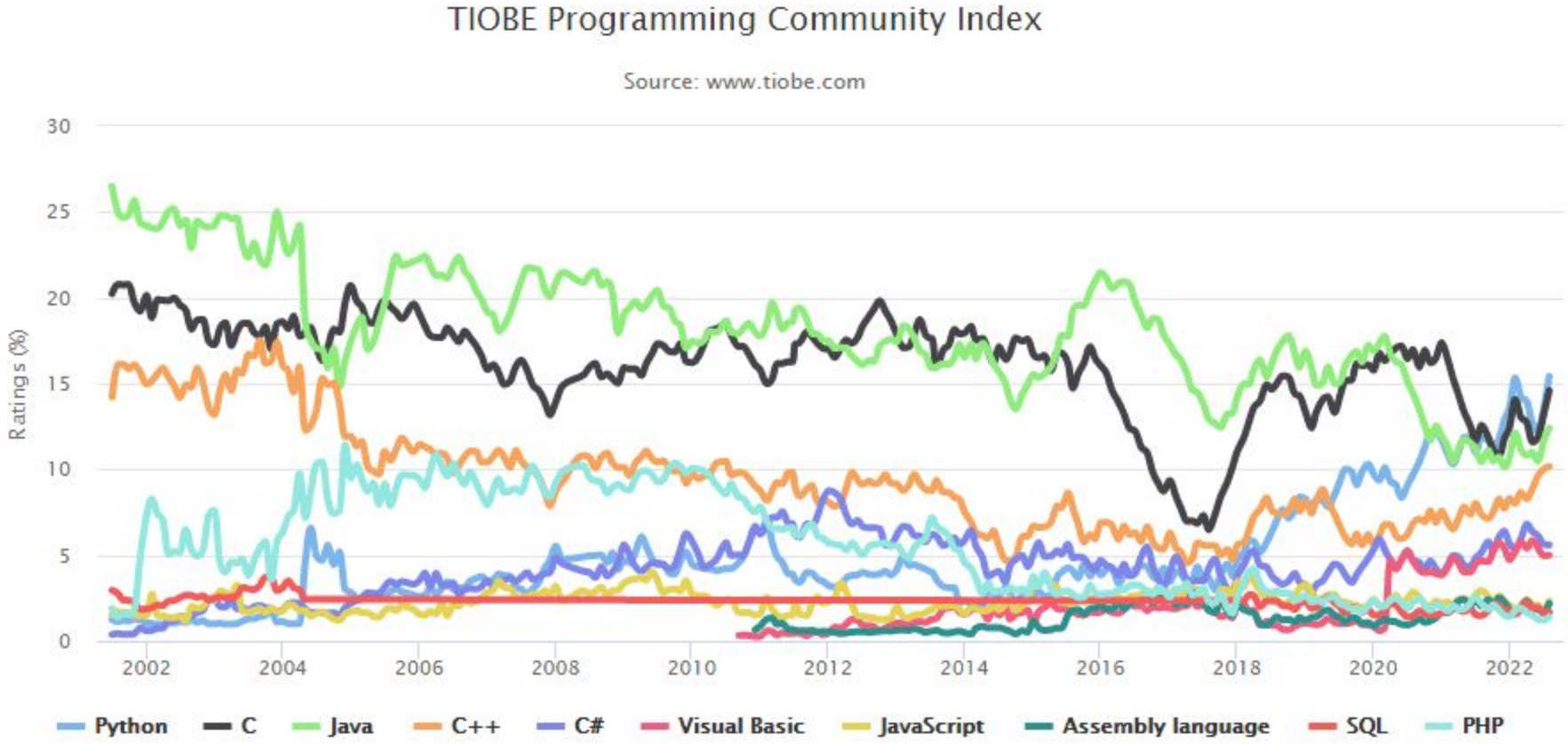
Java

Matthias Colin

Programmation Orienté Objets

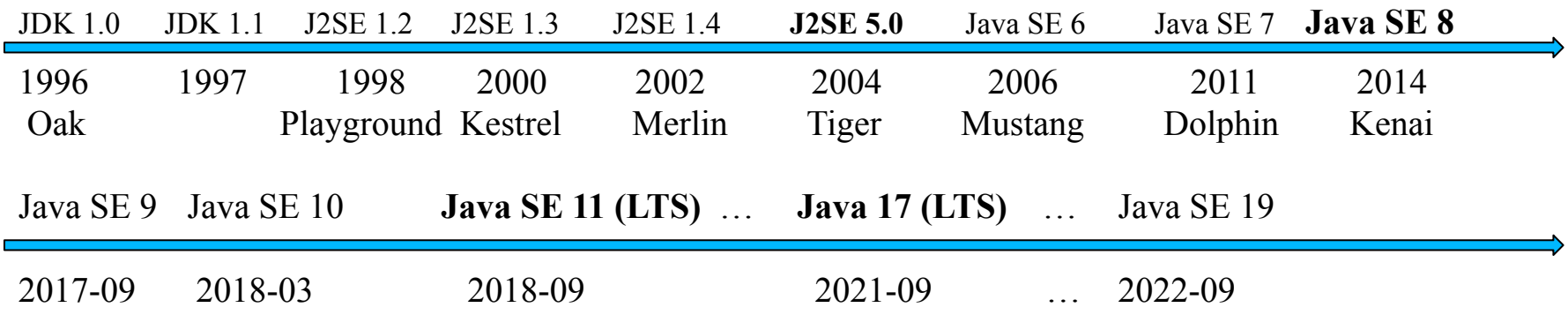
- POO / OOP
- C++
- Java
- .NET (C#, VB.NET, ..)
- Python
- JavaScript, TypeScript, Google App Script
- Php


