Projekt Lombok jako sposób na uelastycznienie języka Java

Aleksander Wójcik
Informatyka IV stacjonarne
spec. inżynieria oprogramowania

Plan prezentacji

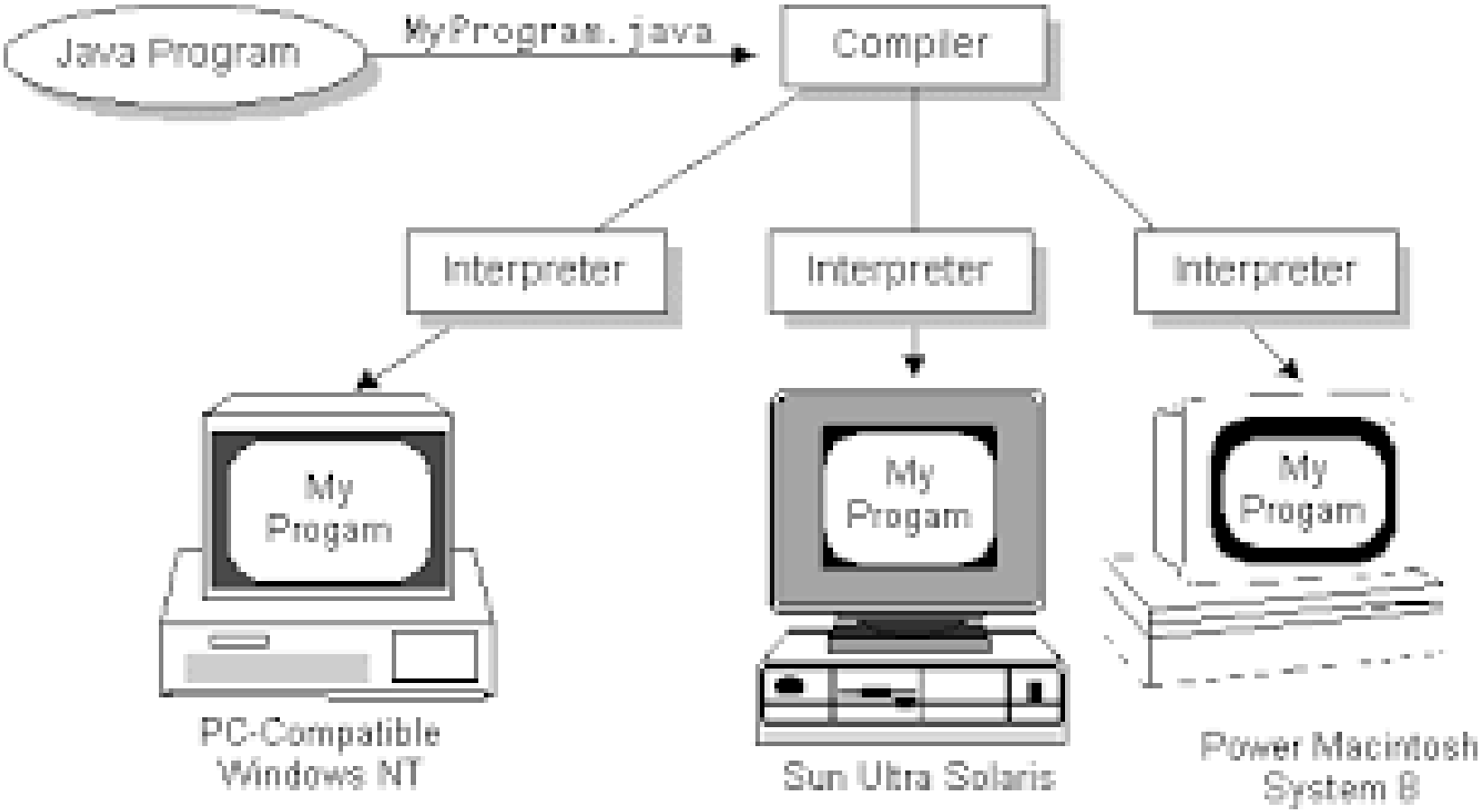
- Powstanie języka Java
- Opis języka i maszyny wirtualnej
- Zalety i problemy języka
- Próby rozwoju
- Projekt Lombok

Java - 1995

- **Niezależność od sprzętu**
- Projektowany do ogromnych projektów

- Garbage Collector, wyjątki, obiektowość

Niezależność od sprzętu



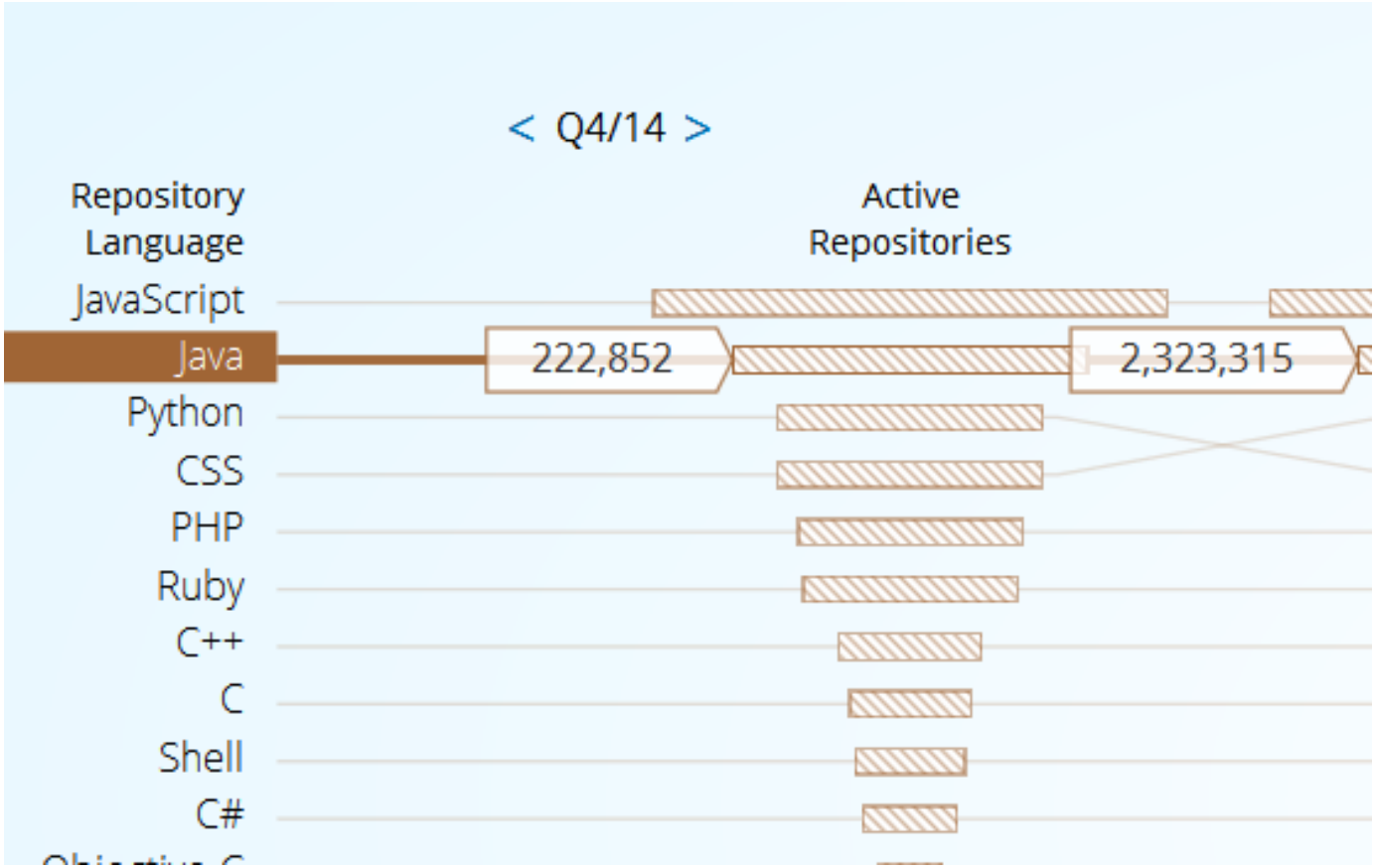
Java - 2015

- Wolna ewolucja języka – dla programistów
- Firma Oracle
- Pamięciożerność

- Wolna ewolucja języka – dla korporacji
- Ogromna ilość bibliotek i **dobrego softu**
- Ogromna architektura enterprise
- Kompatybilność wstecz

