Uniwersytet Wrocławski

Rafał Lewandków

WSTĘP DO PROGRAMOWANIA URZĄDZEŃ MOBILNYCH KOTLIN, JAVA

WYKŁAD 1

- PODSTAWOWE INFORMACJE
- TREŚCI PROGRAMOWE
- ZASADY ZALICZENIA

Uniwersytet Wrocławski

PODSTAWOWE INFORMACJE

Rafał Lewandków
pokój 075
rafal.lewandkow@uwr.edu.pl
rafal.lewandkow2@uwr.edu.pl

Forma zajęć i liczba godzin:
Wykład 15 godz. /Laboratorium 30 godz.

Literatura obowiązkowa i zalecana:

- Java. Efektywne programowanie. Joshua Bloch
- Atomic Kotlin. Bruce Eckel

Nakład pracy studenta:
Praca własna studenta: 30 godz.
Łączna liczba godzin 75
Liczba punktów ECTS: 3

PODSTAWOWE INFORMACJE

Rafał Lewandków
pokój 075
rafal.lewandkow@uwr.edu.pl
rafal.lewandkow2@uwr.edu.pl

Konsultacje:

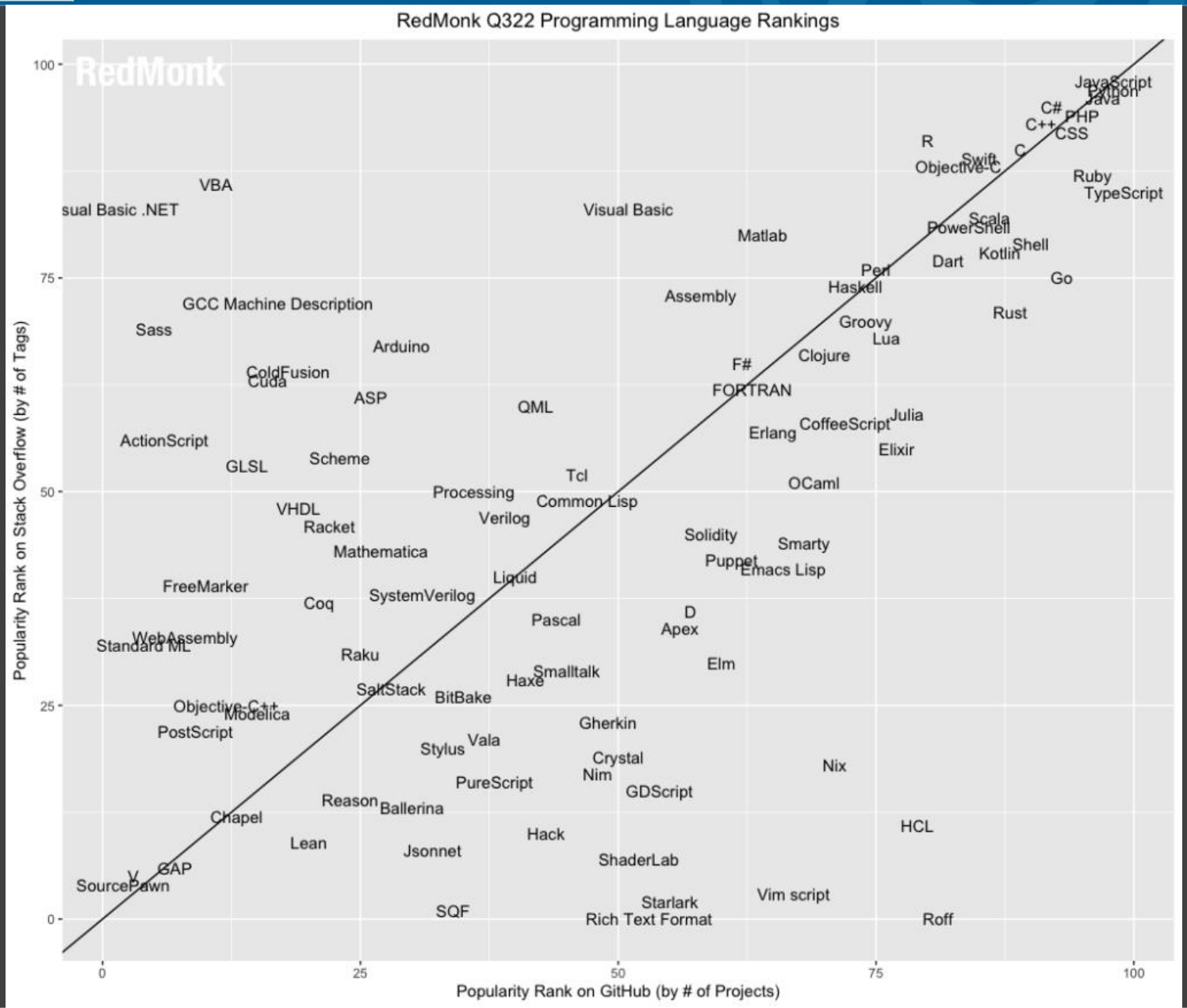
ZASADY ZALICZENIA

Warunkiem zaliczenia laboratorium jest uzyskanie pozytywnej oceny z wykonywanych zadań umieszczanych na listach

- Na zajęcia przewidzianych jest 8 list zadań
- Z każdej listy wystawiana jest osobna ocena
- Nie jest konieczne zaliczenie wszystkich list aby otrzymać ocenę pozytywną z laboratorium
- Każda lista posiada informację o liczbie punktów wymaganych na konkretną ocenę
- Każda lista posiada termin zwrotu
- Za każdy tydzień opóźnienia otrzymana ocena jest obniżana o 0,5
