JAVA

Installation

- JDK Installation
- https://www.oracle.com/in/java/technologies/javase-jdk15-downloads.html

POPULAR CODE EDITORS

- NetBeans
- Eclipse
- IntelliJ

“ANATOMY OF JAVA PROGRAM”

Function

A block of code that performs a task

```
ReturnType Name() {
    …
}
```

```
void Name() {
    …
}
```

```
void sendEmail() {
    …
}
```

```
void main() {
    …
}
```

A container for related functions

```
class Main {
    void main() {
        …
    }
}
```

```
public class Main {
    public void main() {
        …
    }
}
```

PascalNamingConvention — **Classes**

camelNamingConvention — **Methods**

“HOW JAVA CODE GET EXECUTES”

Compilation

Execution


Source Code
Java Compiler
Byte Code
*.java
*.class

JRE

- JRE: Java Run Time Environment
- https://www.oracle.com/java/technologies/javase-jre8-downloads.html
- JRE is a software component called Java Virtual Machine (JVM) that takes byte code and translate into to native code for the underlying operating system
- cd src folder → java com.company.**Main**

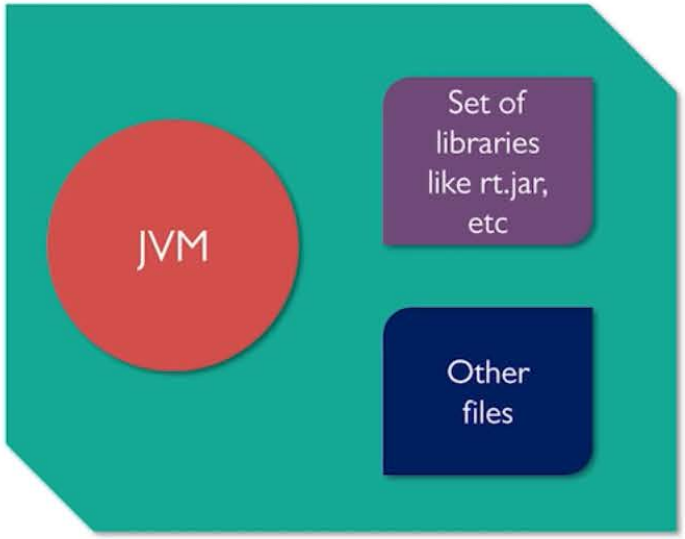

Byte Code
Java Virtual Machine
Native Code
*.class
Windows, Mac, ...

Java Runtime Environment (JRE)

- *Java Runtime Environment (JRE)* provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language

JRE = JVM+ Set of libraries+ Other Additional files

- The JRE does not contain tools and utilities such as compilers or debuggers for developing applets and application



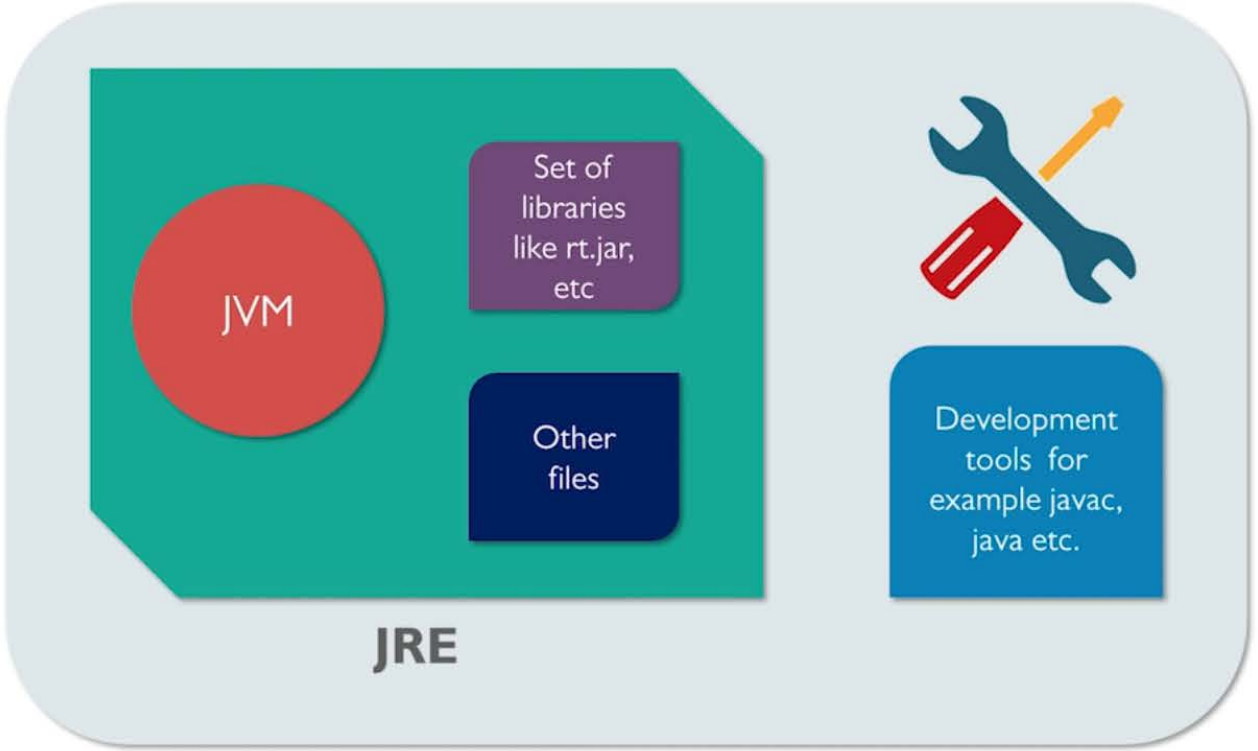
JRE

Java Development Kit (JDK)