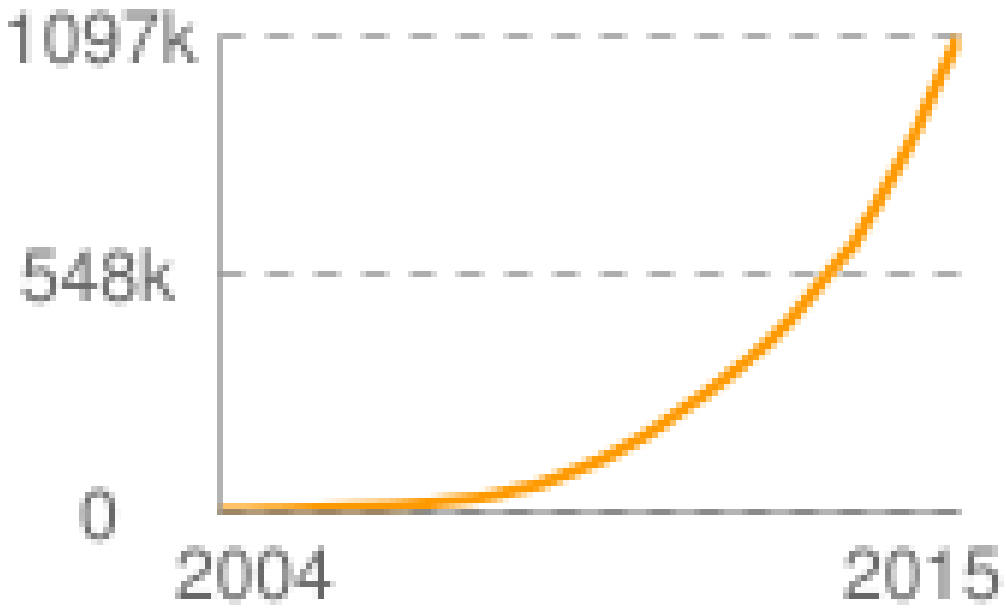
Biblioteki open source

- Github

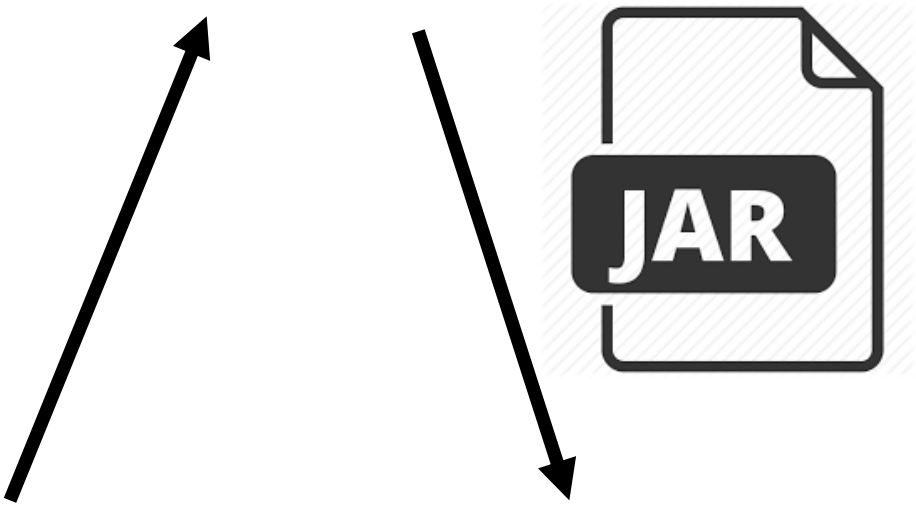


Biblioteki open source

- Maven

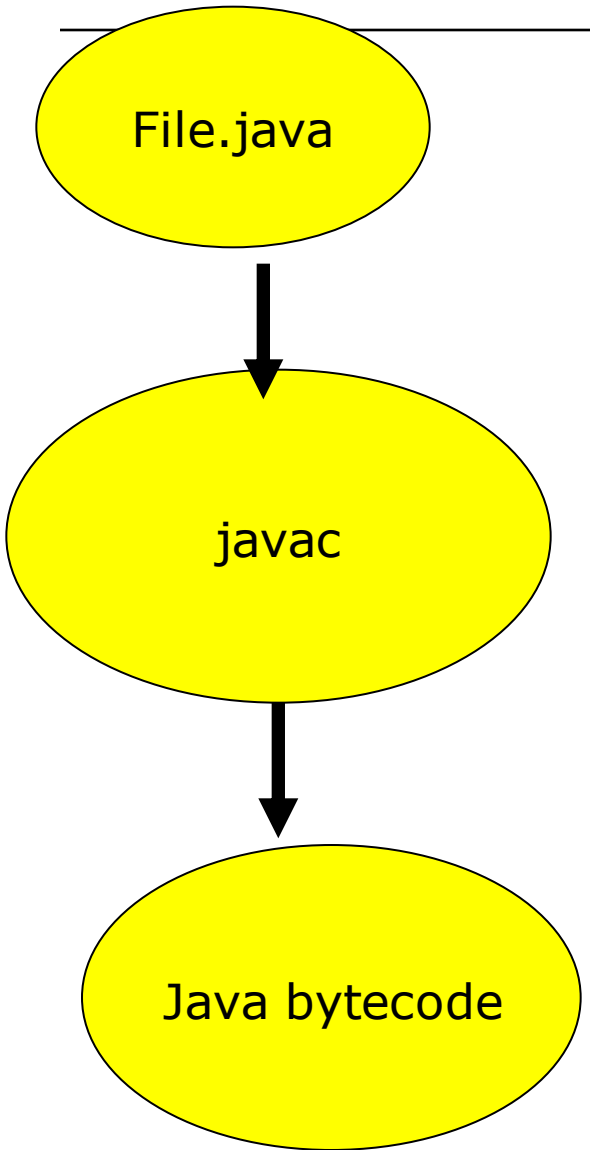


Maven™

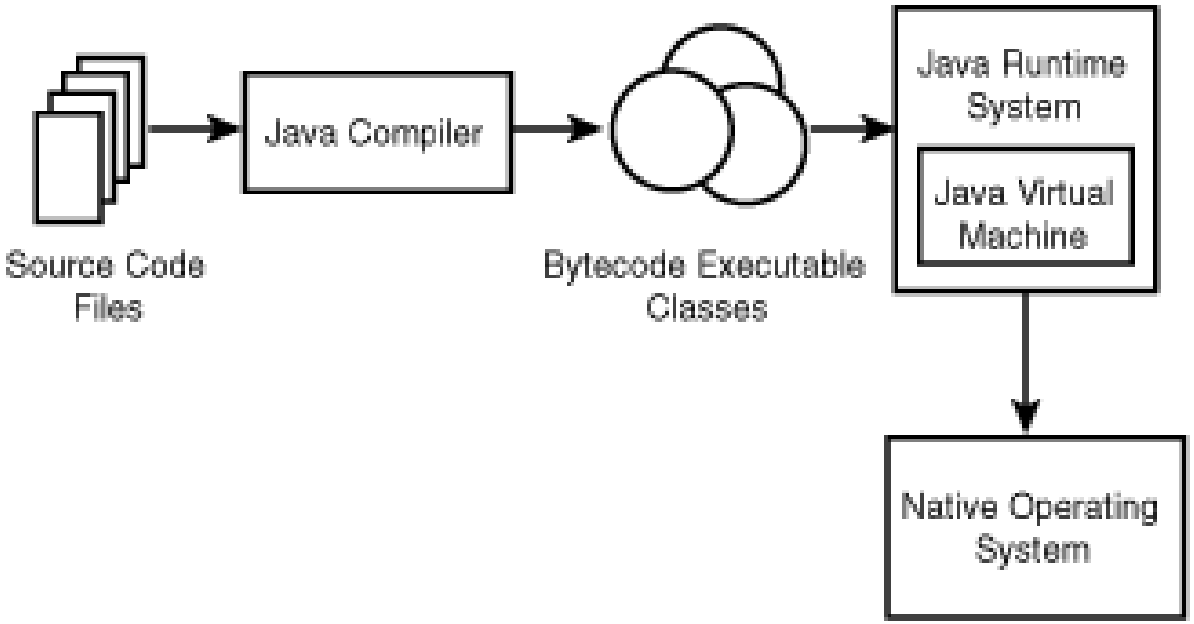


```
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.36</version>
</dependency>
```