- Każdą listę można poprawić w ciągu 4 tygodni od terminu zwrotu listy - za wyjątkiem końca semestru
- Poprawa list jest możliwa pod warunkami uzyskania co najmniej 30% punktów i oddania listy w terminie
- Ocena końcowa jest średnią arytmetyczną ze wszystkich ocen z list.
- Na ocenę 3,0 wymagana jest średnia co najmniej 3,0

Uniwersytet Wrocławski

Wstęp



Uniwersytet Wrocławski

Treści Programowe

1. Porównanie języków Java i Kotlin
2. Typy danych
3. Wyrażenia, instrukcje, pętle
4. Funkcje
5. Klasy, obiekty
6. Interfejsy
7. Wielowątkowość
8. Wzorce projektowe

IntelliJ Idea

Kotlin – Język Przemysłowy

Kotlin (Introduced 2011, Version 1.0: 2016)

- FORTRAN: FORmula TRANslation (1957)
- LISP: LISt Processor (1958)
- ALGOL: ALGOrithmic Language (1958)
- COBOL: COmmon Business-Oriented Language (1959)
- BASIC: Beginners' All-purpose Symbolic Instruction Code (1964)
- Simula 67, the Original Object-Oriented Language (1967)
- Pascal (1970)
- C (1972)
- Smalltalk (1972)
- C++: A Better C with Objects (1983)
- Python: (1990)
- Haskell: Pure Functional Programming (1990)
- Java: Virtual Machines and Garbage Collection (1995)
- JavaScript: (1995)
- C#: (2000)
- Scala: (2003)
- Groovy: (2007)

