetypeArtifactId=maven-archetype-quickstart -DgroupId=pl.edu.pg.ftims -
DartifactId=oopl(2)
```

(1) generowanie projektu na bazie archetypu wybranego z listy

(2) generowanie projektu na bazie konkretnego archetypu

# Budowanie i uruchomienie projektu

```
mvn package(1)

java -cp target/oopl-1.0-SNAPSHOT.jar pl.edu.pg.ftims.App(2)
```

(1) zbudowanie projektu

(2) uruchomienie

# Cykl budowania aplikacji

# Cykl budowania aplikacji

## Jasno określnone (i powtarzalne) reguły

# Cykl budowania aplikacji

## Jasno określnone (i powtarzalne) reguły

## Wystarczy znajomość zaledwie kilku komend do efektywnego budowania aplikacji

# Cykl budowania aplikacji

Jasno określnone (i powtarzalne) reguły

Wystarczy znajomość zaledwie kilku komend do efektywnego budowania aplikacji

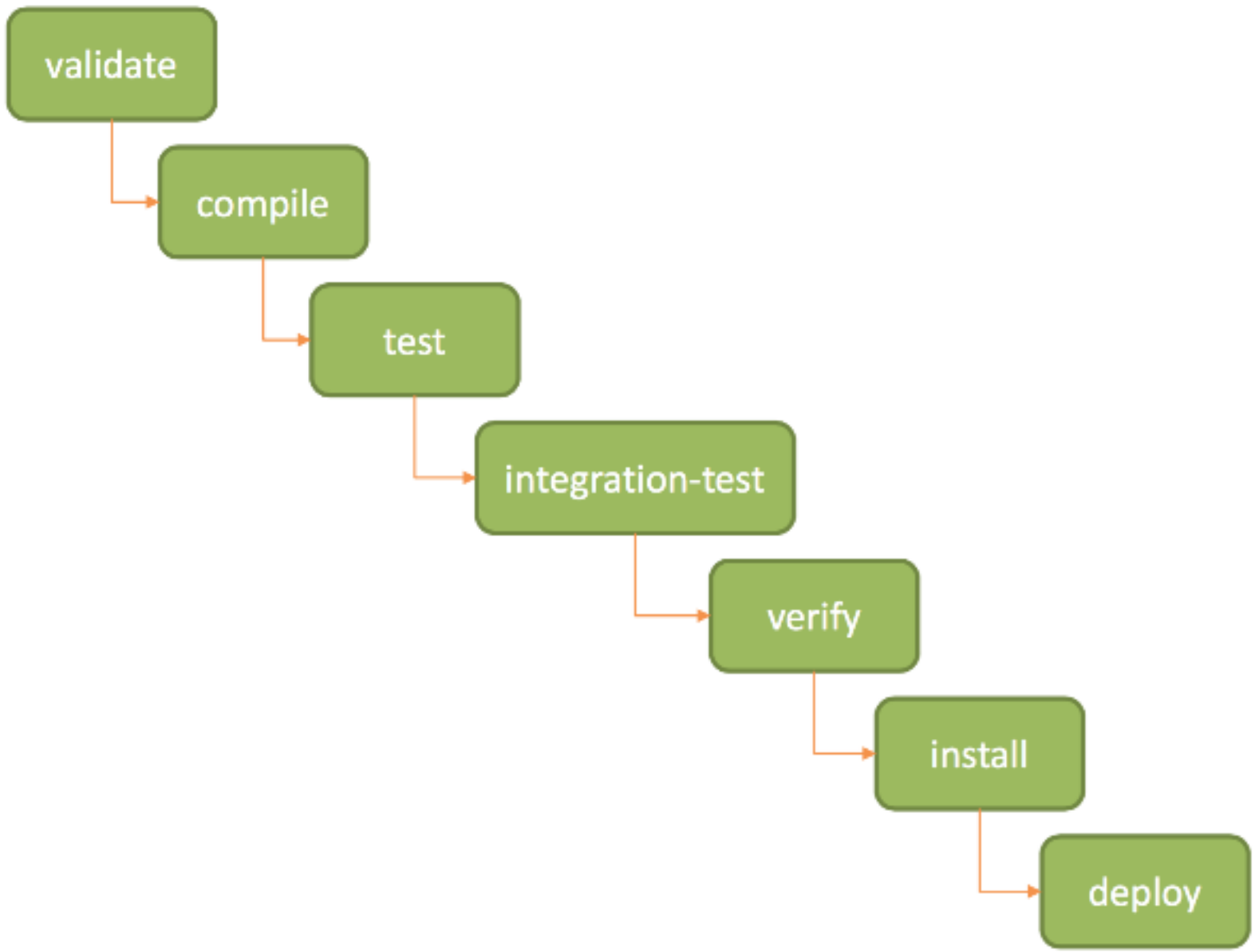
## Wbudowane cykle budowania:

**default** - deployment aplikacji

**clean** - czyszczenie projektu

**site** - tworzenie dokumentacji

# Domyślny cykl życia projektu Maven'owego



# ANY QUESTIONS?