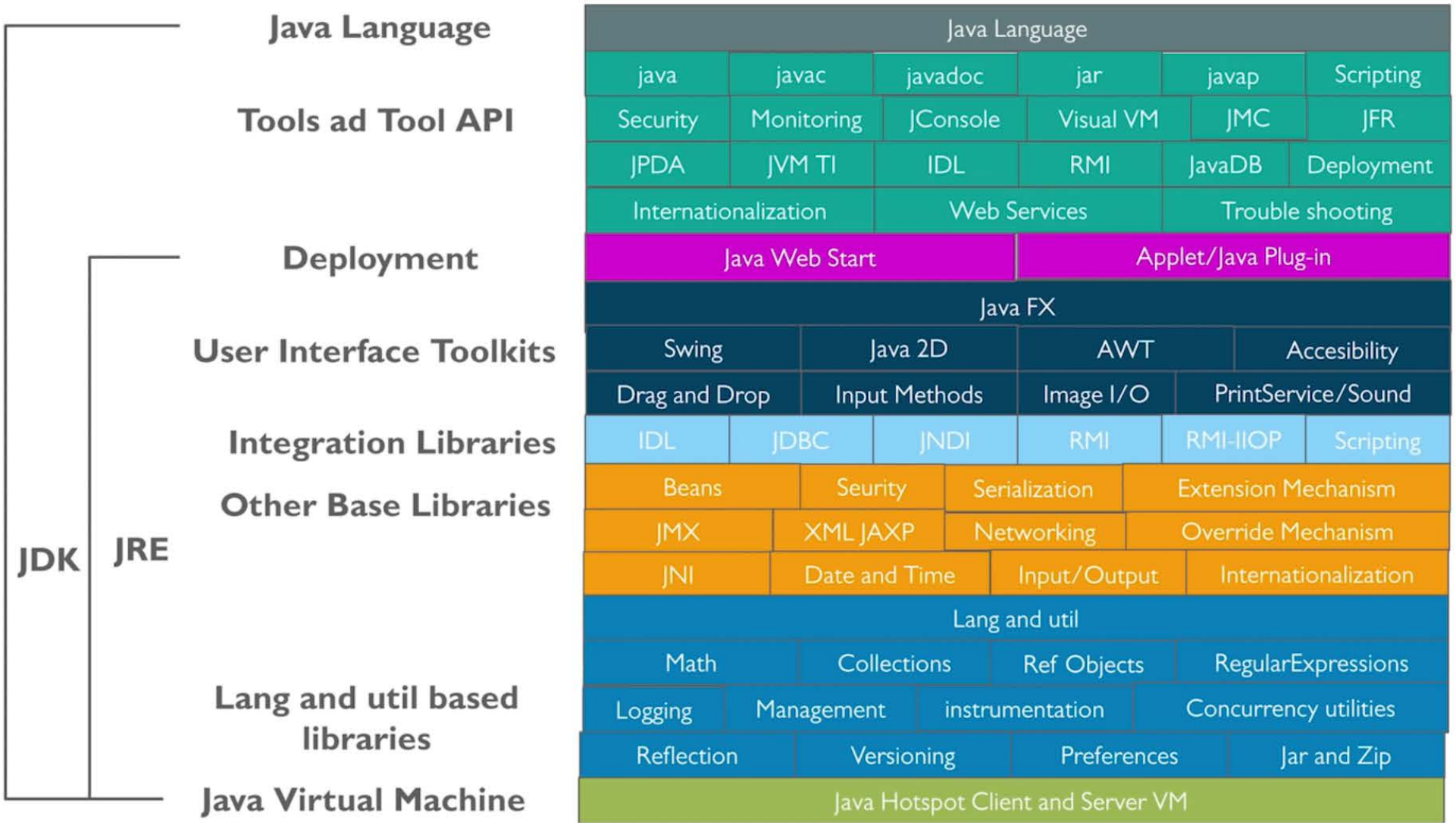
- *JDK* is a superset of the *JRE*, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications

JDK = JRE + Development Tools

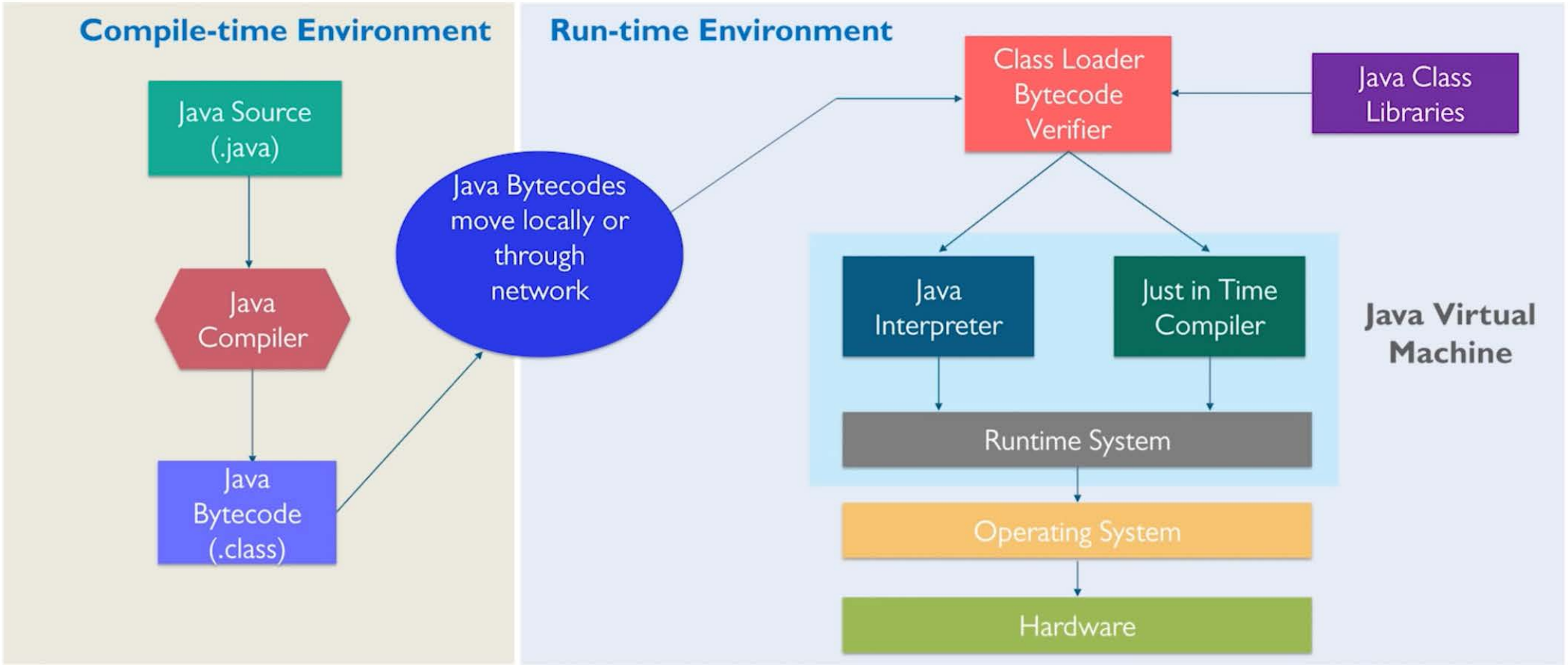
JDK = (JVM+ Set of libraries+ Other Additional files) + Development Tools