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
Python
C
Java
C++
C#
Visual Basic
JavaScript
Assembly language
SQL
PHP

Java éditions

- Java SE (Standard Edition)
  - JVM : Java Virtual Machine
  - JRE : Java Runtime Environment (JVM + lib Java SE)
  - JDK : Java Development Kit (JRE + tools javac, jar, …)
  - Providers: Oracle et Openjdk
- Java ME (Mobile Edition)
- Java EE / Jakarta EE (Entreprise Edition)
  - collection de spécifications
  - servlet (http), jsp (Java Server Page), el, jstl
  - JPA (Java Persistence API) : RDBMS + SQL
  - JAX-WS : web services (SOAP+WSDL), axé XML
  - JAX-RS : API Rest avec data XML ou JSON
  - JNDI : annuaire
  - ...

Java SE


JDK 1.0
JDK 1.1
J2SE 1.2
J2SE 1.3
J2SE 1.4
J2SE 5.0
Java SE 6
Java SE 7
Java SE 8
1996
Oak
1997
1998
Playground
2000
Kestrel
2002
Merlin
2004
Tiger
2006
Mustang
2011
Dolphin
2014
Kenai
Java SE 9
Java SE 10
Java SE 11 (LTS)
…
Java 17 (LTS)
…
Java SE 19
2017-09
2018-03
2018-09
2021-09
…
2022-09

Java/Jakarta EE

JDK 1.0	JDK 1.1	J2SE 1.2	J2SE 1.3	J2SE 1.4	J2SE 5.0	Java SE 6	Java SE 7	**Java SE 8**	**Java SE 11**
1996	1997	1998	2000	2002	2004	2006	2011	2014	2018

	J2EE 1.2	J2EE 1.3	J2EE 1.4	JEE 5	**JEE 6**	**JEE 7**	**JEE 8**	**Jakarta EE**
	1999	2001	2003	2006	2009	2013	**2017**	**2018**
Servlets :	2.2	2.3	2.4	2.5	3.0	3.1	4.0	
JSP :	1.1	1.2	2.0	2.1	2.2	2.3		
JSTL :		1.0	1.1	1.2	-	-		
EL :				2.1	2.2	3.0		
JPA :				1.0	2.0	2.1	2.2	
Bean validation :				1.0		2.0		
Outils :	JDBC		WS	JAXWS	JAXRS		JSON	JSON-B
	JNDI			JSF		SAAJ	WebProfile	WebSocket

Vocabulary

- JVM: Java Virtual Machine
  - execute Java Bytecode
- JRE
  - JVM + libraries included in the language
- JDK
  - JRE + tools
  -

Tools

- java (JVM)
- javac (compiler)
  - *.java (source) => *.class (bytecode)
- jar (package)
  - inclus : bytecode, resource, jar
  - exemples:
    - appli.jar, library.jar
      - java -jar appli.jar
    - webapp.war (to be deployed in application server)
- javadoc (documentation)

Gestionnaire de projet

- Différents systèmes:
  - Ant (deprecated)
  - Maven : pom.xml
    - dépendances avec repository
  - Gradle : build.gradle
- Dépendances
  - Maven Repository
  - Maven Central

Gestionnaire de projet (2)

- Organisation projet (Maven et Gradle)
- MonProjet
  - pom.xml
  - src
    - main
      - java
      - resources
    - test
      - java
      - resources

IDE

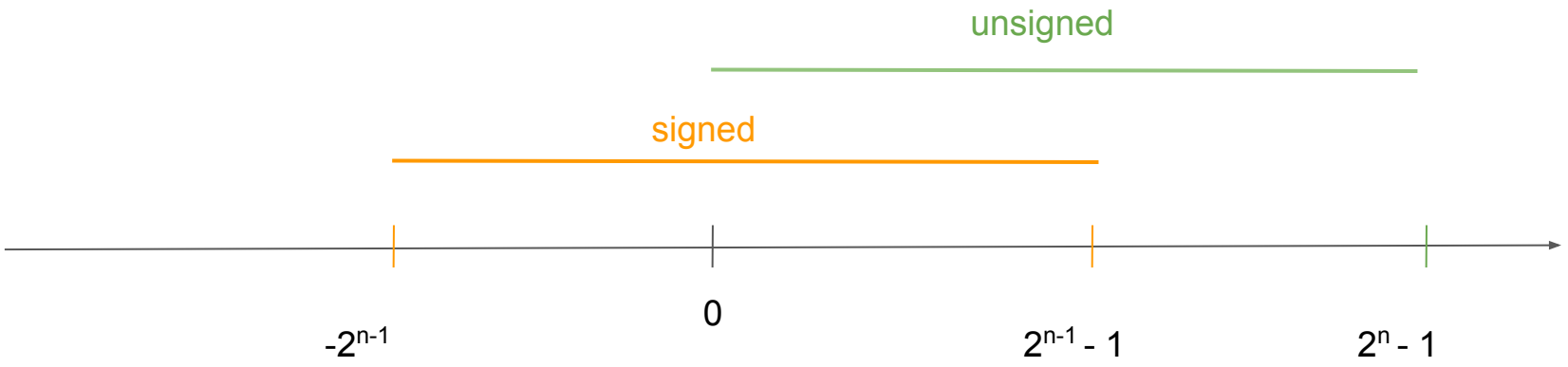
- IntelliJ Idea (JetBrains)
- Eclipse
- Netbeans