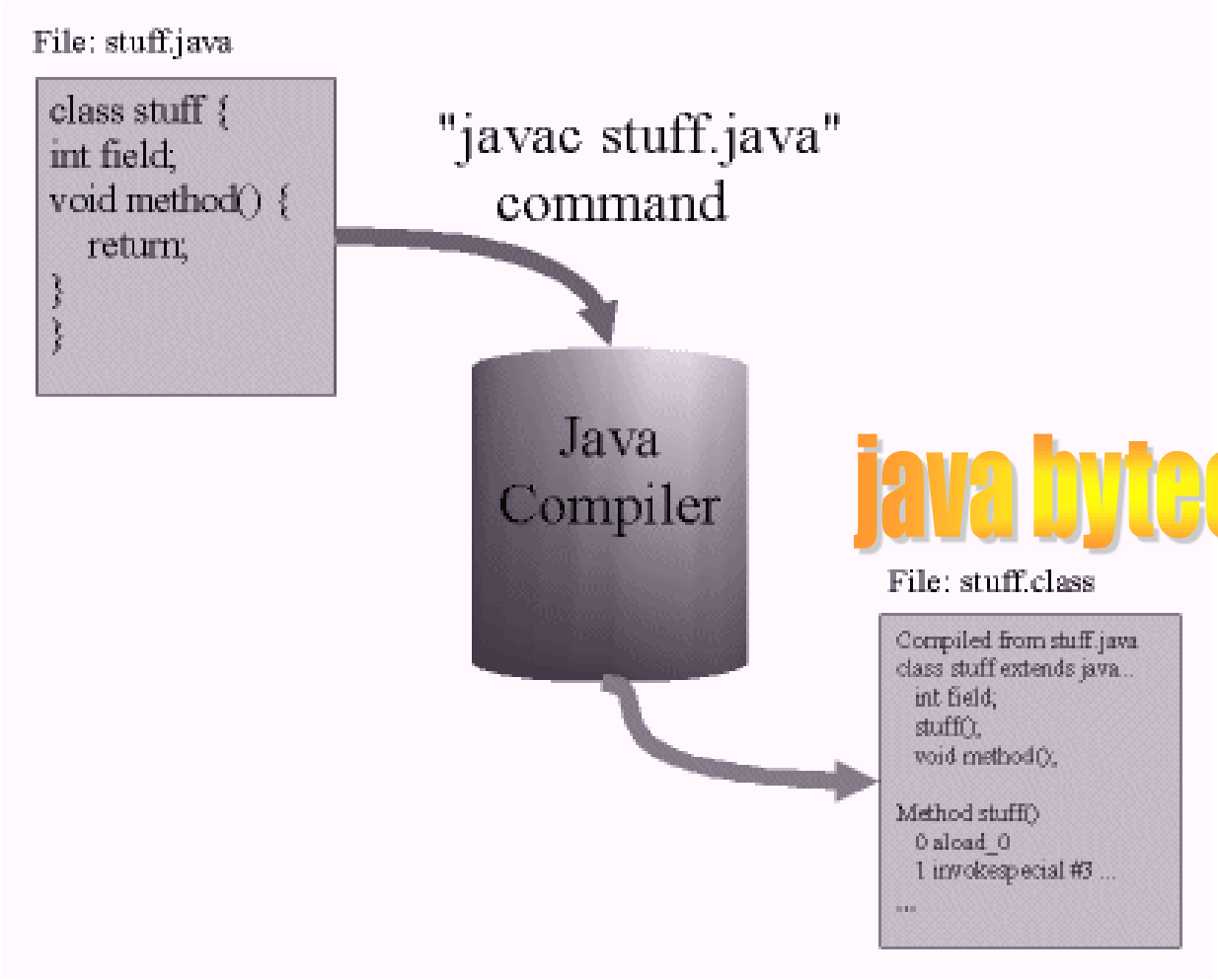
Kompilacja programu Java



Kompilacja programu



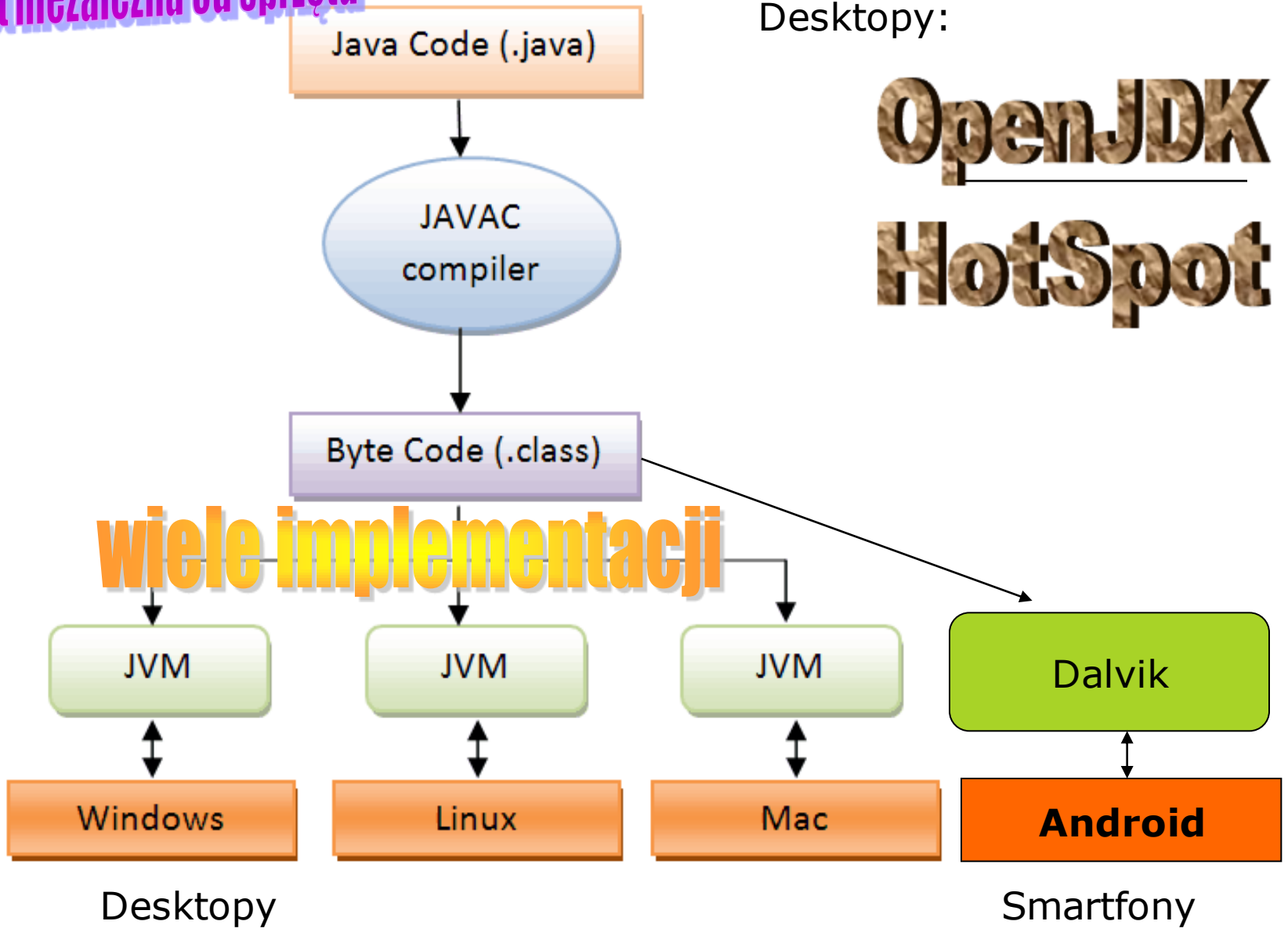
Kompilacja programu



Maszyna wirtualna oraz środowisko zdolne do wykonywania **kodu bajtowego** Javy

- 1. System operacyjny
- 2. Hardware – procesor itp.

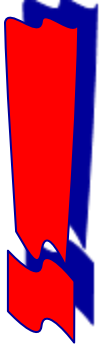
Java jest niezależna od sprzętu



Desktopy:

OpenJDK

HotSpot



Problem – słaba ewolucja

- **Assert –** nowe słowa kluczowe

```
public void assert(){

}
```

- JDK <=1.3

- JDK >1.4

Nowe feature'y - Java vs C#

	Java	C#
val/var	Brak – tylko Lombok	2007
lambda	2014	2007

Cele rozszerzeń Javy

Zostaje!:

1. Biblioteki – JDK,
2. JVM – środowisko uruchomieniowe
3. Niezależność od sprzętu

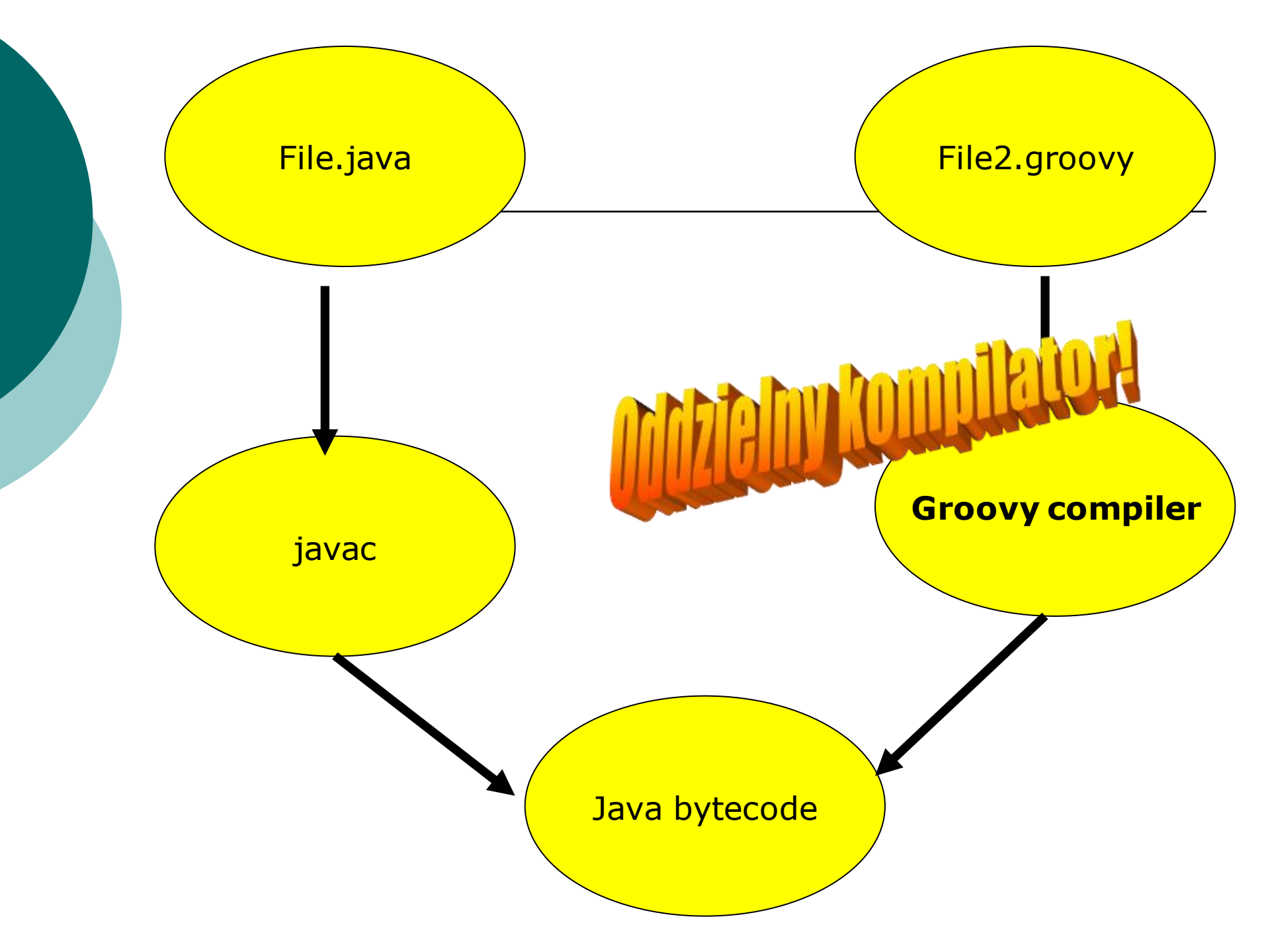
Nowe:

1. Składnia
2. Nowe elementy języka
3. Paradygmaty programowania (funkcyjne)
4. Możliwość typowania dynamicznego

JVM Languages

- Potrzeba osobnego kompilatora
- Niski udział w rynku -> uzależnienie projektu od dostępności programistów

- Rozszerzona funckjonalność języka
- Korzystają ze wszystkich dobrodziejstw Javy:
  - Bibliotek - Maven, Github itp.
  - JDK