JDK, JRE & JVM – Deep Dive



Java Working



5 Interesting Facts about Java

Java was developed by James Gosling in 1995


Initially named it as OAK

Later named it as **GREEN**

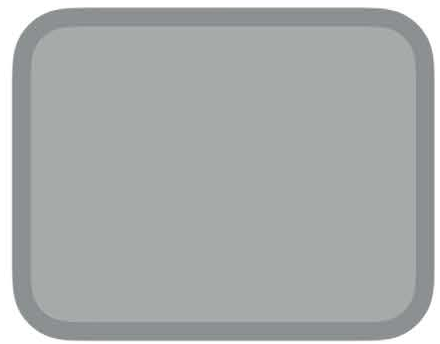

Finally named it as JAVA


Java™

Java Editions

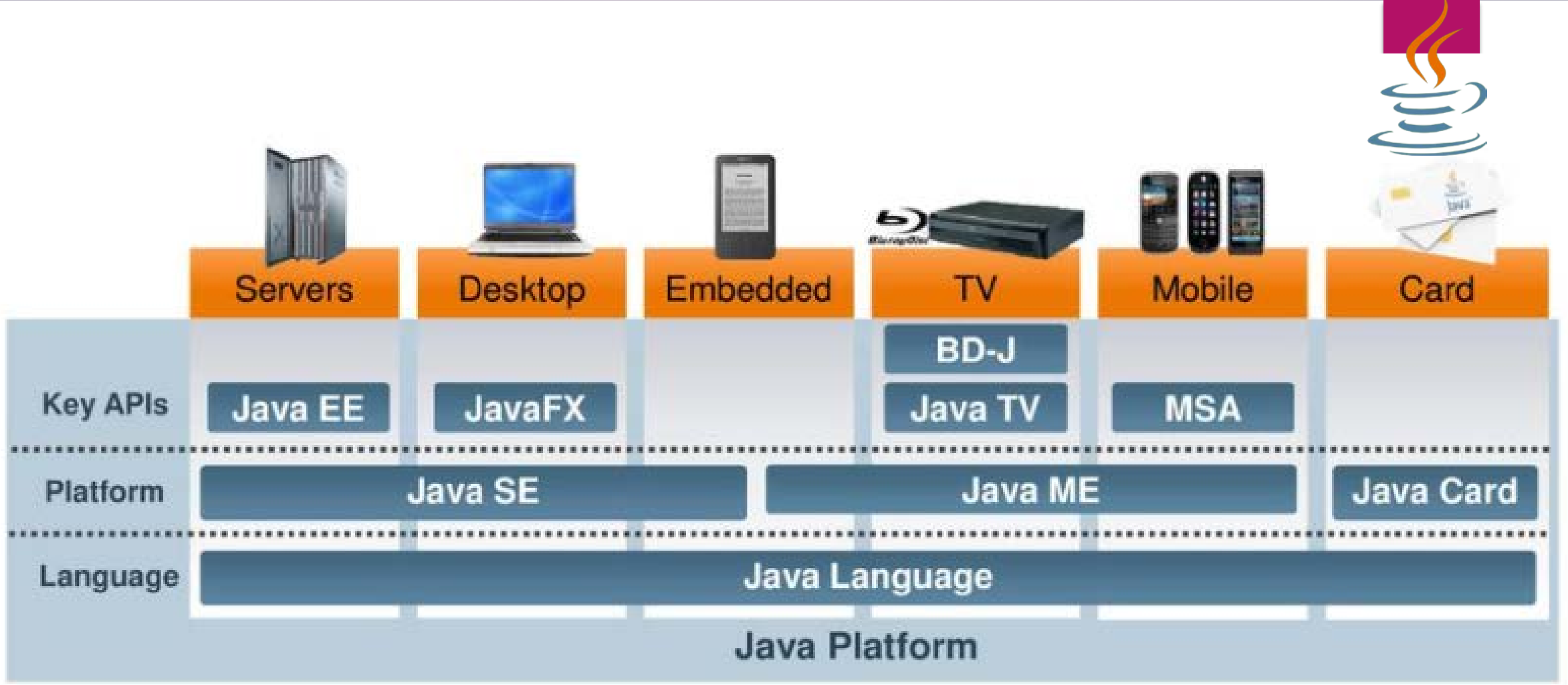
Standard
Edition (SE)

Enterprise
Edition (EE)

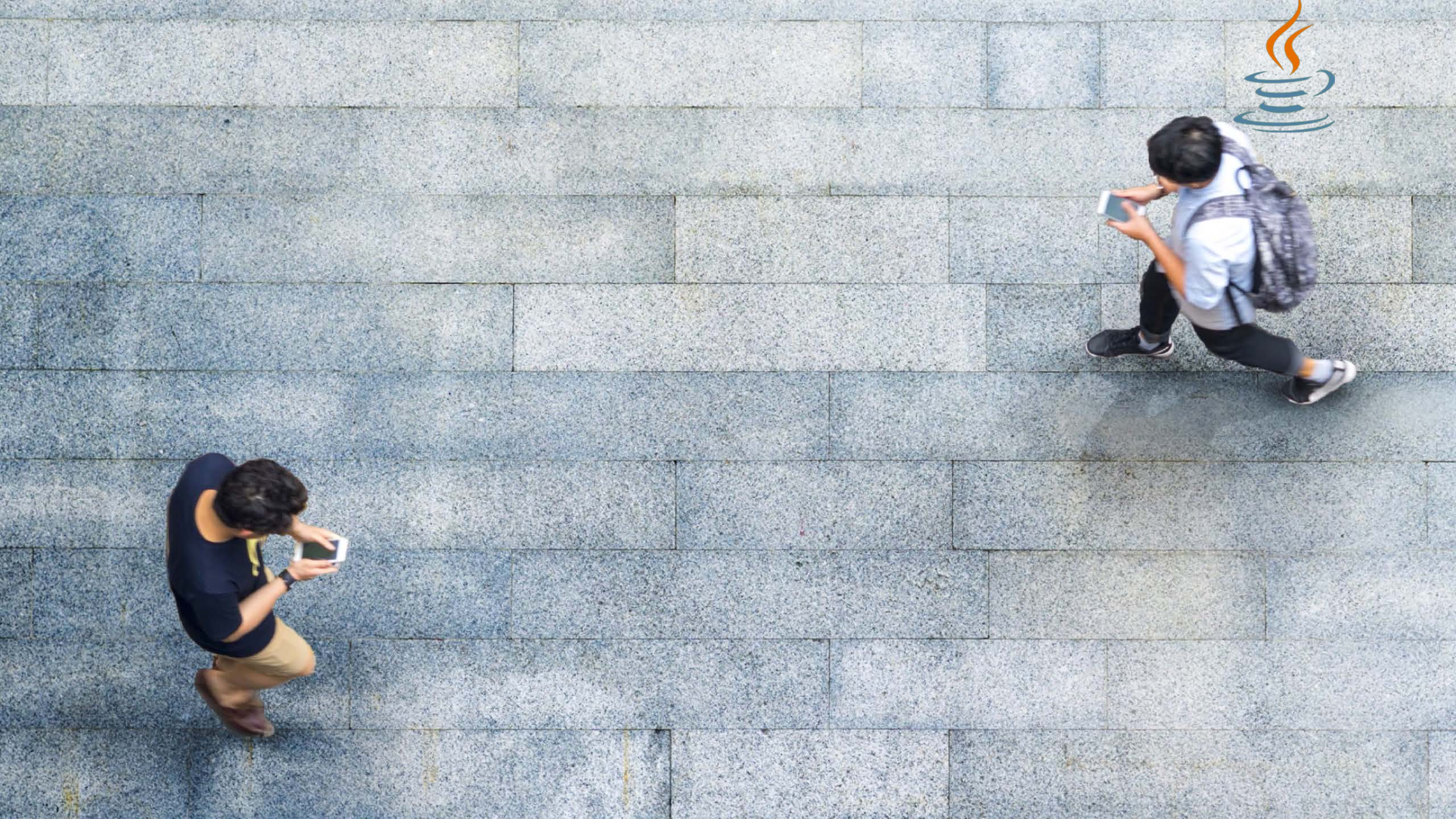


Micro
Edition (ME)

Java Card


Servers
Desktop
Embedded
TV
Mobile
Card
Key APIs
Java EE
JavaFX
BD-J
Java TV
MSA
Platform
Java SE
Java ME
Java Card
Language
Java Language
Java Platform


9 Million Developers



What Is Java?