Variables et Opérateurs

- types primitifs
  - entiers : short, int, long : 4, -5 => +, -, *, /, %
  - flottants : float, double : 1.23, 3.45E24, NaN, Inf => +, -, *, /, %
  - boolean : true, false
  - char : 1 caractère : ‘A’
  - byte : 1 octet (donnée binaire)
- types objets
  - String : "Paris" => +
- opérateurs de comparaison
  - égalité : ==, !=
    - types primitifs : contenu
    - types objets : adresse mémoire (equals pour = de contenu)
  - ordre : <, <=, >, >= (primitifs)
- autres calculs numériques: class java.lang.Math

Integers

- integer stored in n bits: short (16), int (32), long (64)
- by default signed, can be interpreted as unsigned
  - n=16, signed -32768 to 32767 unsigned 0 to 65535
- 11111111 : 255 (unsigned), -1 (signed)



Fonctions et méthodes

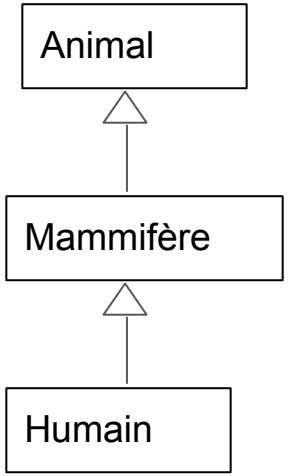
- rangée dans une classe (obligatoire)
- visibilité :
  - public : tout le monde
  - protected : package + classes filles
  - (no keyword) : package private
  - private : intérieur de la classe uniquement

Classes

- constructeur
  - par défaut implicite si aucun constructeur écrit
  - explicite(s)
- attributs
  - valeur par défaut
    - 0 pour les attributs numériques
    - false pour les booléens
    - null pour les objets
  - Encapsulation
    - attribut privé
    - getter et/ou setter
- lombock pour gérer getter/setter, constructeur, …

Héritage

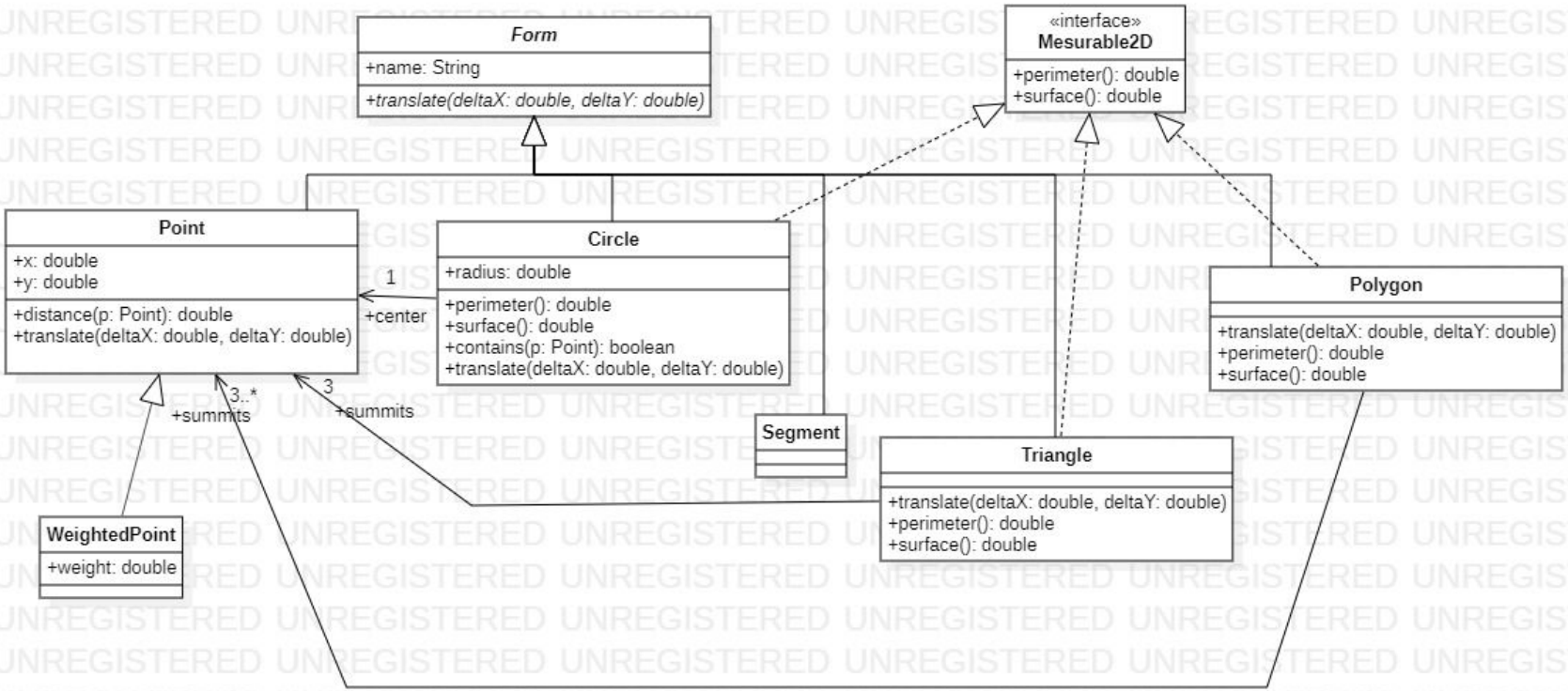
- 1 humain est un mammifère
- 1 mammifère **est** un animal
- 1 mammifère **peut être** un humain
- classe Humain hérite de la classe Mammifère
- classe Humain **spécialise** la classe Mammifère
- classe Mammifère **généralise** la classe Humain
- principe de substitution de Liskov & Wing (LSP)