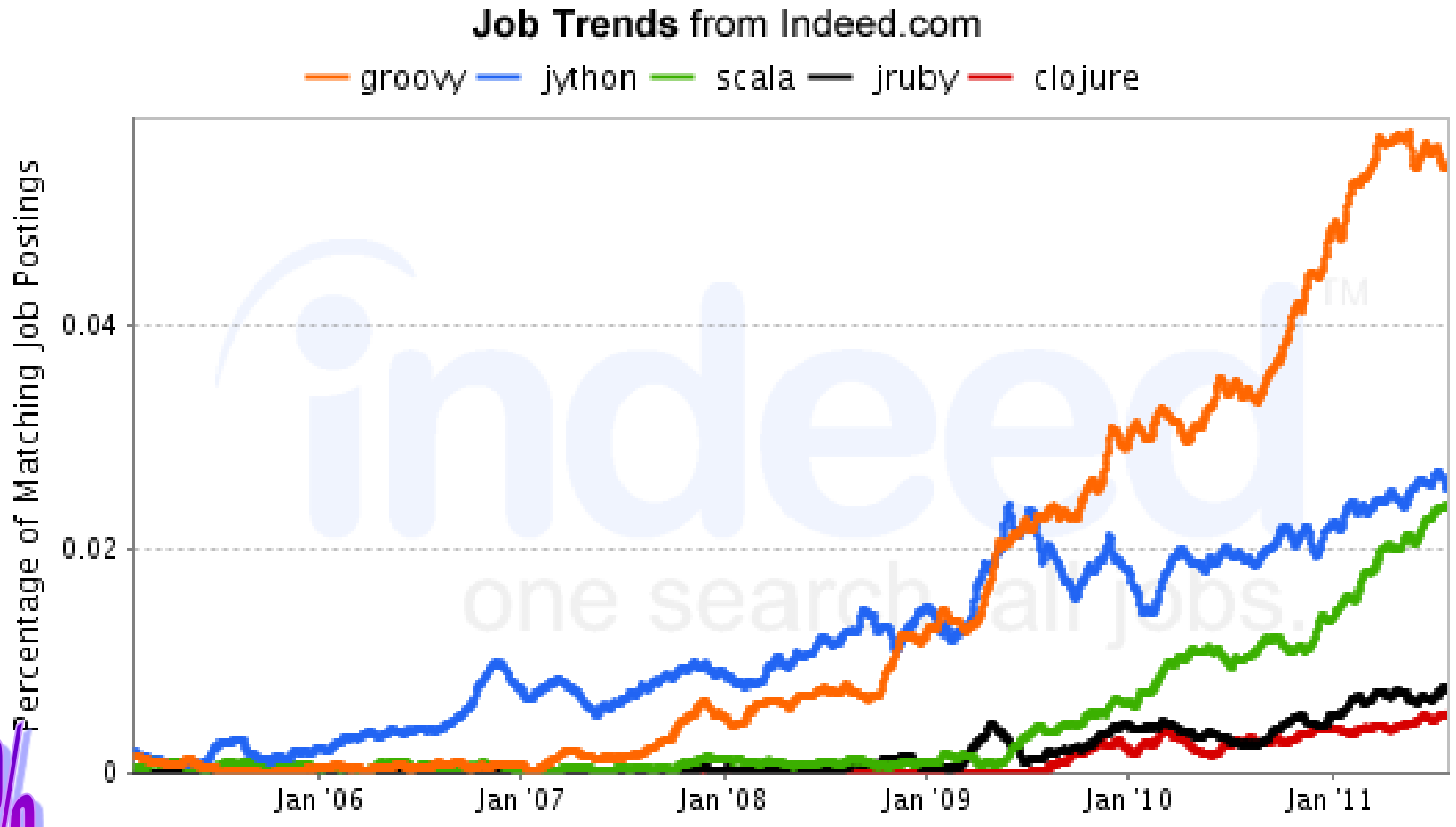

File.java
File2.groovy
Oddzielny kompilator!
javac
Groovy compiler
Java bytecode

JVM Languages

Lessons Learned:
Use of Modern JVM Languages Besides Java

MaibornWolff et al

Niskie zainteresowanie pracodawców

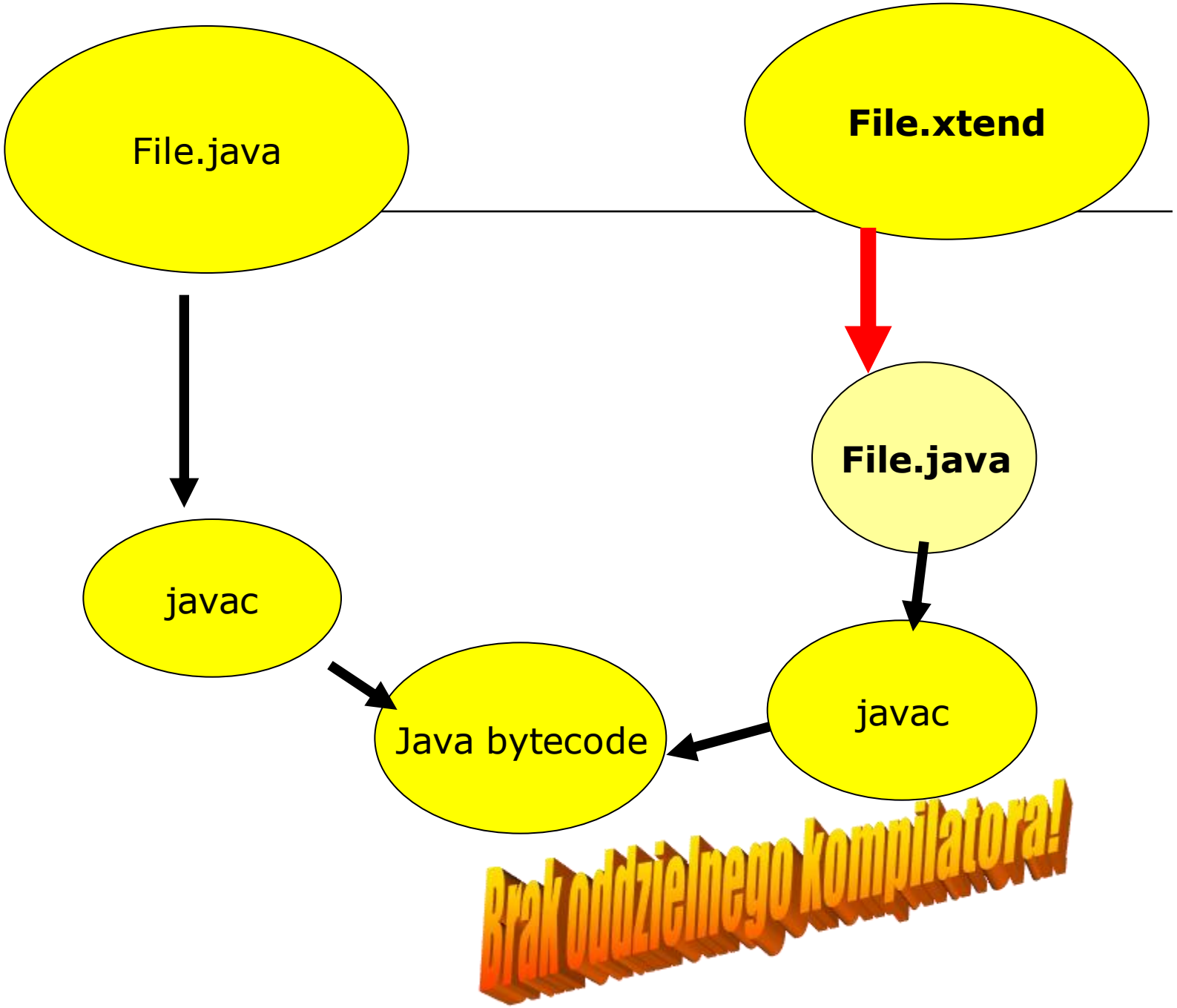


19%

Xtend

- Słabe wsparcie IDE

- Nauka w 1-2 dni
- Brak konieczności osobnego kompilatora


File.java
File.xtend
File.java
javac
javac
Java bytecode
Brak oddzielnego kompilatora!

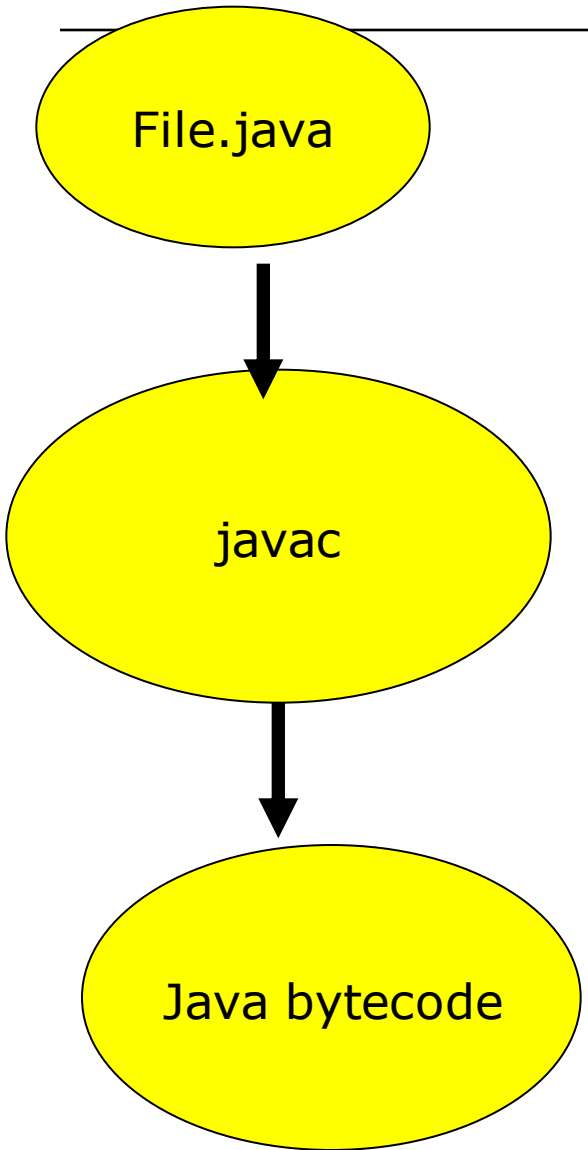
```
1. class HelloWorld {
2.   def static void main(String[] args) {
3.     println("Hello World")
4.   }
5. }
```

```
1. // Generated Java Source Code
2. import org.eclipse.xtext.xbase.lib.InputOutput;
3.
4. public class HelloWorld {
5.   public static void main(final String[] args) {
6.     InputOutput.<String>println("Hello World");
7.   }
8. }
```