Java is a computer programming language that enforces an **object-oriented** programming model

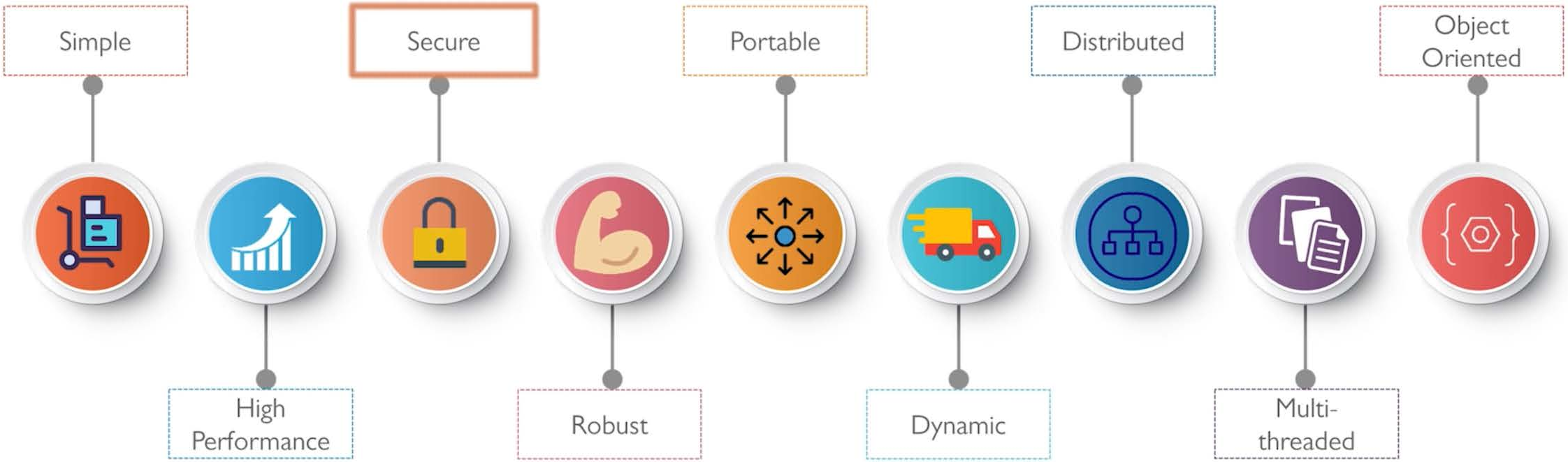
Java is a programming language and computing platform first released by **Sun Microsystems in 1995**

Java was created by a team lead by ***James Gosling***

Java is a platform independent programming language that follows the logic of **"Write once, Run anywhere"**

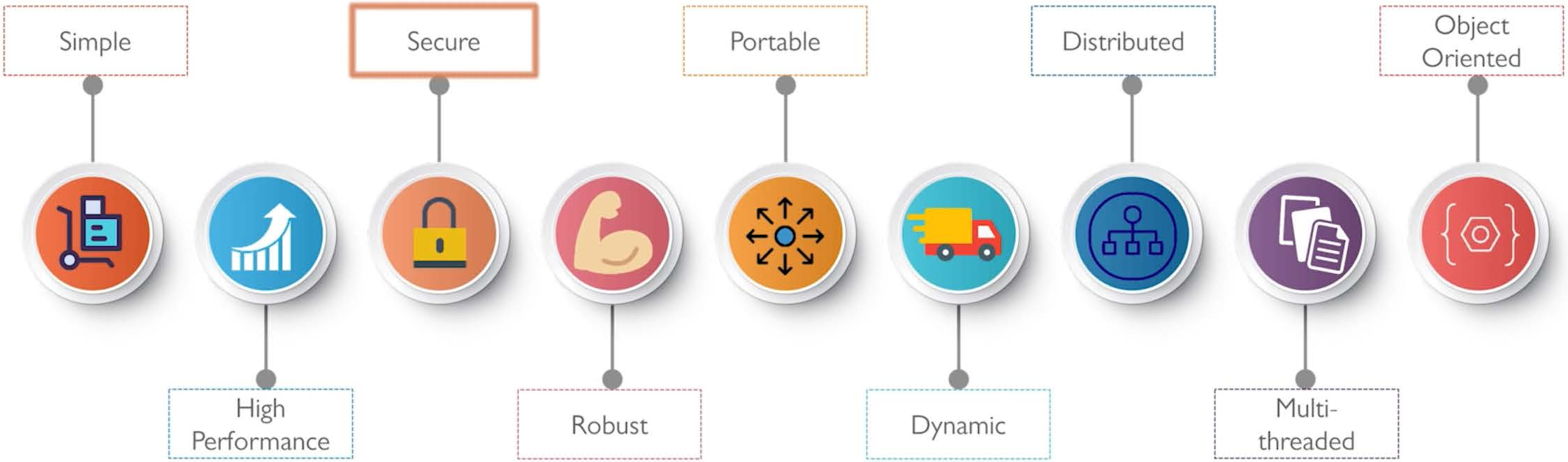
Java can be used to create complete applications that may run on a single computer or can be distributed among servers and clients in a network

Java Features



Although Java is an interpreted language, it was designed to support "just-in-time" compilers, which dynamically compile bytecodes to machine code

Java Features

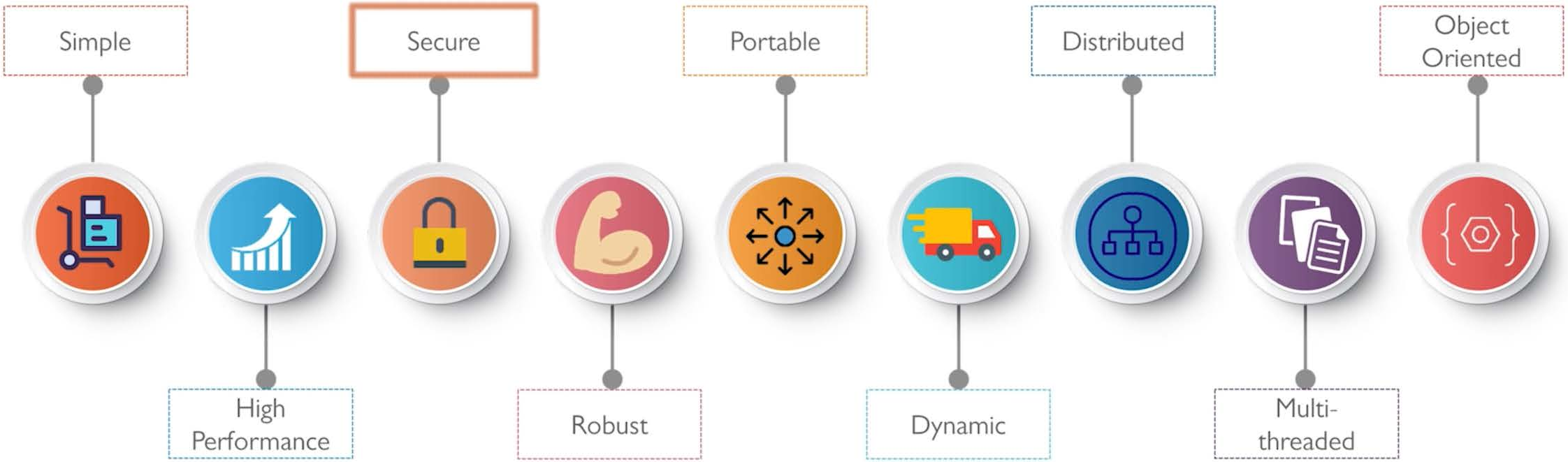


Java can be used to develop virus-free systems.