Visibility

- private : UML - : only in this class
- : UML ~ : (package private) : only in this package
- protected : UML # : package + children classes
- public : UML + : everyone

Model geometry



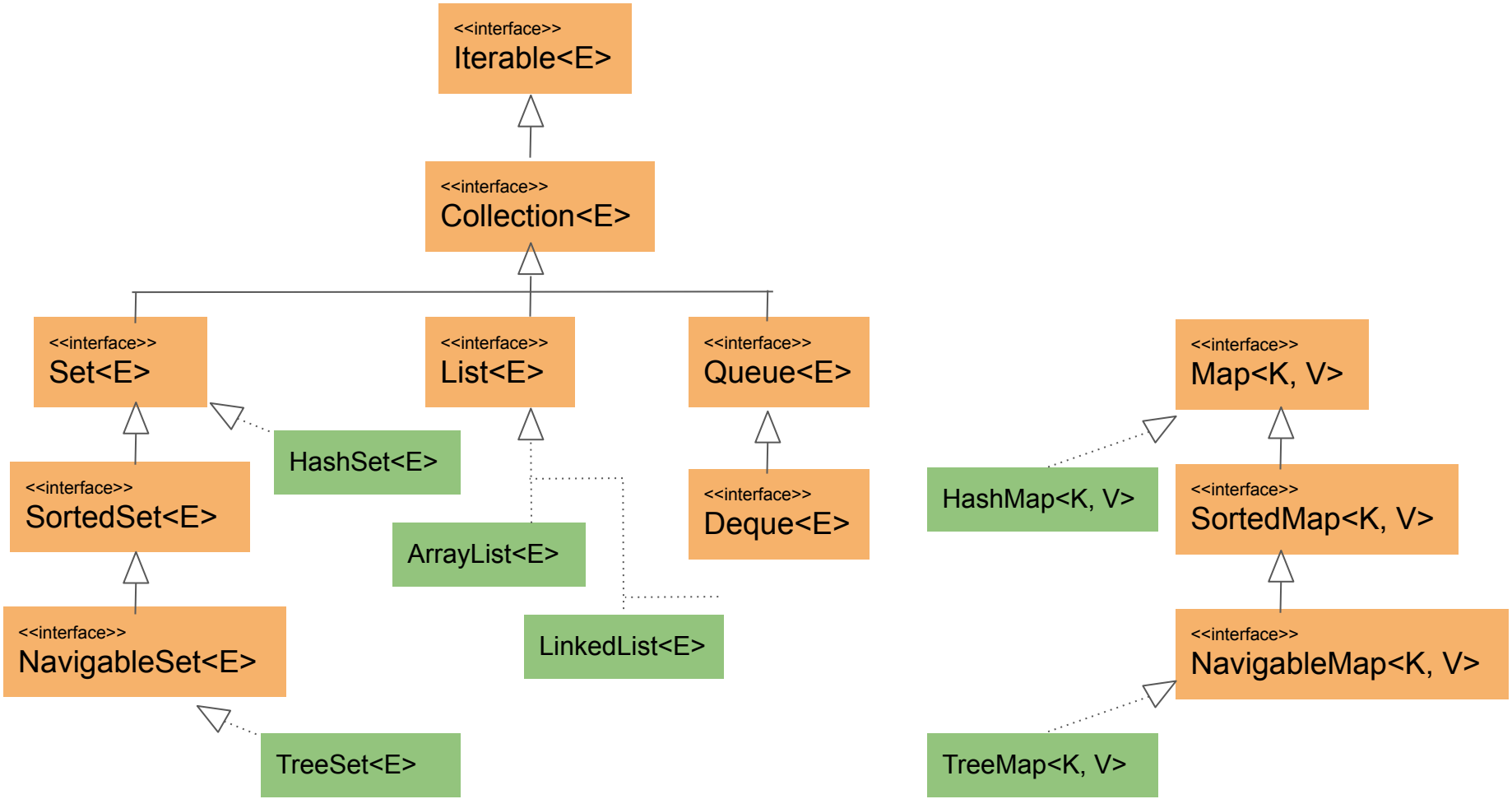
Autoboxing

Pour chaque type primitif => 1 type objet

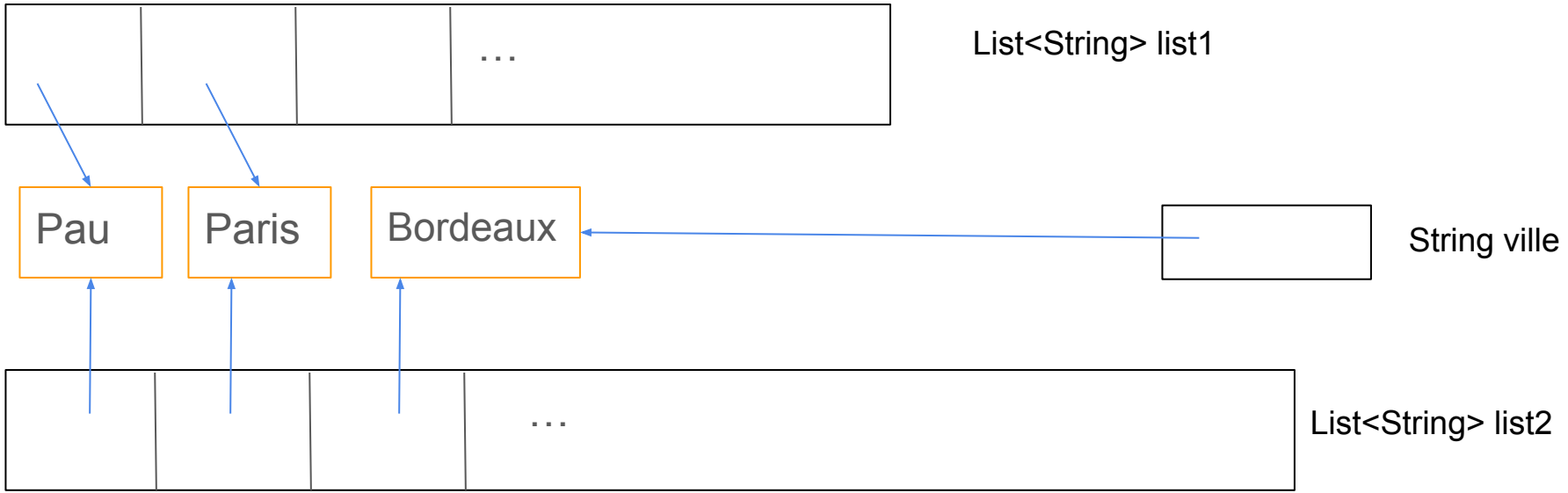
- short ⇔ Short
- int ⇔ Integer
- long ⇔ Long
- float ⇔ Float
- double ⇔ Double
- boolean ⇔ Boolean
- char ⇔ Character

Array vs Collection

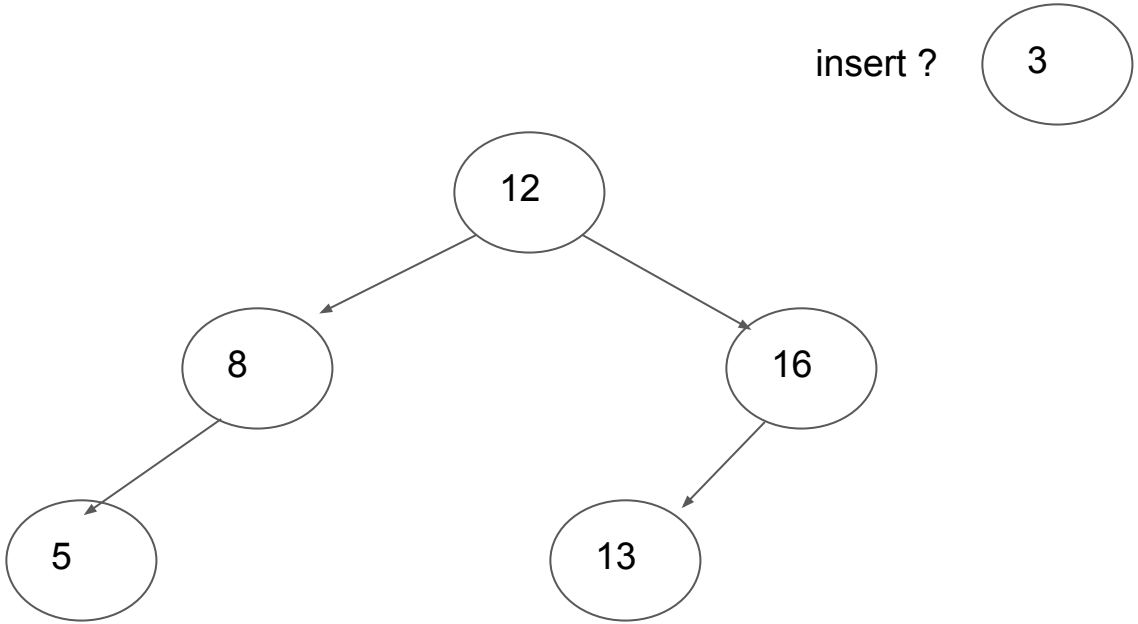
- Array : String[ ] villes
  - taille fixe
- Collection<E> : Collection<String>, Collection<Double>, Collection<Plane>
  - taille dynamique (en général)
  - List<E> : éléments rangés avec un index 0, 1, …, size-1
  - Set<E> : pas de doublons
    - SortedSet<E>, NavigableSet<E> : éléments triés
- Map<K,V> : données indexées
- Type interface et plusieurs implémentations possibles
  - List<E> => ArrayList<E>, LinkedList<E>, Vector<E>, …


<<interface>>
Iterable<E>
<<interface>>
Collection<E>
<<interface>>
Set<E>
<<interface>>
List<E>
<<interface>>
Queue<E>
HashSet<E>
<<interface>>
SortedSet<E>
ArrayList<E>
<<interface>>
Deque<E>
<<interface>>
NavigableSet<E>
LinkedList<E>
TreeSet<E>
<<interface>>
Map<K, V>
HashMap<K, V>
<<interface>>
SortedMap<K, V>
<<interface>>
NavigableMap<K, V>
TreeMap<K, V>