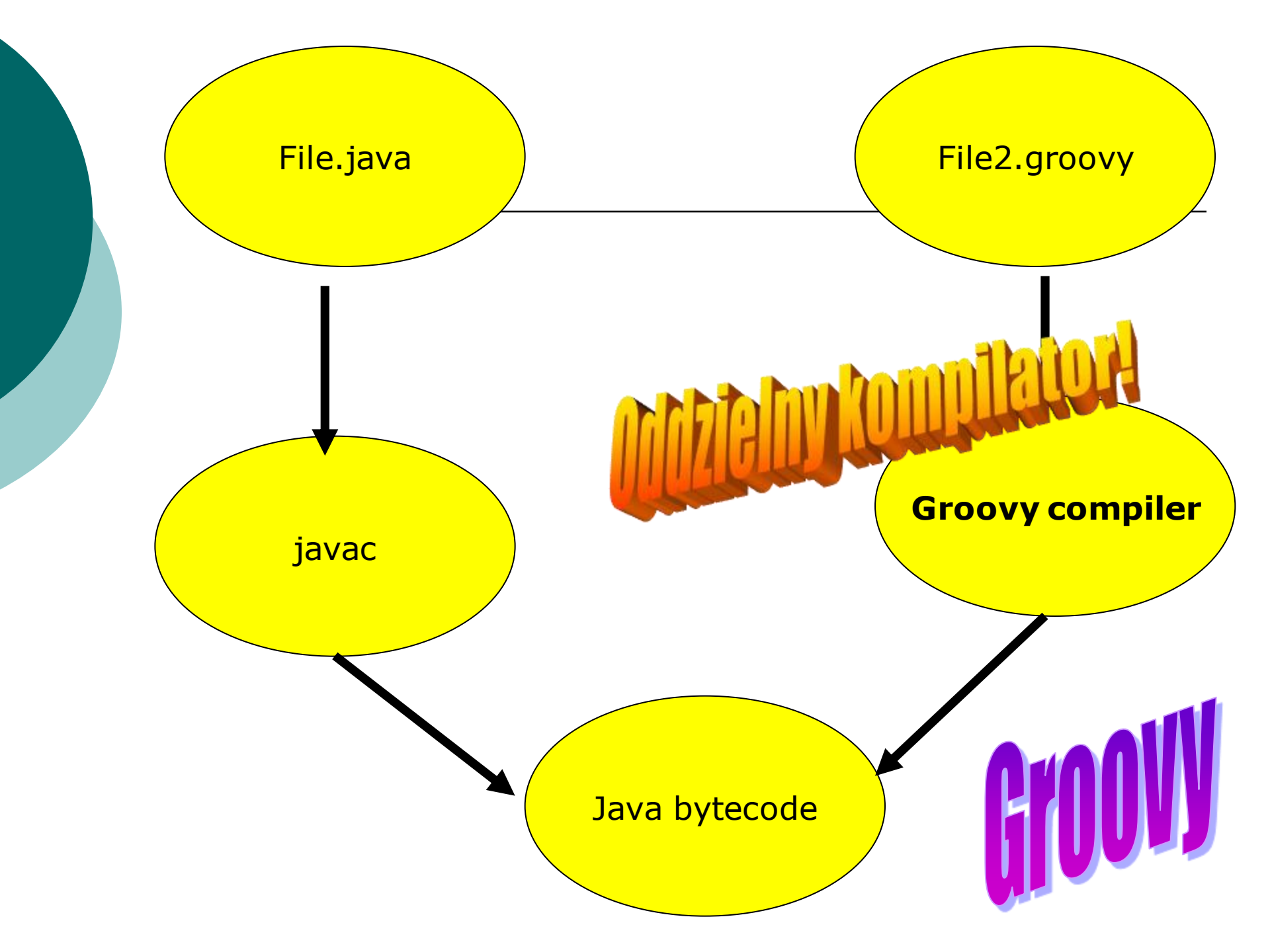
Projekt Lombok

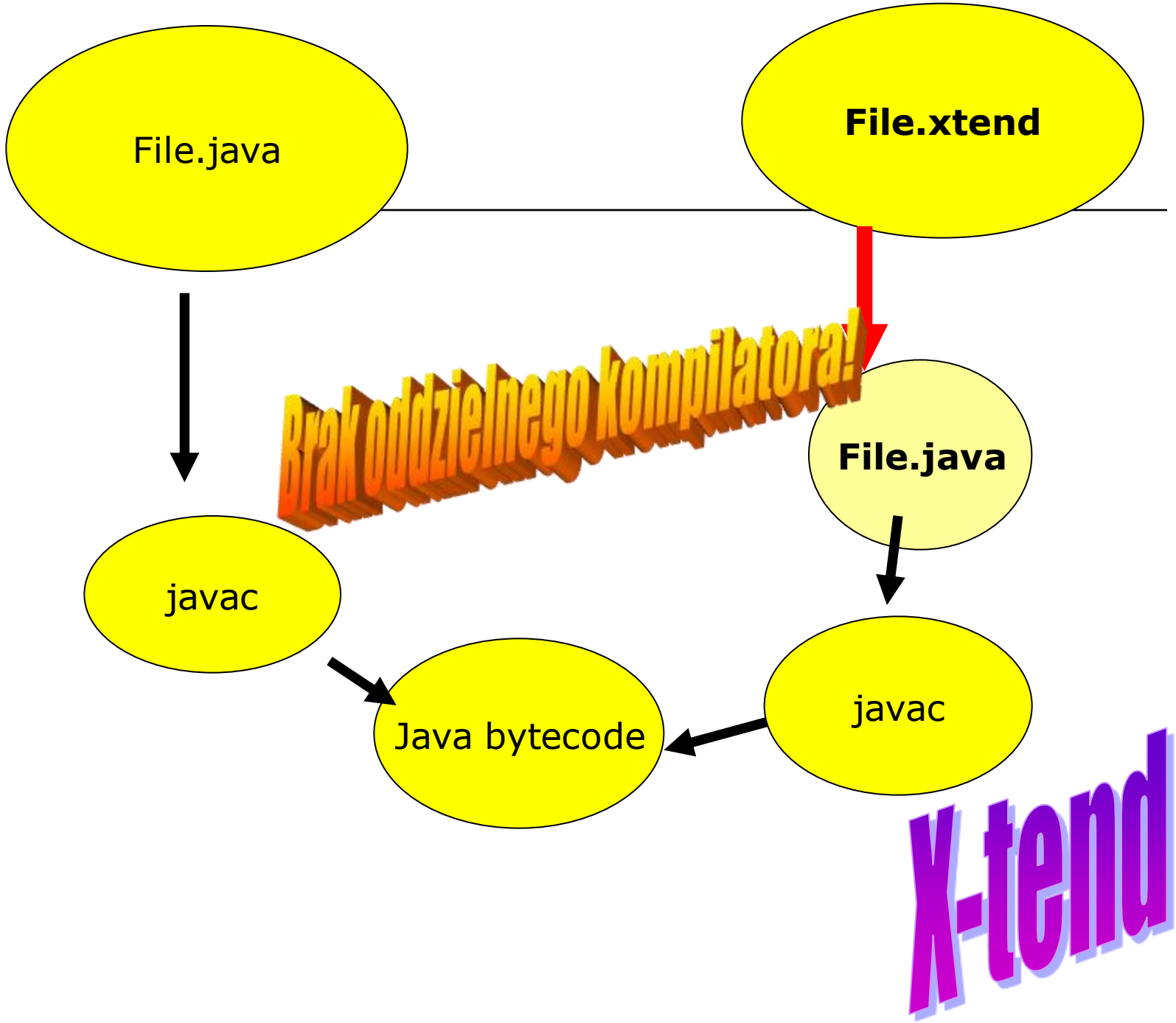
- Naruszenie kontraktu adnotacji
- Konieczność generowania AST dla każdego IDE