Java is secured because:

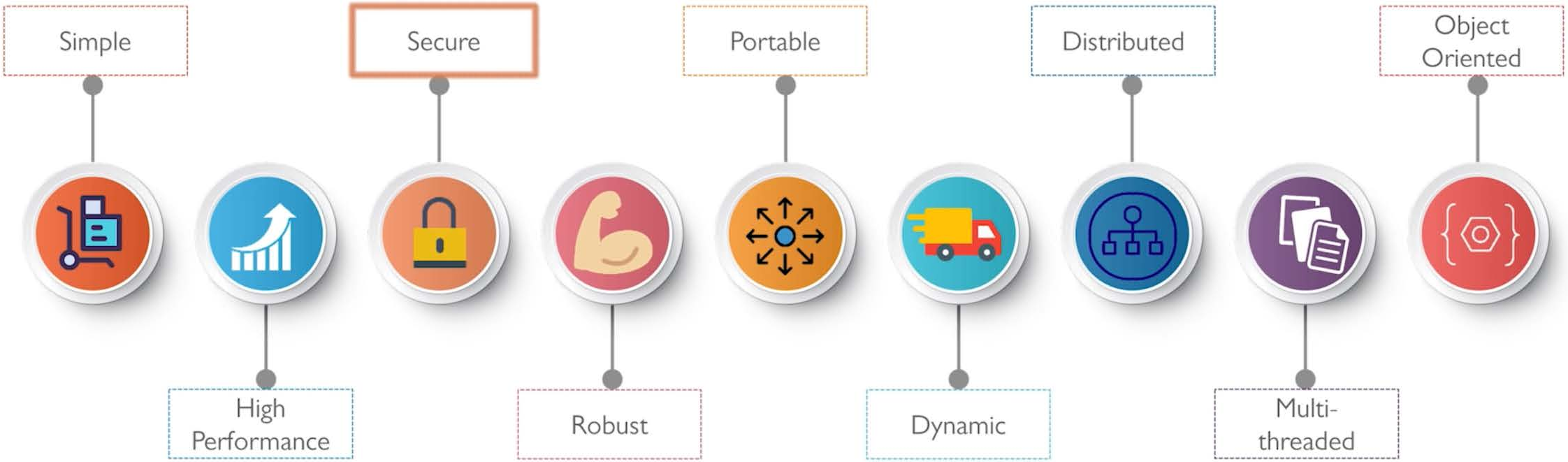
- Java Programs runs inside virtual machine sandbox to prevent any activity from untrusted sources
- No use of explicit pointers

Java Features



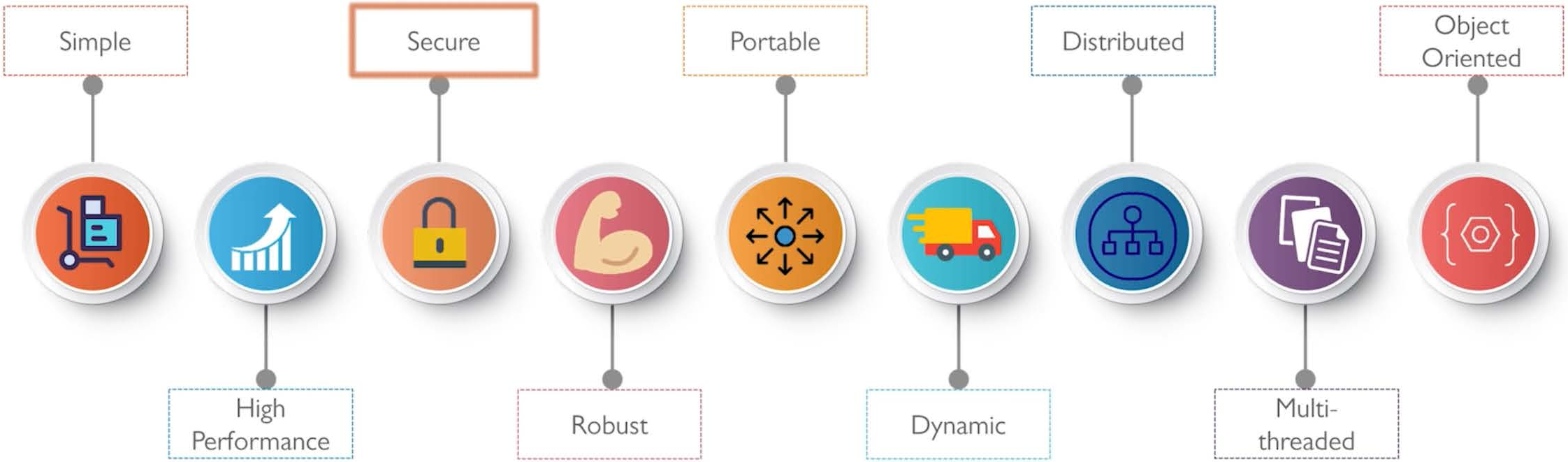
- Java checks the code during the compilation time and run time
- Java completely takes care of memory allocation and releasing, which makes Java more robust

Java Features



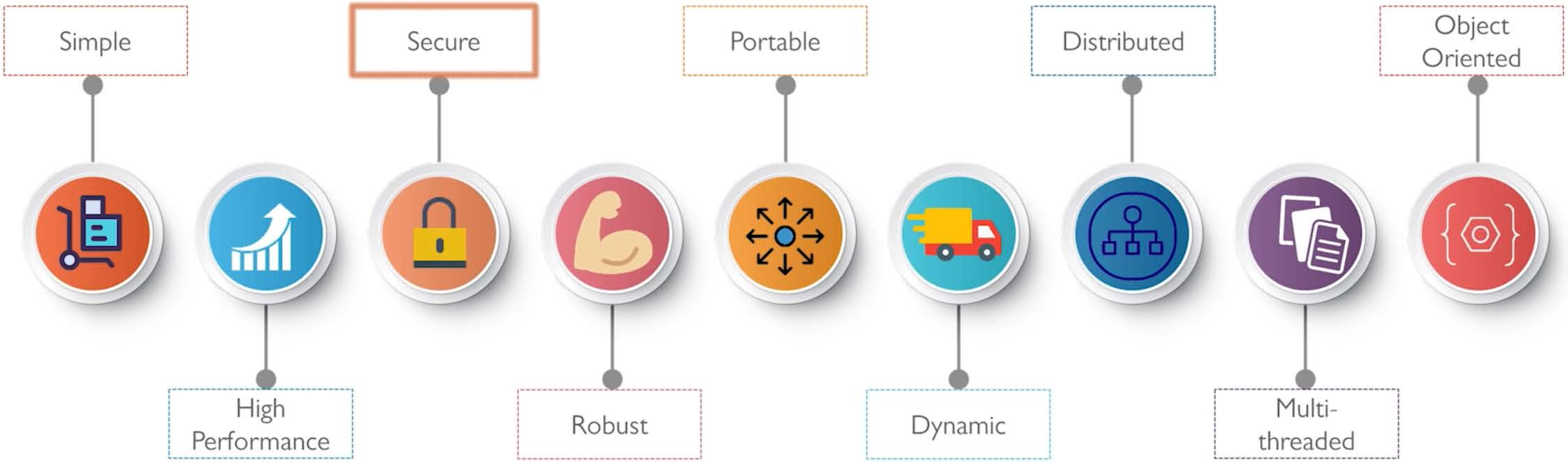
Applications written on one platform of Java can be easily ported to another platform as it is platform independent