Uniwersytet Wrocławski

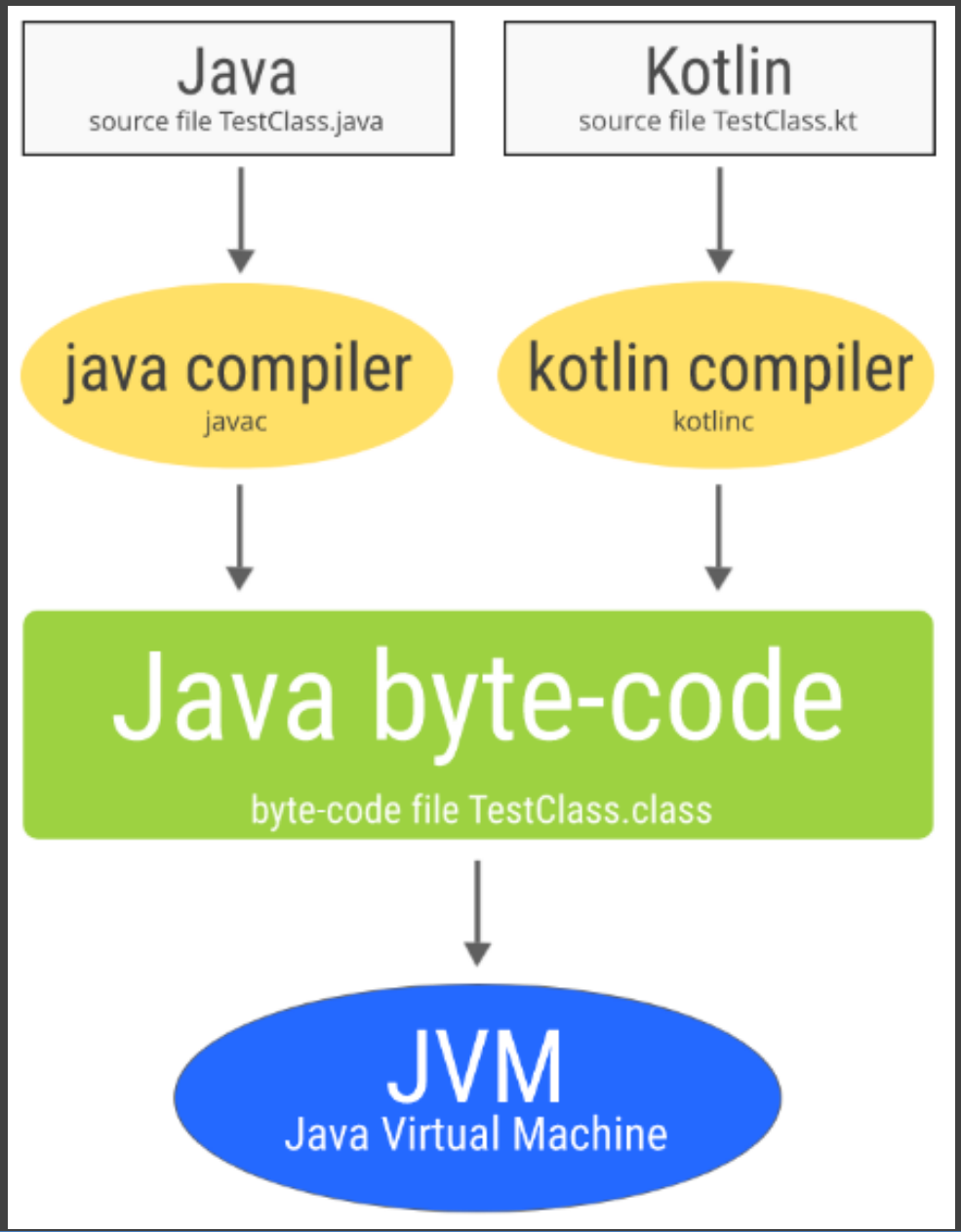
JVM, JRE, JDK

JDK (**Java Development Kit**) - Pakiet Programisty Javy. JDK zawiera Środowisko Uruchomieniowe Javy (tzn. JRE) oraz zestaw narzędzi niezbędnych do wytwarzania oraz kompilowania oprogramowania tworzonego w języku JAVA.