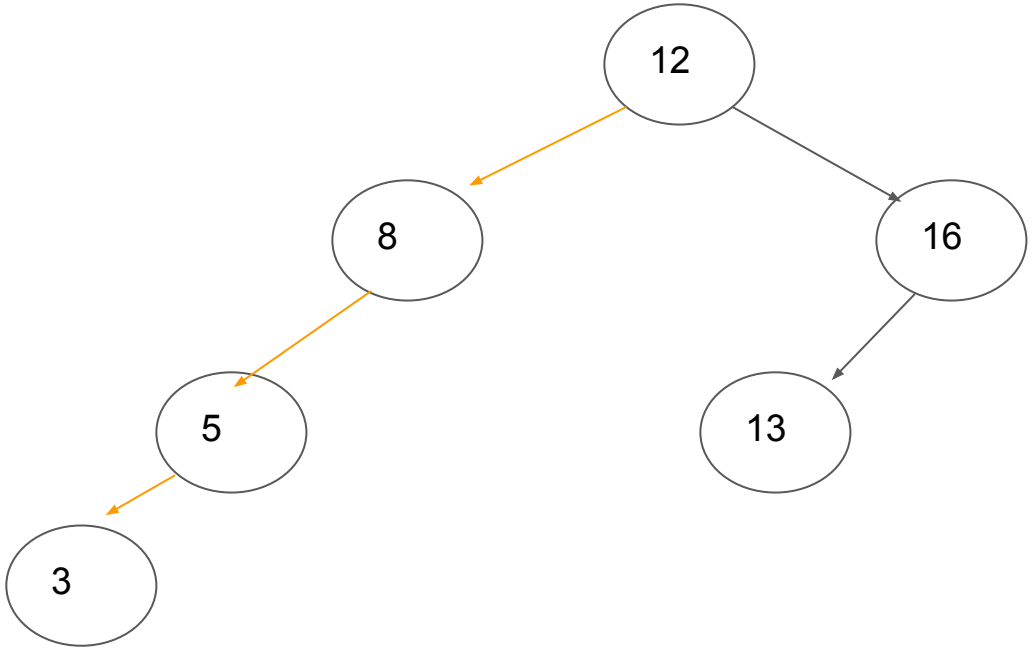
Partage de reference



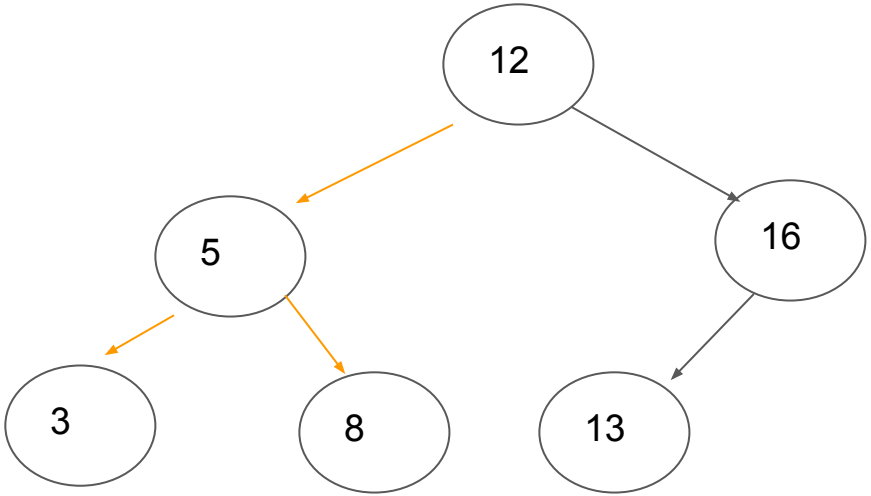
TreeSet



TreeSet (2)



TreeSet (3) : rééquilibrage



Stream: pipeline map/reduce

Etapes

- Source : Collection, JPA repository (SQL), Generator, …
- Intermédiaire(s)
  - map(f) : transformation de chaque donnée avec la fonction f
  - filter(p) : garde que les données respectant le prédicat p
  - peek() : jeter un coup
  - limit(n), skip(n) : coupe le stream après n valeurs ou les premières n valeurs
- Finale
  - Object
    - collect : toList, toCollection, stats, …
    - forEach : print, save file, insert bdd (result void)
    - findFirst
  - entiers/flottants
    - sum, min, max, avg, statisticts
    - reduce

Stream: objects vs primitive type

- Stream<Double>, Stream<Integer>, Stream<Long>
  - les données sont traitées en mode objet et allouées en mémoire (heap)
- DoubleStream, IntStream, LongStream
  - les données sont traitées sans allocation dynamique, uniquement dans le stack

Functional type = interface with one method

Comparator<T> : T x T -> int

Function<T,R> : T -> R

    BiFunction<T,U,R> : T x U -> R

Supplier<T> : () -> T

Consumer<T> : T -> void

    BiConsumer<T,U> : T x U -> void

Predicate<T> : T -> boolean

    BiPredicate<T,U> : T x U -> boolean

UnaryOperator<T> : T -> T

BinaryOperator<T> : T x T -> T

Fonctions et types fonctionnels

- 1 type fonctionnel :
  - 1 interface avec une seule méthode à implémenter
  - annotée avec @FunctionalInterface (pas obligatoire)
  - anciennes interfaces : Comparator, ActionListener
  - nouvelles interfaces : package java.util.function
    - Function<T,R> : T -> R
      - UnaryOperator<T> : T -> T
      - Predicate<T> : T -> boolean
    - Consumer<T> : T -> void
    - Supplier<T> : () -> T
    - BiFunction<T,U,R> : T x U -> R
      - BinaryOperator<T> : T x T -> T
      - BiPredicate<T> : T x T -> boolean
    - BiConsumer<T, U> : T x U -> void
    - + toutes les variantes avec types primitifs : IntFunction, ToIntFunction, ….