Java Features



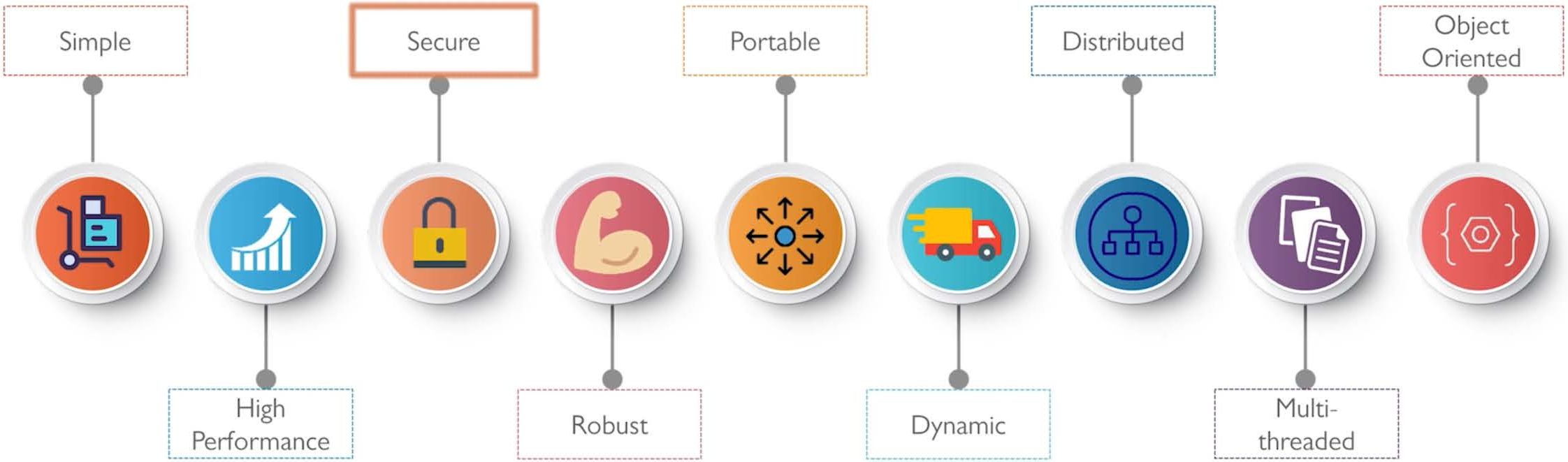
Many objects are evaluated at run time and execution is carried out. For example: Runtime polymorphism

Java Features



Many objects are evaluated at run time and execution is carried out. For example: Runtime polymorphism

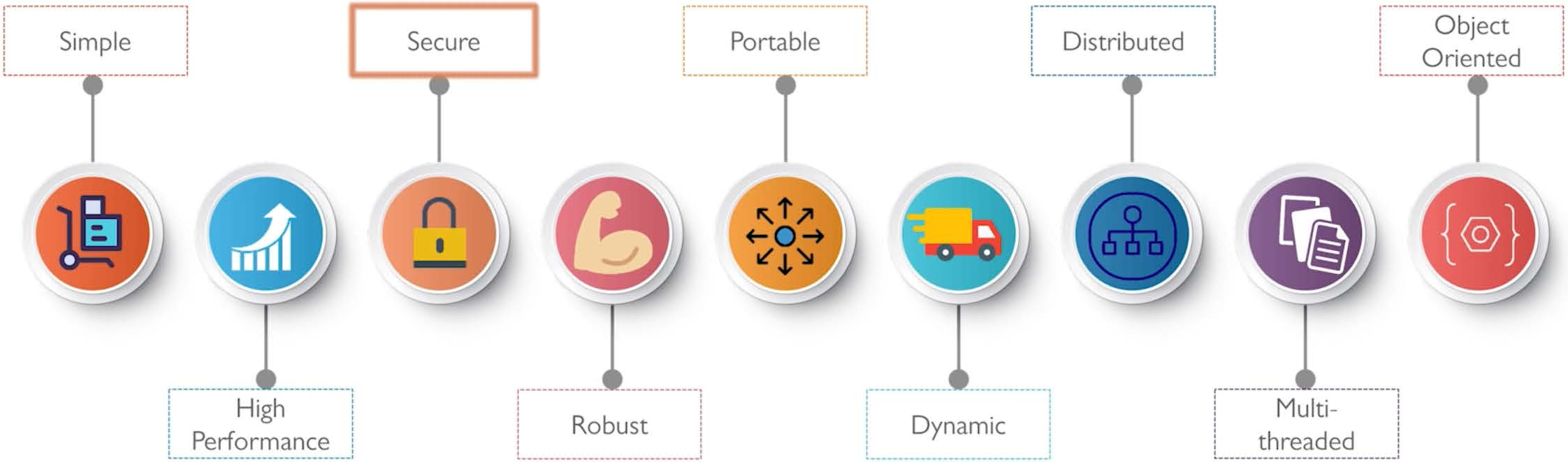
Java Features



RMI (Remote Method Invocation), EJB (Enterprise Java Beans) etc. are used for creating distributed applications using Java

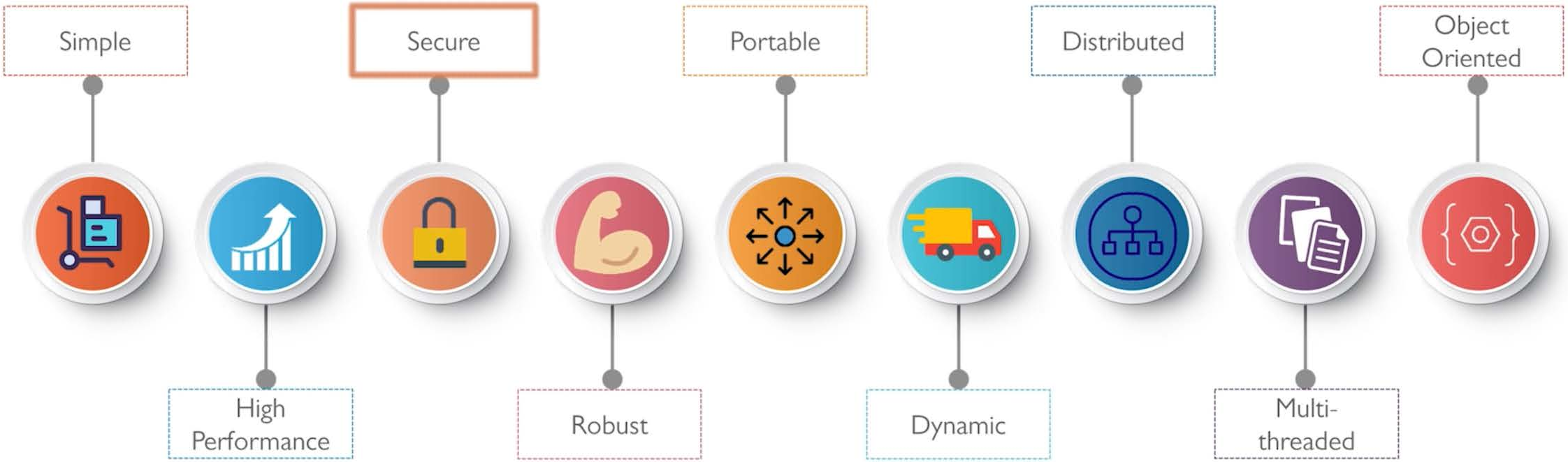
- Using this a program can call a method of another program running in some other computers in the network

Java Features



- Thread is a task in a process/program
- Multi-threading is multiple tasks running/executing at the same time
- This facility is provided by Java so that many tasks can be executed at the same time

Java Features



Java is an object oriented programming language. Everything is performed using “objects”. Java can be easily extended since it is based on the Object model.

Where Is Java Used?

Android Apps

All Android applications are written in Java programming language, with Google's Android API, which is similar to JDK

Server Apps at Financial Services Industry

Lots of global Investment banks like Citigroup, Barclays, Standard Chartered and other banks use Java for writing front and back office electronic trading system, writing settlement and confirmation systems, data processing projects and several others

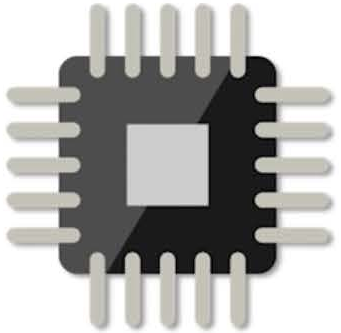
Java Web applications

Many of government, healthcare, insurance, education, defence and several other department have their web application built in Java using Servlets, JSP, Struts, Spring MVC

Where Is Java Used?