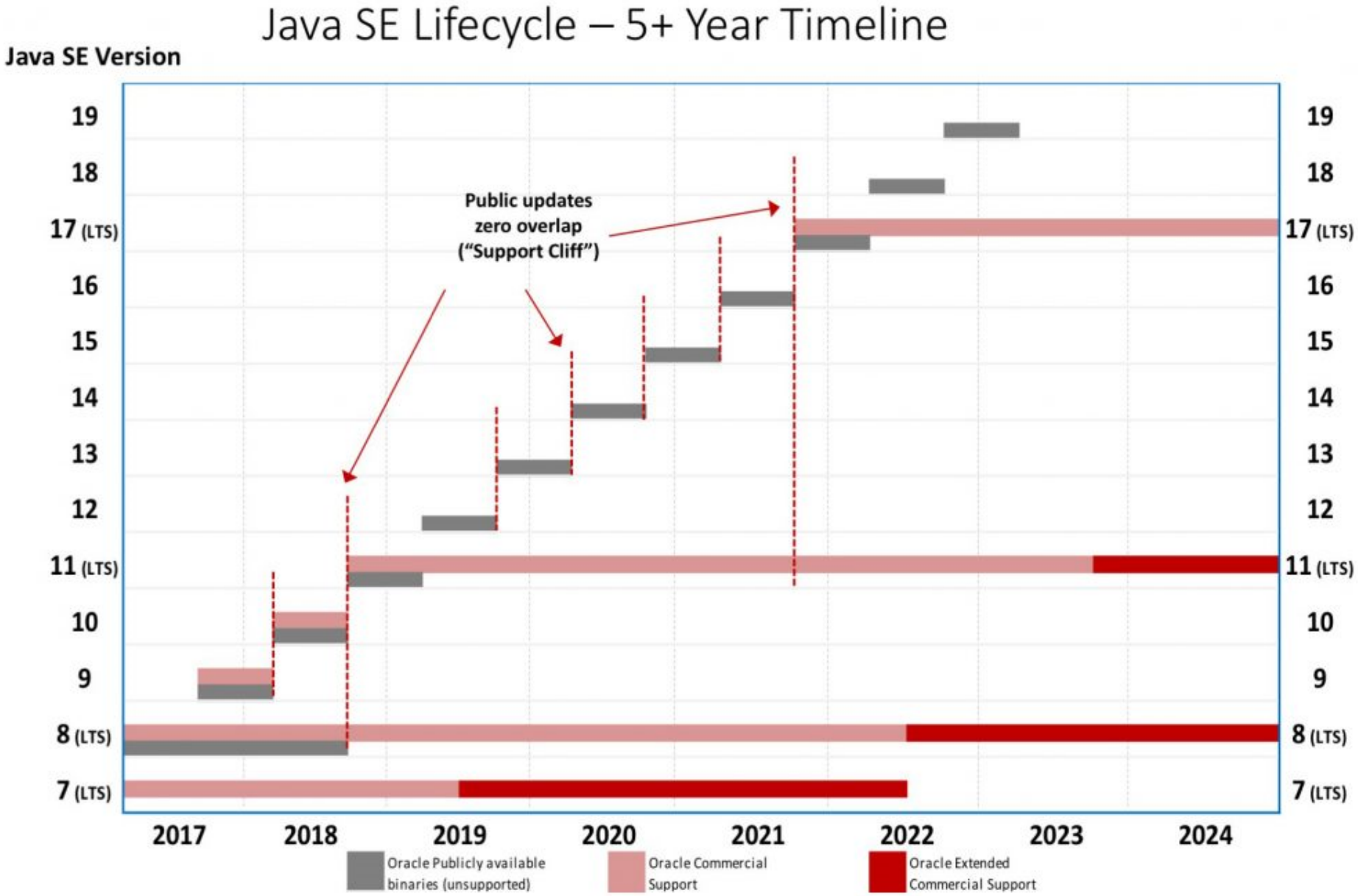
JRE (**Java Runtime Environment**) - Środowisko Uruchomieniowe Javy. W skład JRE wchodzi Wirtualna Maszyna Javy (JVM) + zbiór klas oraz narzędzi wymaganych do uruchomienia aplikacji wytworzonych w języku JAVA.

JVM (Java Virtual Machine) - Wirtualna Maszyna Javy. Środowisko zdolne do wykonywania skompilowanego kodu aplikacji (kod bajtowy Javy).