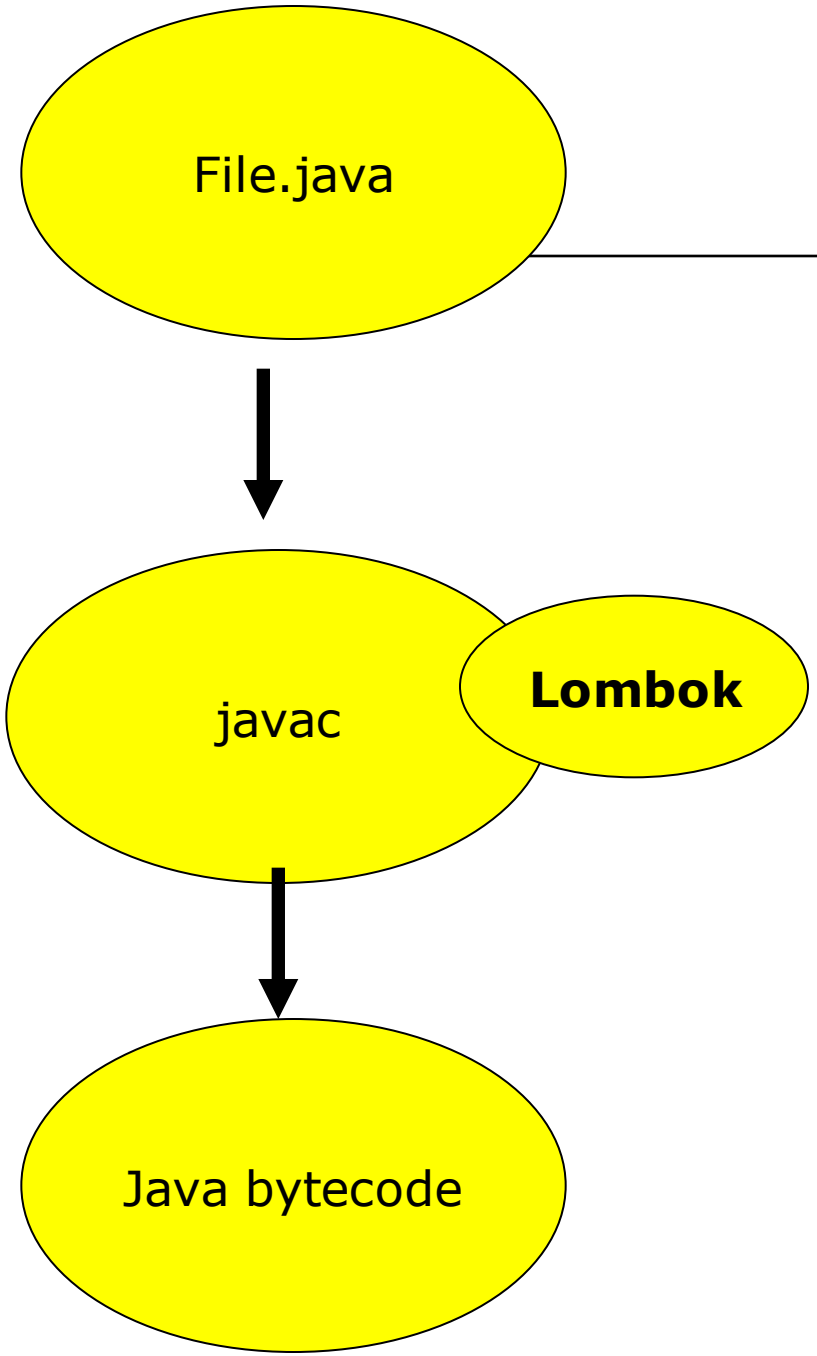
- Nauka w 1-2 godziny
- Możliwość definiowania własnych adnotacji – b. trudna


File.java
javac
Java bytecode

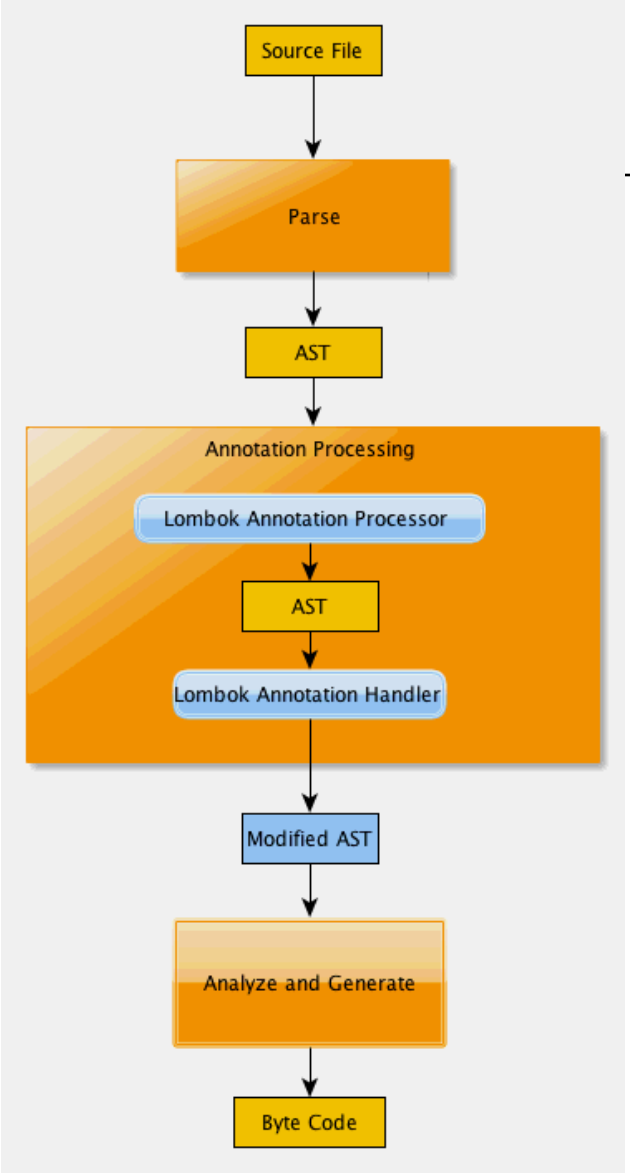

Java


File.java
File2.groovy
Oddzielny kompilator!
javac
Groovy compiler
Java bytecode
Groovy


File.java
File.xtend
Brak oddzielnego kompilatora!
File.java
javac
Java bytecode
javac
X-tend