Embedded Systems

Java is the platform where you only need 130 kb memory to use Java technology(on a smart card or sensor)



Web Servers and Application Servers

- Apache Tomcat, Simple, Jo!, Rimfaxe Web Server (RWS) and Project Jigsaw are web server space
- WebLogic, WebSphere, and Jboss EAP dominate commercial application server space

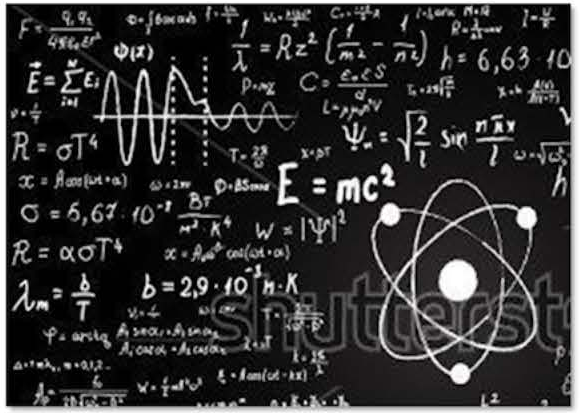
Enterprise Applications

Java Enterprise Edition (Java EE) is a popular platform that provides API and runtime environment for scripting and running enterprise softwares

Where Is Java Used?

Scientific Applications

- Java is the choice of many software developers for writing applications involving scientific calculations and mathematical operations
- These programs are generally considered to be fast and secure, have a higher degree of portability and low maintenance



Big Data technologies

- Hadoop and other big data technologies are also using Java in one way or other
- For Example Apache Kafka components API – Producer and Consumer are written in java

Internet of Things (IOT)

Internet of things which is a way to blend hardware and software together to solve problems faced in real life.

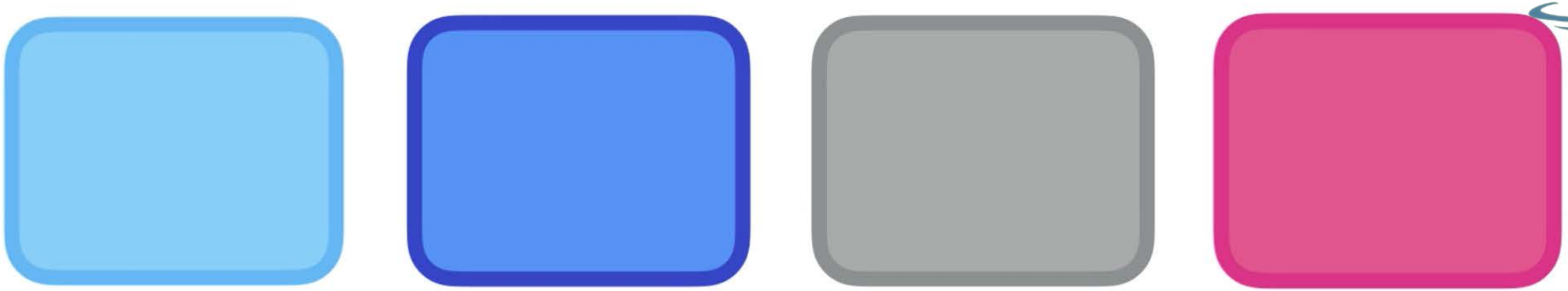
$101,929

indeed.com - May 2019

Fundamentals

- Types
- Control flow
- Clean coding
- Finding and fixing errors
- Packaging your applications