Interoperacyjność

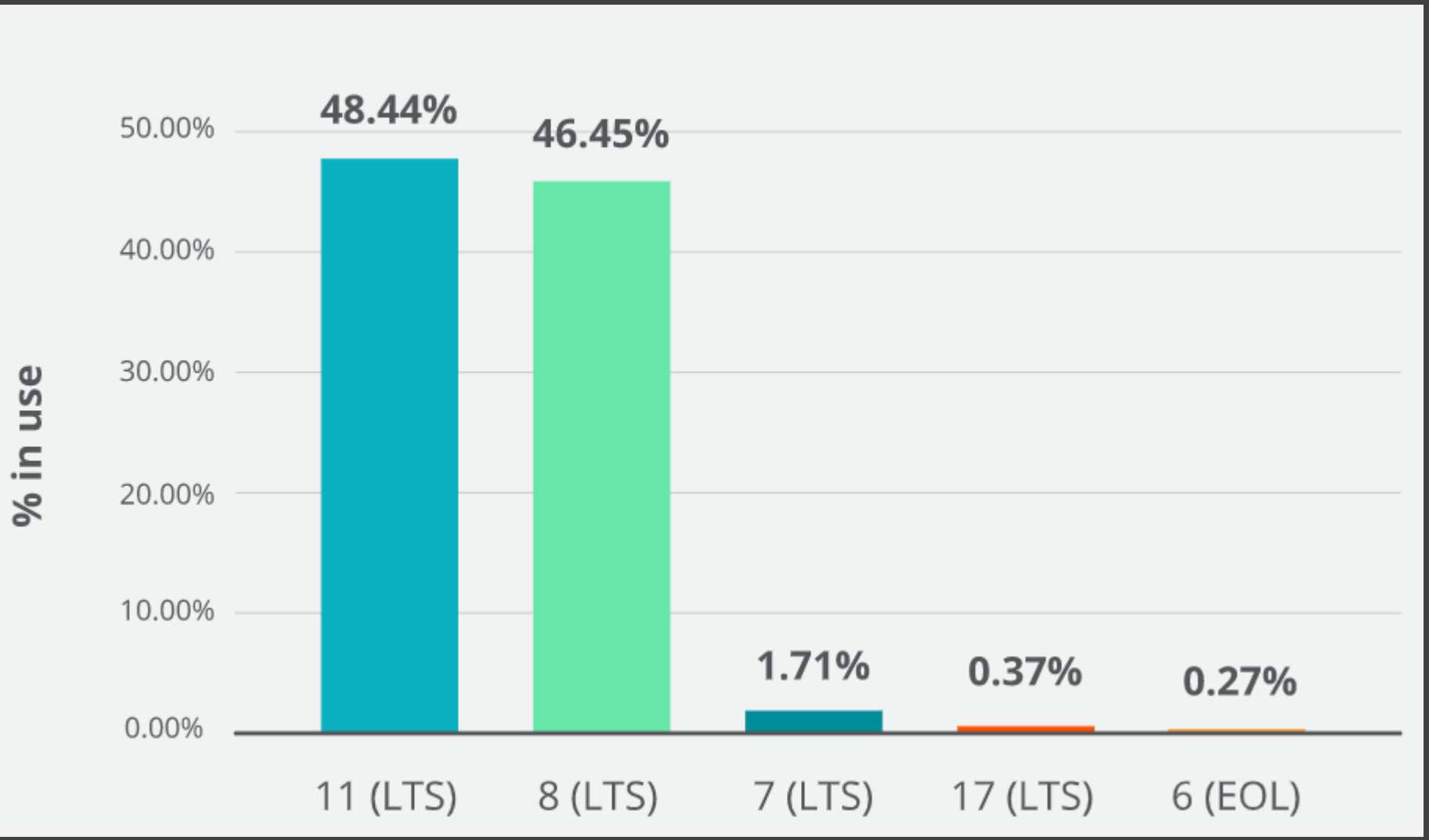


Java


Java SE Lifecycle – 5+ Year Timeline
Java SE Version
19
18
17 (LTS)
16
15
14
13
12
11 (LTS)
10
9
8 (LTS)
7 (LTS)
Public updates
zero overlap
("Support Cliff")
2017
2018
2019
2020
2021
2022
2023
2024
Oracle Publicly available binaries (unsupported)
Oracle Commercial Support
Oracle Extended Commercial Support