Lombok



AST-

Abstract

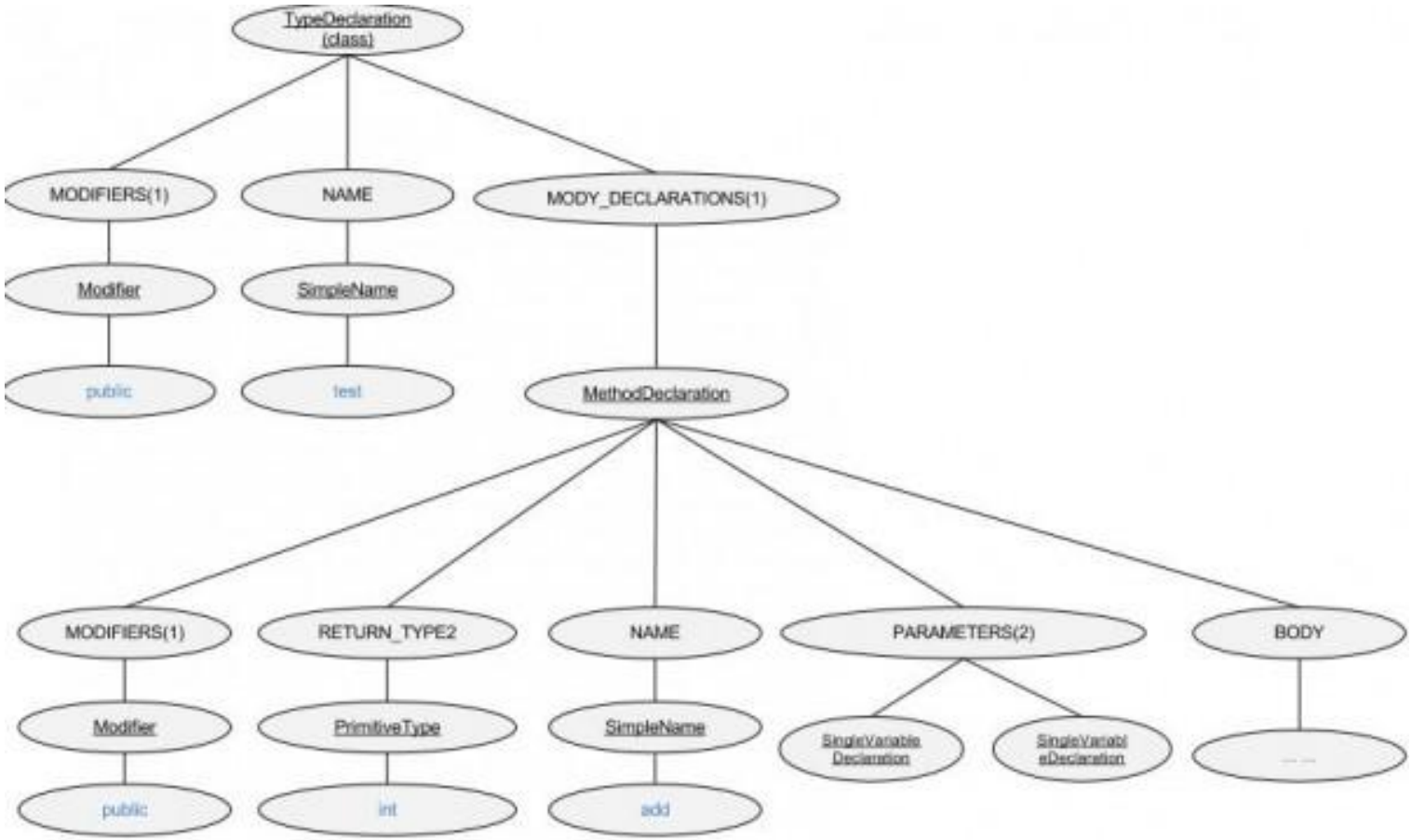
Syntax

Tree

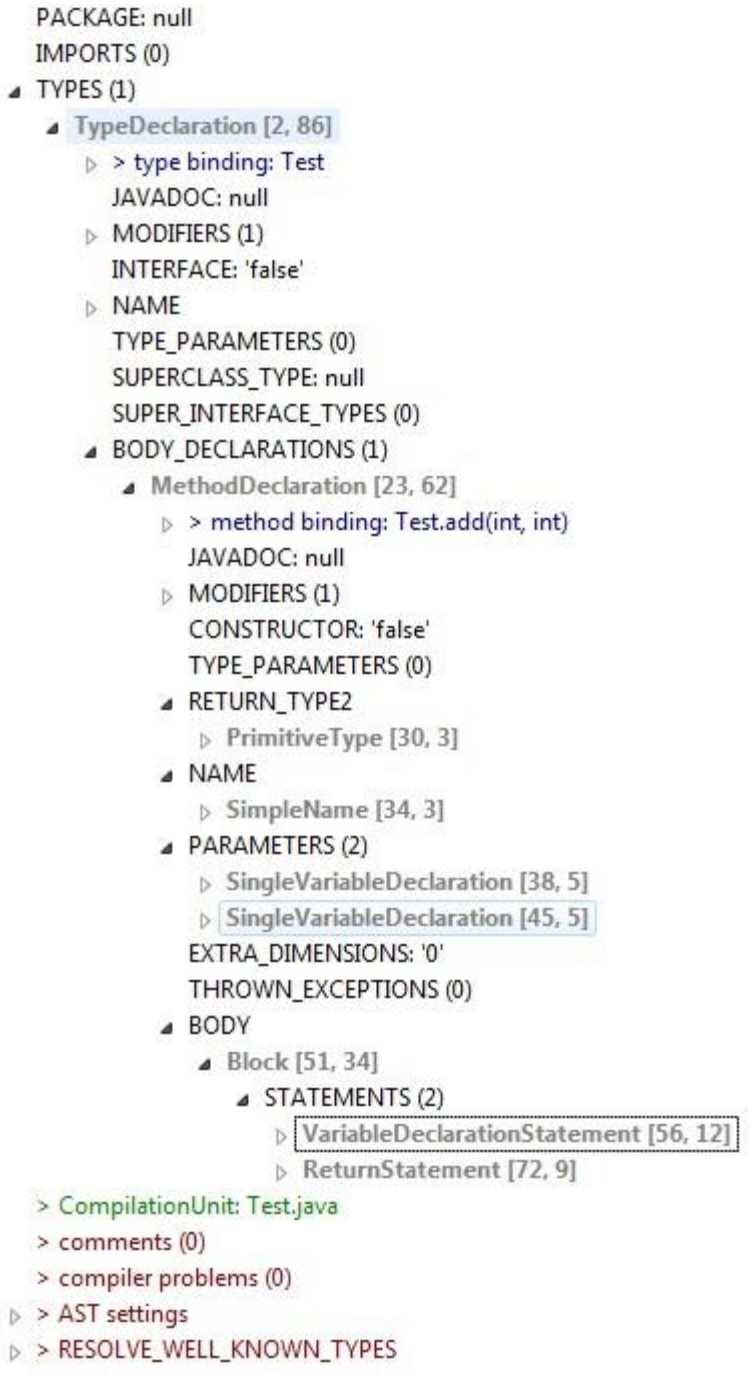
Transformacja do AST

```
public class Test {
        public int add(int a, int b){
                int c = a+b;
                return c;
        }
}
```

```java
public class Test {
        public int add(int a, int b){
                int c = a+b;
                return c;
        }
}
```



podgląd AST w IDE



Lombok

encapsulation

C#

```
//C#
public class Rectangle{
    private int width{get;set;};
    private int height{get;set;};
}
```

Plain Java

```
//Java
public class Rectangle{
    private int width;
    private int height;

    public int getWidth() {
        return width;
    }
    public void setWidth(int width) {
        this.width = width;
    }
    public int getHeight() {
        return height;
    }
    public void setHeight(int height) {
        this.height = height;
    }
}
```

Single field

```
//Lombok
public class Rectangle{
    @Getter @Setter
    private int width;
    @Getter @Setter
    private int height;
}
```

All fields in class

```
//Lombok
@Getter @Setter
public class Rectangle{
    private int width;
    private int height;
}
```

With Lombok


```
@NoArgsConstructor
public static class NoArgsExample {
  @NonNull private String field;
}
```

@NoArgsConstructor
@RequiredArgsConstructor
@AllArgsConstructor

Plain Java

```
public static class NoArgsExample {
  @NonNull private String field;

  public NoArgsExample() {
  }
}
```

With Lombok

undefined type

```
public class ValExample {
  public String example() {
    val example = new ArrayList<String>();
    example.add("Hello, World!");
    val foo = example.get(0);
    return foo.toLowerCase();
  }
```

Plain Java

```
public class ValExample {
  public String example() {
    final ArrayList<String> example = new ArrayList<String>();
    example.add("Hello, World!");
    final String foo = example.get(0);
    return foo.toLowerCase();
  }
```

With Lombok

checked exceptions

```java
public class SneakyThrowsExample implements Runnable {
  @SneakyThrows(UnsupportedEncodingException.class)
  public String utf8ToString(byte[] bytes) {
    return new String(bytes, "UTF-8");
  }
```

Plain Java

```java
public class SneakyThrowsExample implements Runnable {
  public String utf8ToString(byte[] bytes) {
    try {
      return new String(bytes, "UTF-8");
    } catch (UnsupportedEncodingException e) {
      throw Lombok.sneakyThrow(e);
    }
  }
```

With Lombok

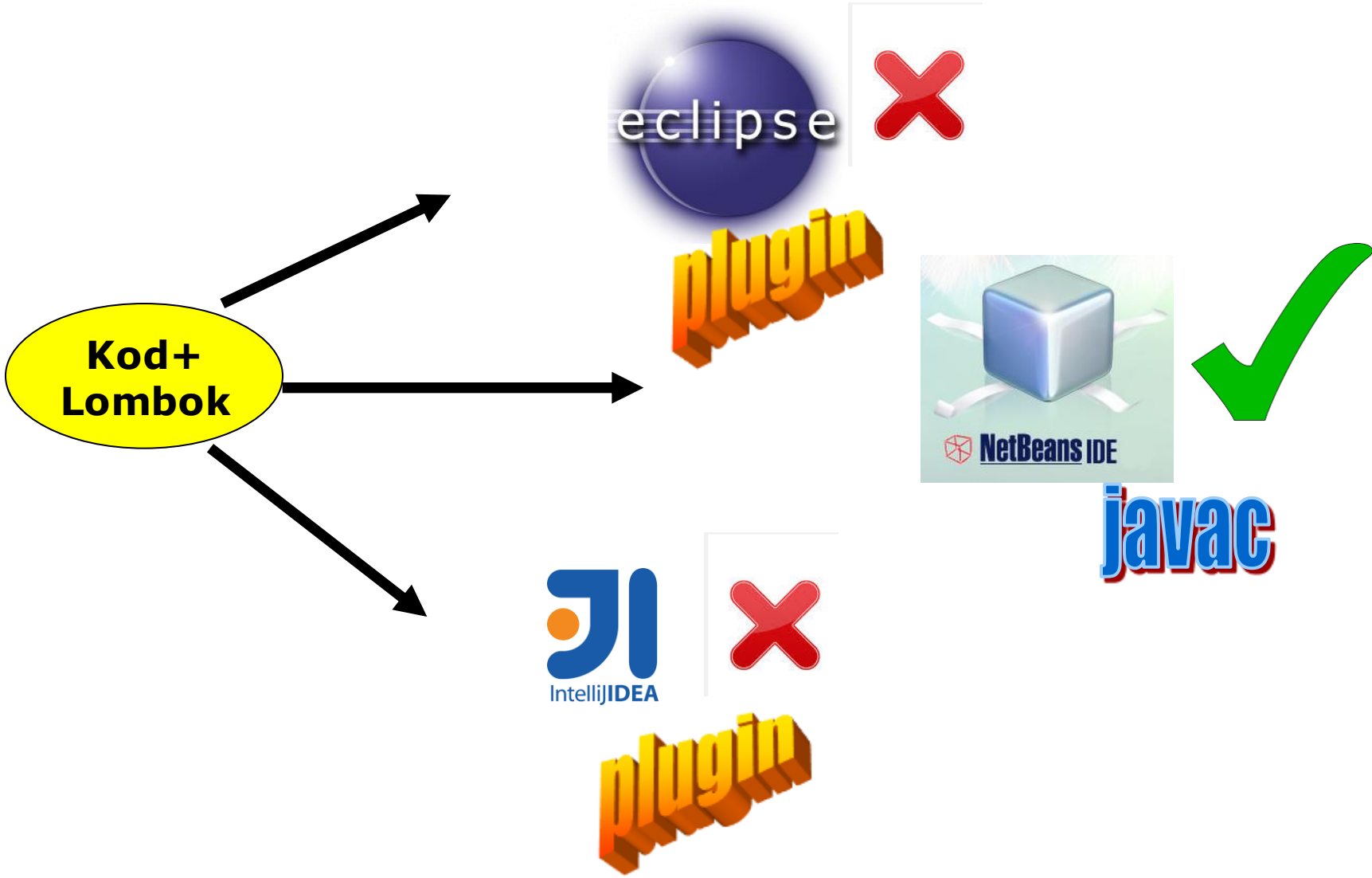

```java
public class WitherExample {
  @Wither private final int age;
  @Wither(AccessLevel.PROTECTED) @NonNull private final String name;

  public WitherExample(String name, int age) {
    if (name == null) throw new NullPointerException();
    this.name = name;
    this.age = age;
  }
}
```