Comparable, Comparator, Sort

- par défaut, un objet n’est pas comparable
- le sont:
  - types primitifs (<) ou via leur type objet correspondant
    - exemple: int et Integer
  - String
  - données temporelles
- interface Comparable<E>
  - définit un ordre naturel pour le type E
  - méthode: int compareTo(E other)
  - Exemple:
    - int cmp = a.compareTo(b)
    - cmp < 0 : a < b
    - cmp = 0 : a = b
    - cmp > 0 : a > b

Comparable, Comparator, Sort (2)

- interface Comparator<T>
  - méthode int compare(T t1, T t2)
  - même sémantique que compareTo sur le résultat
  - exemple
    - Comparator<String> comparator = ???
    - int cmp = comparator .compare(“Nancy”, “naNTes”)

Inheritance and genericity

```
static <T>
boolean addAll(Collection<? super T> c, T... elements)

static <T extends Object & Comparable<? super T>>
T max(Collection<? extends T> coll)

static <T>
T max(Collection<? extends T> coll, Comparator<? super T> comp)
```

static <T>

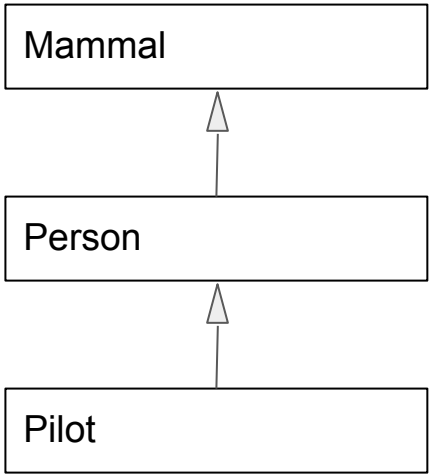
boolean addAll(Collection<? super T> c, T... elements)

static <T extends Object & Comparable<? super T>>

T max(Collection<? extends T> coll)

static <T>

T max(Collection<? extends T> coll, Comparator<? super T> comp)



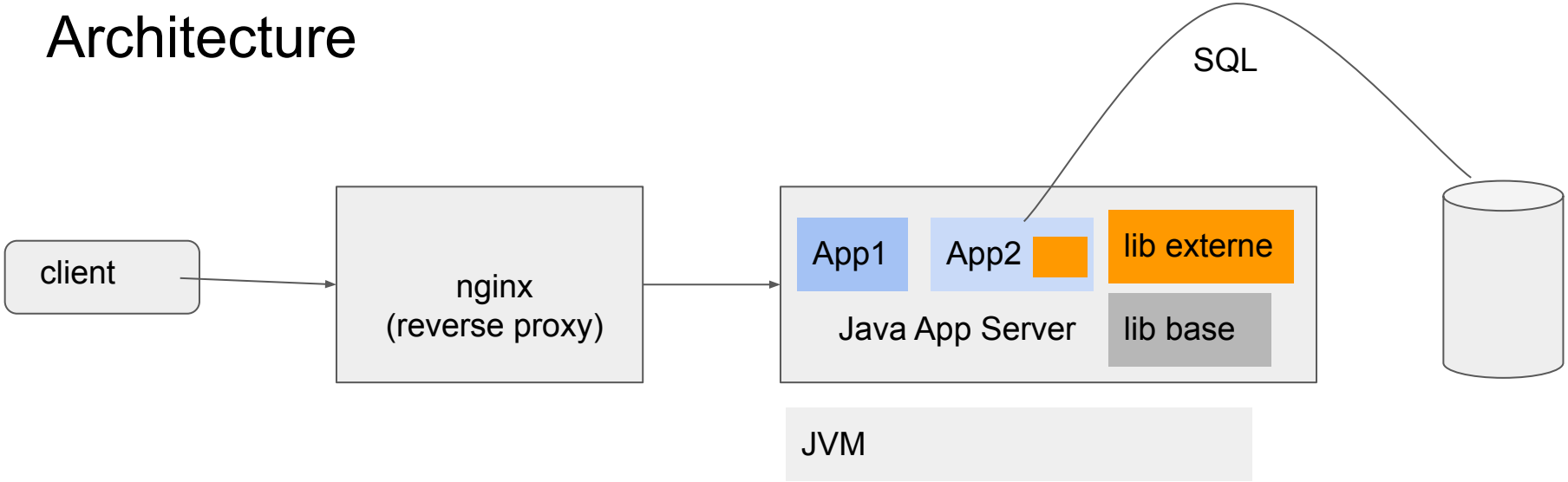
Application Java Backend

- Webapp MVC (Model View Controller)
- API Rest (Web Services)
  - micro-services
  - framework Spring
- Plateforme d'exécution (Java EE, JEE, Jakarta EE)
  - JVM : java / java.exe
  - Serveur Application Java
    - Tomcat (JEE Profile Web)
    - Full JEE:
      - JBoss / WildFly (RedHat)
      - Oracle WebLogic
      - IBM WebSphere

Exemple

- Développement d'un application avec
  - Java SE 11
  - servlet (spec JEE)
  - JPA (spec JEE)
  - dependance spring
- Déploiement & exécution
  - JRE 11
  - serveur application Java
    - Profile Web JEE : tomcat + jar JPA + conf + jar spring
    - Full JEE :
      - RedHat Wildfly + jar spring

Architecture



Wildfly

- https://www.wildfly.org/
- modes
  - standalone : 1 process java
  - domain : plusieurs processus java
    - 1 domain controller (process controller)
    - 1 host controller par host/machine
    - server(s)
- interfaces
  - public : 8080, 8443
  - management : 9990
- HAL: appli web d’administration
- jboss-cli : client en ligne de commande d’administration