Uniwersytet Wrocławski

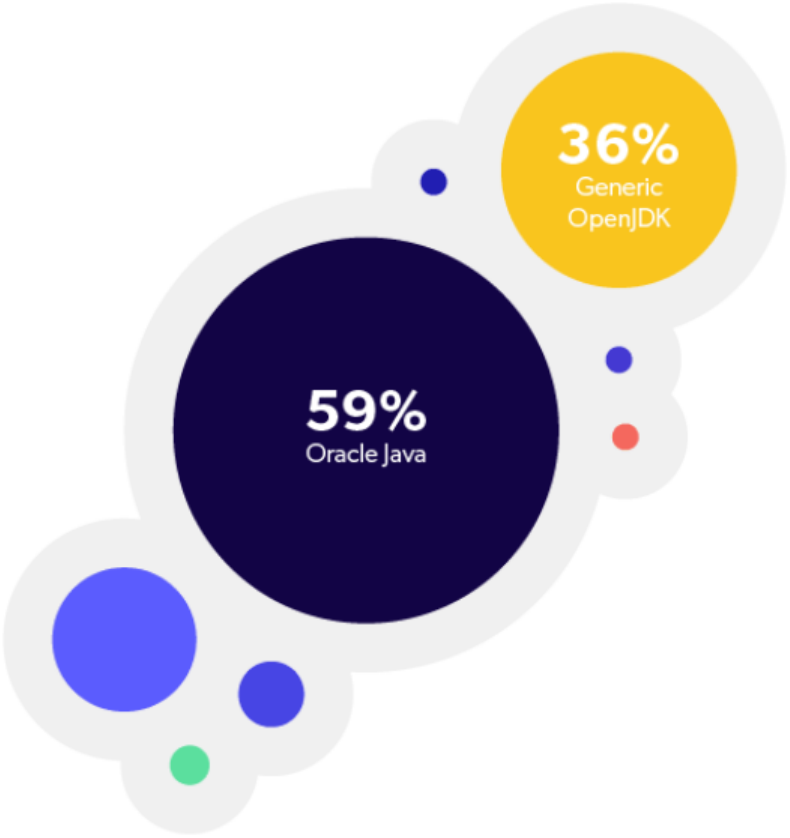
Java



https://newrelic.com/resources/report/2022-state-of-java-ecosystem

Uniwersytet Wrocławski

Java – JRE/JDK Distributions



Oracle Java **59%**

Generic OpenJDK **36%**

Adopt OpenJDK **22%**

Amazon Corretto **10%**

Azul Zulu **6%**

GraalVM **4%**

OpenLogic JDK **4%**

Other **4%**

https://www.jrebel.com/blog/2021-java-technology-report

Java

- Niektóre podobieństwa do C/C++
  - Styl pisania komentarzy
  - Wiele słów kluczowych jest identycznych
  - Typy danych
  - Niemal identyczna struktura kodu
- Niektóre różnice
  - Java wprowadza wiele słów kluczowych nieznanych z C/C++
  - Występują operatory nieznane w C/C++
  - Nie występuje przeciążanie operatorów
  - Brak niektórych cech języka np. wskaźniki
  - Niektóre cechy języka są zmodyfikowane w porównaniu do C/C++ - pętle musza być kontrolowane przez wyrażenia logiczne

Kotlin

- Cechy języka
  - Statyczne typowanie danych – na etapie kompilacji typy wyrażeń są znane a kompilator sprawdza czy istnieją pola i metody w obiektach do których odwołujemy się w kodzie
  - Domniemanie typów – typ danych jest określany przez kompilator na podstawie kontekstu
  - Typy zerowalne
  - Typy funkcyjne
  - Klasy danych
  - Można łączyć programowanie obiektowe i funkcyjne

Kotlin vs Java

- Null Safety

val number: Int? = null

- Extension Functions
- Coroutines Support
- Data Classes
- Functional Programming: Higher-Order Functions
- Primitive types
- Wildcard Types

ArrayList<?>

Uniwersytet Wrocławski

Gradle

```
plugins {
    id 'java'
}

group 'org.example'
version '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.7.2'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.7.2'
}

test {
    useJUnitPlatform()
}
```

Uniwersytet Wrocławski

Maven

pom.xml

```xml
    <modelVersion>4.0.0</modelVersion>
    <groupId>groupId</groupId>
    <artifactId>JavaMavenDemo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <configuration>
                    <archive>
                        <manifestFile>src/main/resources/META-INF/MANIFEST.MF</manifestFile>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```