Plain Java

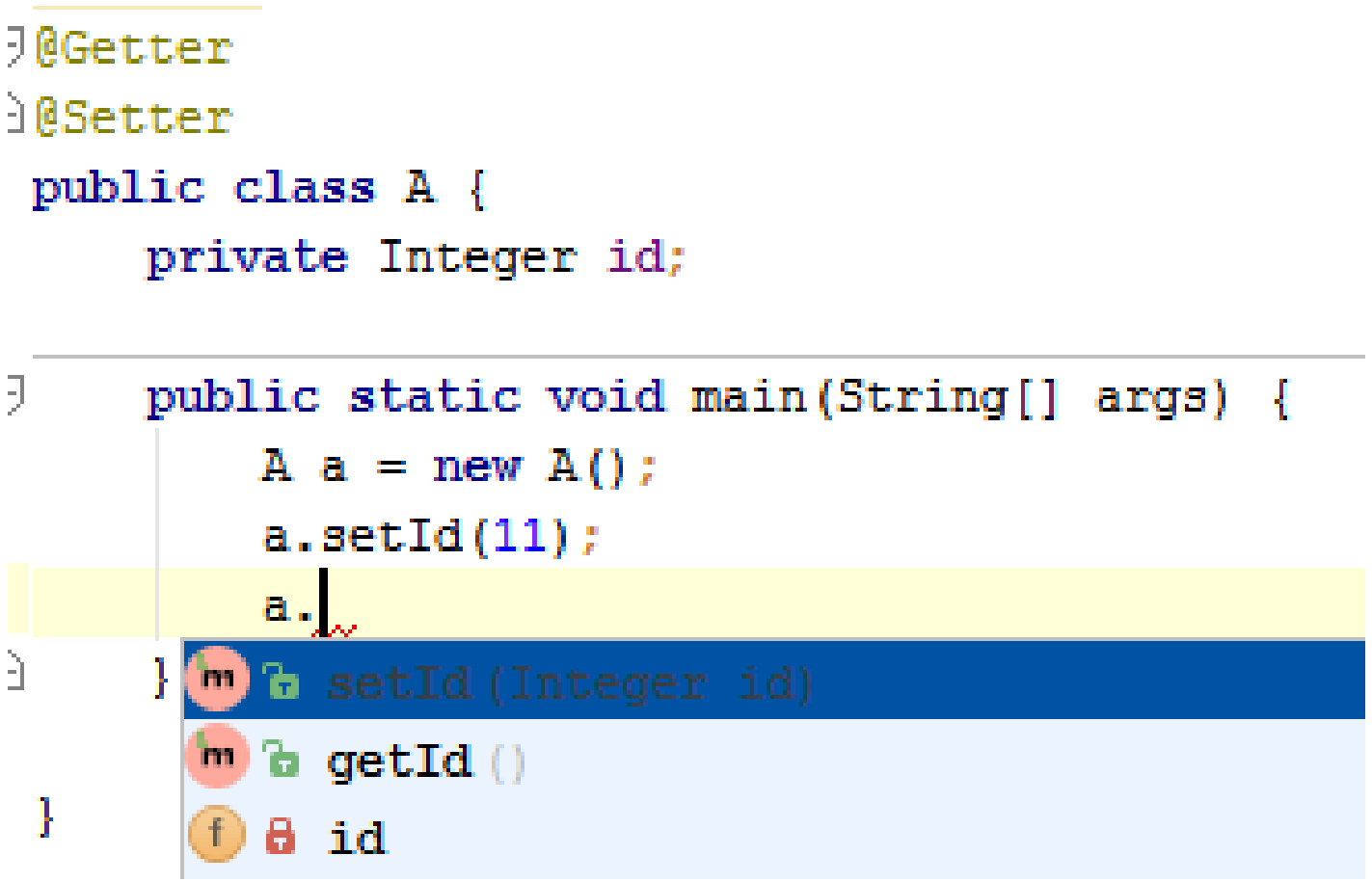
```java
public class WitherExample {
  private final int age;
  private @NonNull final String name;

  public WitherExample(String name, int age) {
    if (name == null) throw new NullPointerException();
    this.name = name;
    this.age = age;
  }

  public WitherExample withAge(int age) {
    return this.age == age ? this : new WitherExample(age, name);
  }

  protected WitherExample withName(@NonNull String name) {
    if (name == null) throw new java.lang.NullPointerException("name");
    return this.name == name ? this : new WitherExample(age, name);
  }
}
```

Lombok

IDE - plugins


eclipse
plugin
Kod+
Lombok
NetBeans IDE
javac
IntelliJIDEA
plugin

IDE - OK


@Getter
@Setter
public class A {
private Integer id;
public static void main(String[] args) {
A a = new A();
a.setId(11);
a.
setId(Integer id)
getId()
id

IDE – nie OK

```
@Getter
@Setter
public class A {
    private Integer id;

    public static void main(String[] args) {
        A a = new A();
        a.setId(11);
        a.
    }
}
```

id
clone()
equals(Object)

Lombok

Projekty rozszerzające

- Lombok-pg
- Lombok-experimental features

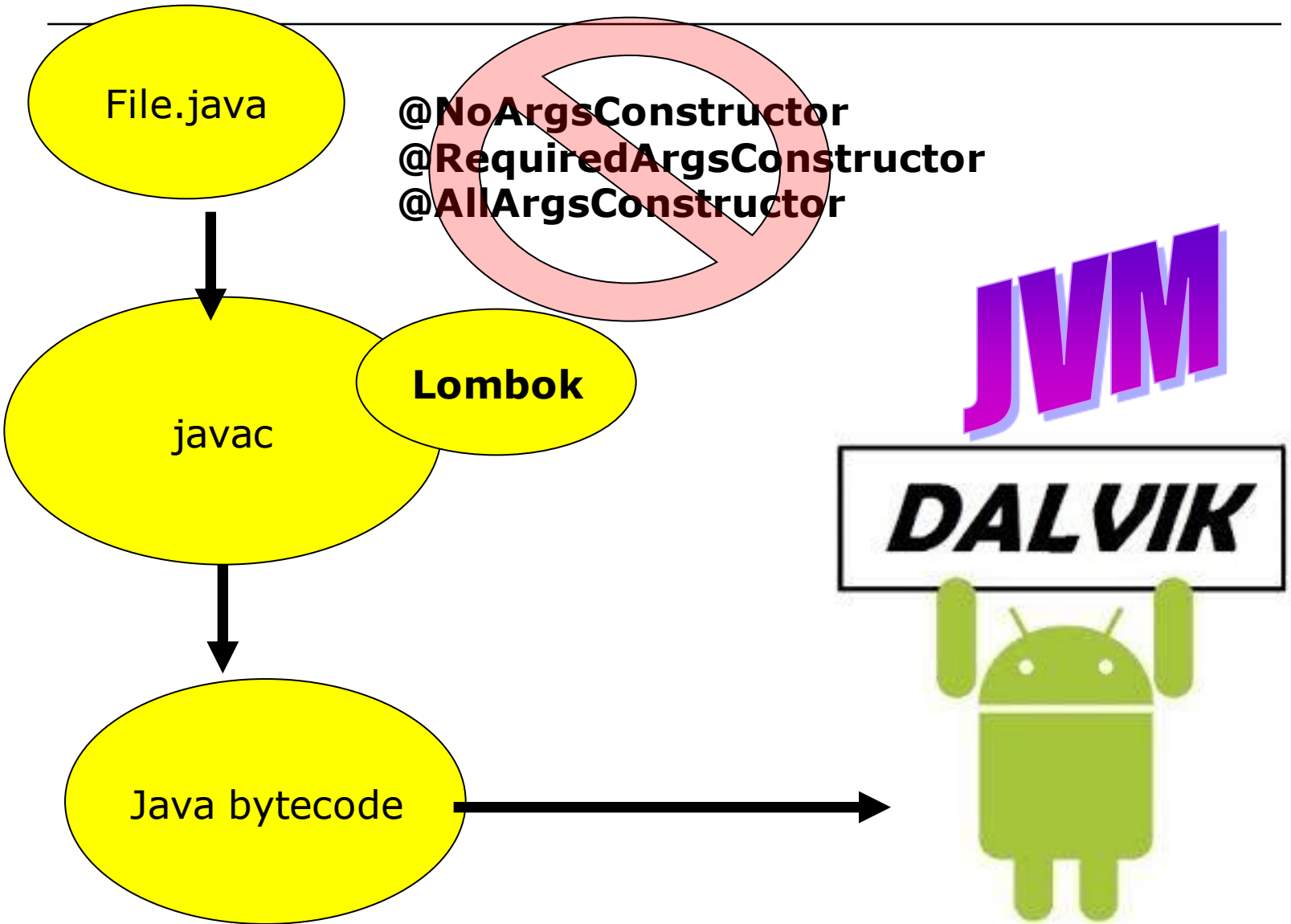
Lombok – własne adnotacje

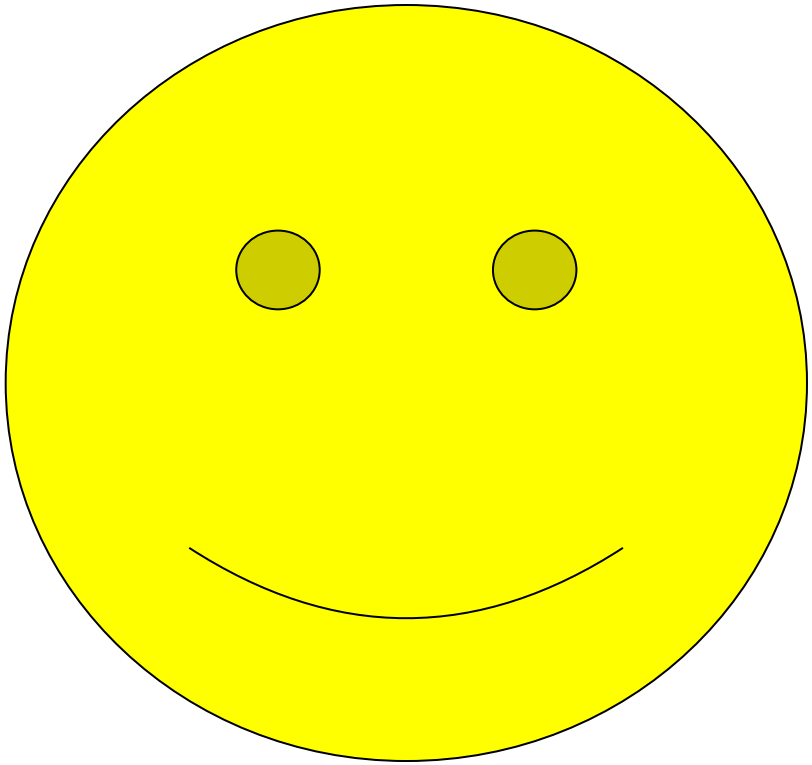
- Utworzyć adnotację
- Zaimplementować węzły AST dla kompilatora javac:

```
public abstract void handle(
        AnnotationValues<T> annotation, JCTree.JCAnnotation ast, JavacNode annotationNode);
```

- (*) Zaimplementować węzły AST dla kompilatorów pod inne IDE (IJ, Eclipse)

Lombok + Android





Dziękuję za uwagę!