Déploiement d’une webapp

http://192.168.56.106:8081/bonjour/index.html

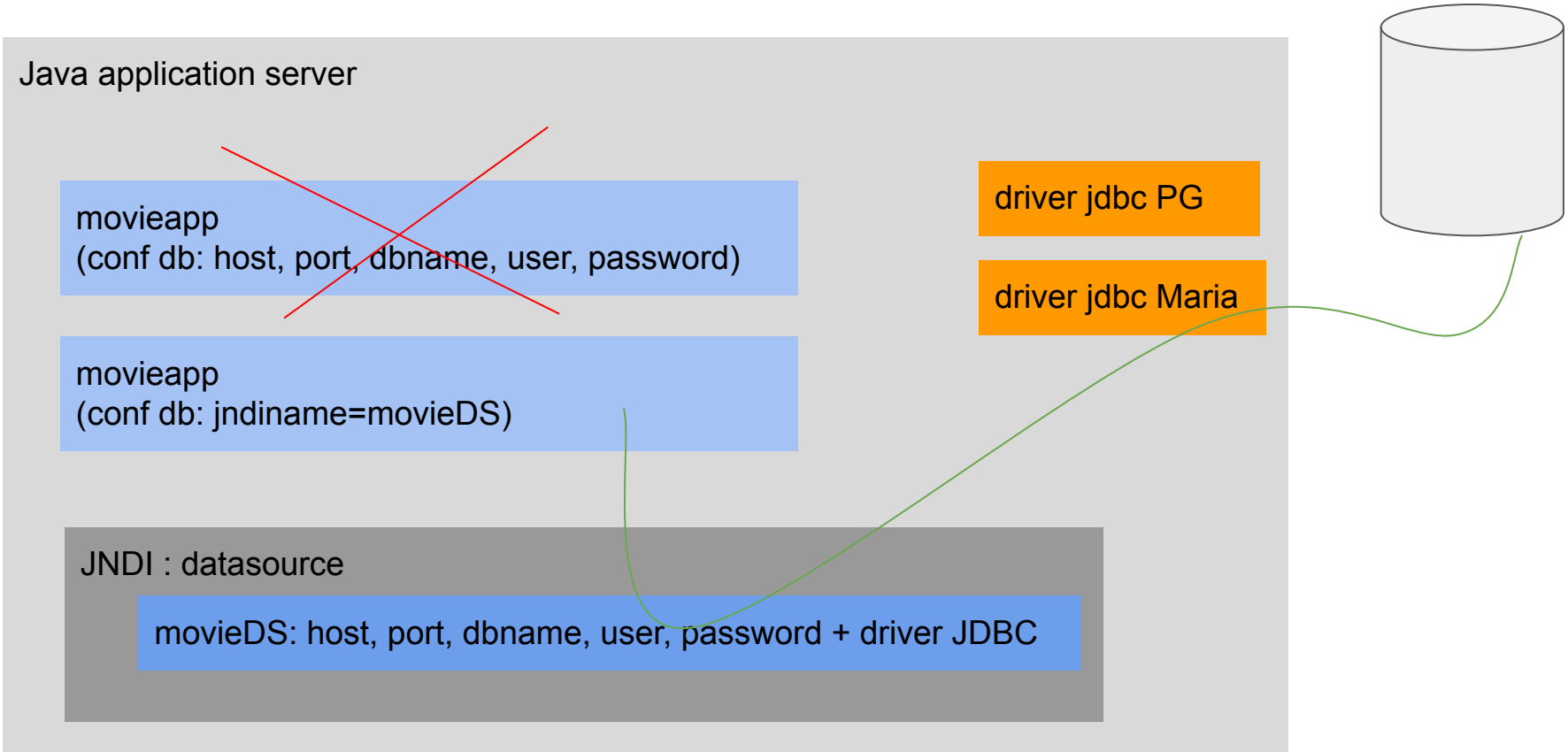
http://192.168.56.106:8081/bonjour/Goodbye

- 192.168.56.106 : hostname ou ip serveur
- 8081 : port pour atteindre le serveur
- /bonjour : contexte de l‘application
- /index.html : ressource publique (html, css, image, jsp)
- /Goodbye : ressource privé routée (servlet, api rest, …)

Java application with Relational Database

- communication appli Java <-> RDBMS
- langage commun de communication SQL
- JDBC : Java Database Connectivity (inclus Java SE)
  - Comment gérer des requêtes (insert, update, delete, select)
  - package java.sql et javax.sql
    - Driver : spécification d'un driver éditeur
    - Connection : établir une connexion avec la base de données
      - host, port, dbname, user (, password)
    - DataSource : pool de connexion(s)
    - Statement : exécuter une requête
      - select * from movies where year = 2020
    - PreparedStatement : exécuter une requête préparée
      - select * from movies where year = ?
      - paramètre #1 pourra être 2020, 2021, …
    - ResultSet : résultat d'une requête
  - Driver JDBC apporté par l'éditeur ou la communauté
    - postgresql-42.2.20.jar
- JNDI : externaliser les settings JDBC de l'appli => serveur appli

JNDI


Java application server
movieapp
(conf db: host, port, dbname, user, password)
movieapp
(conf db: jndiname=movieDS)
JNDI : datasource
movieDS: host, port, dbname, user, password + driver JDBC
driver jdbc PG
driver jdbc Maria

Configure JNDI Datasource

- tomcat: ajouter en XML l’entrée JNDI
  - http://tomcat.apache.org/tomcat-9.0-doc/jndi-datasource-examples-howto.html
- wildfly