A solid foundation on programming with Java



Fundamentals

Object-oriented Programming

Core Java APIs

Advanced Features

Primitive

for storing
simple values

Reference

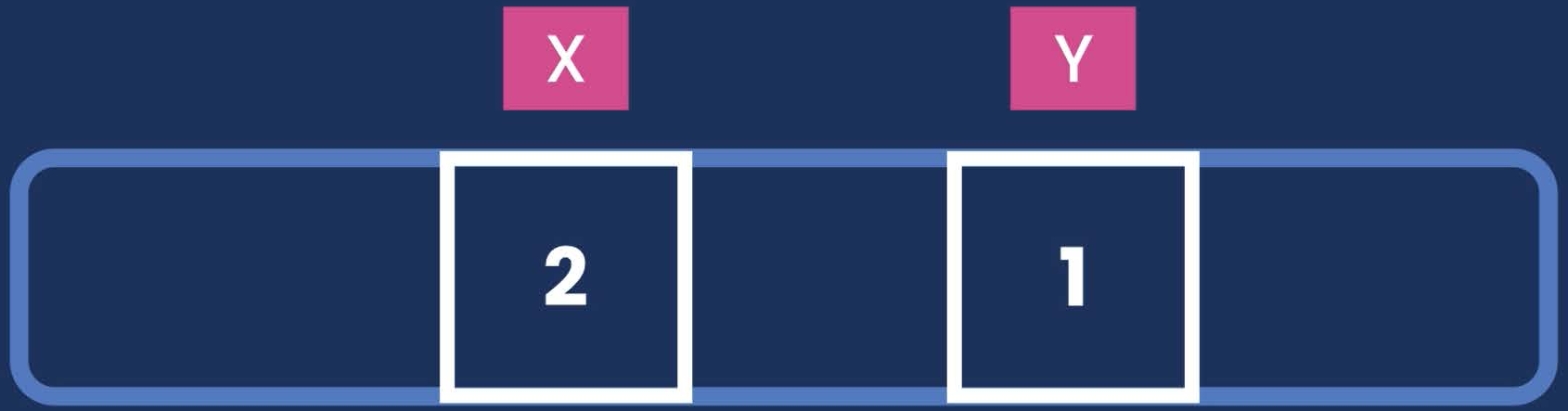
for storing
complex objects

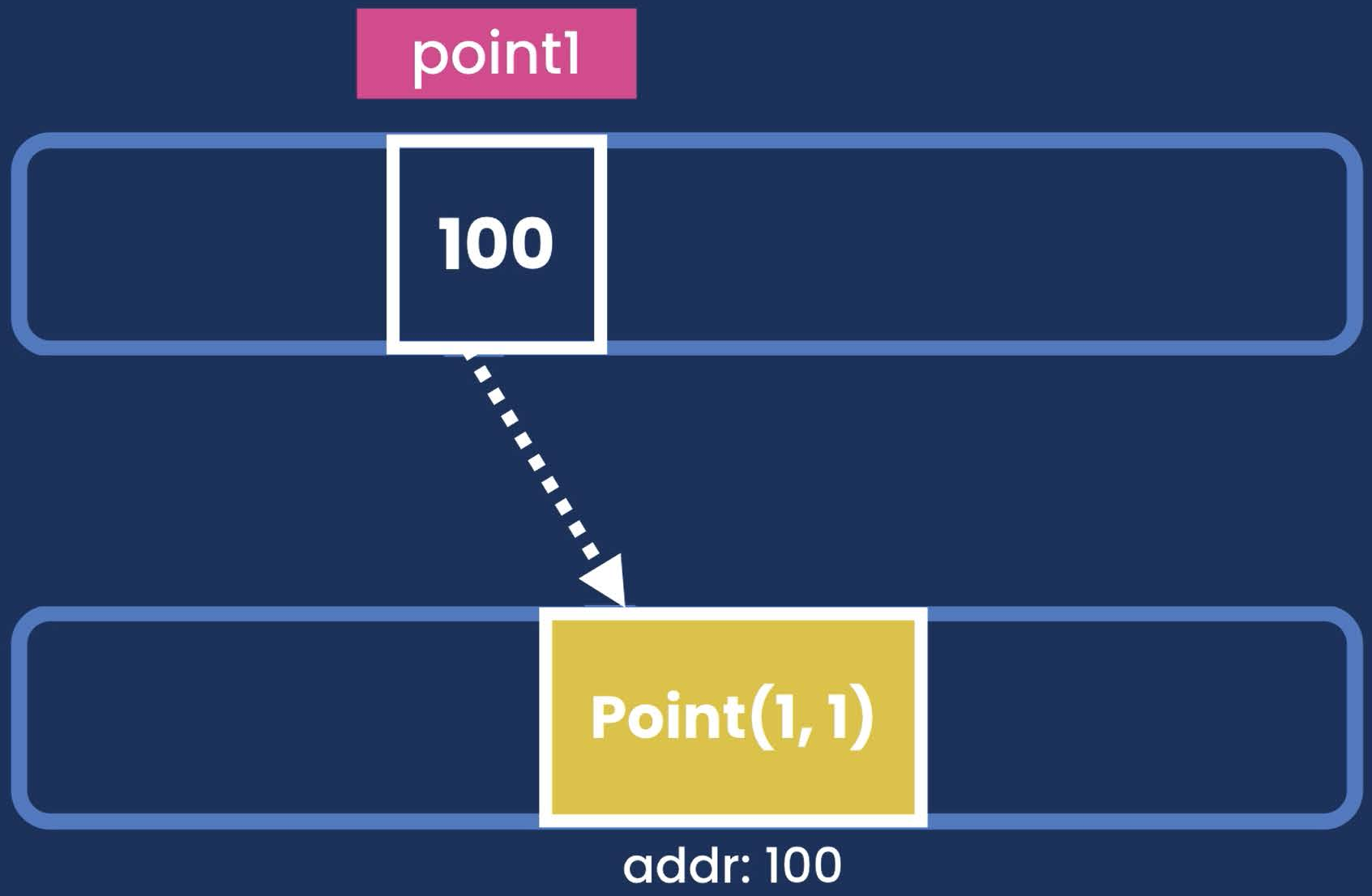
Primitive Types

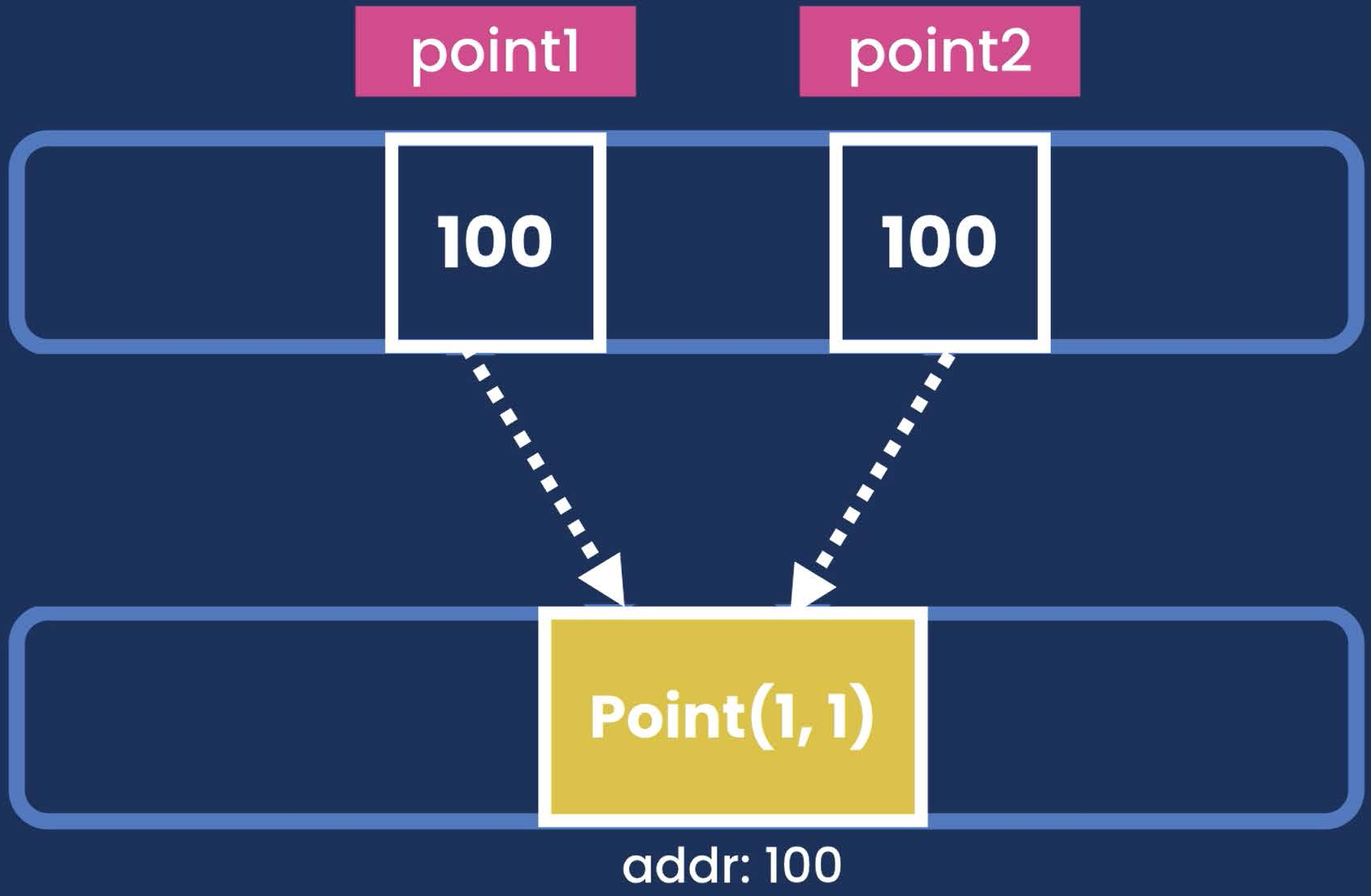
Type	Bytes	Range	
byte	1	[-128, 127]	
short	2	[-32K, 32K]	
int	4	[-2B, 2B]	
long	8		
float	4		
double	8		
char	2	A, B, C, ...	
boolean	1	true / false	

Primitive	Reference
numbers, characters, booleans	date, mail message

Primitive Types vs Reference types


X
Y
2
1


point1
100
Point(1, 1)
addr: 100


point1
point2
100
100
Point(1, 1)
addr: 100

STRINGS

ESCAPE SEQUENCES

ARRAYS

“MULTI-DIMENSIONAL ARRAYS”

CONSTANTS

OPERATORS

ORDER OF OPERATORS

Order of Operators

()

* /

+ -

CASTING

MATH

“FORMATTING NUBERS”

READING INPUT

EXERCISE: MORTGAGE CALCULATOR

https://www.wikihow.com/Calculate-Mortgage-Payments

```
Principal: 100000
Annual Interest Rate: 3.92
Period (Years): 30
Mortgage: $472.81
```

- Variables and Constants
- Primitive and Reference Types
- Casting
- Numbers, Strings and Arrays
- Read Input

Control Flow

Comparison Operators

Logical Operators

Conditional Statements

Loops

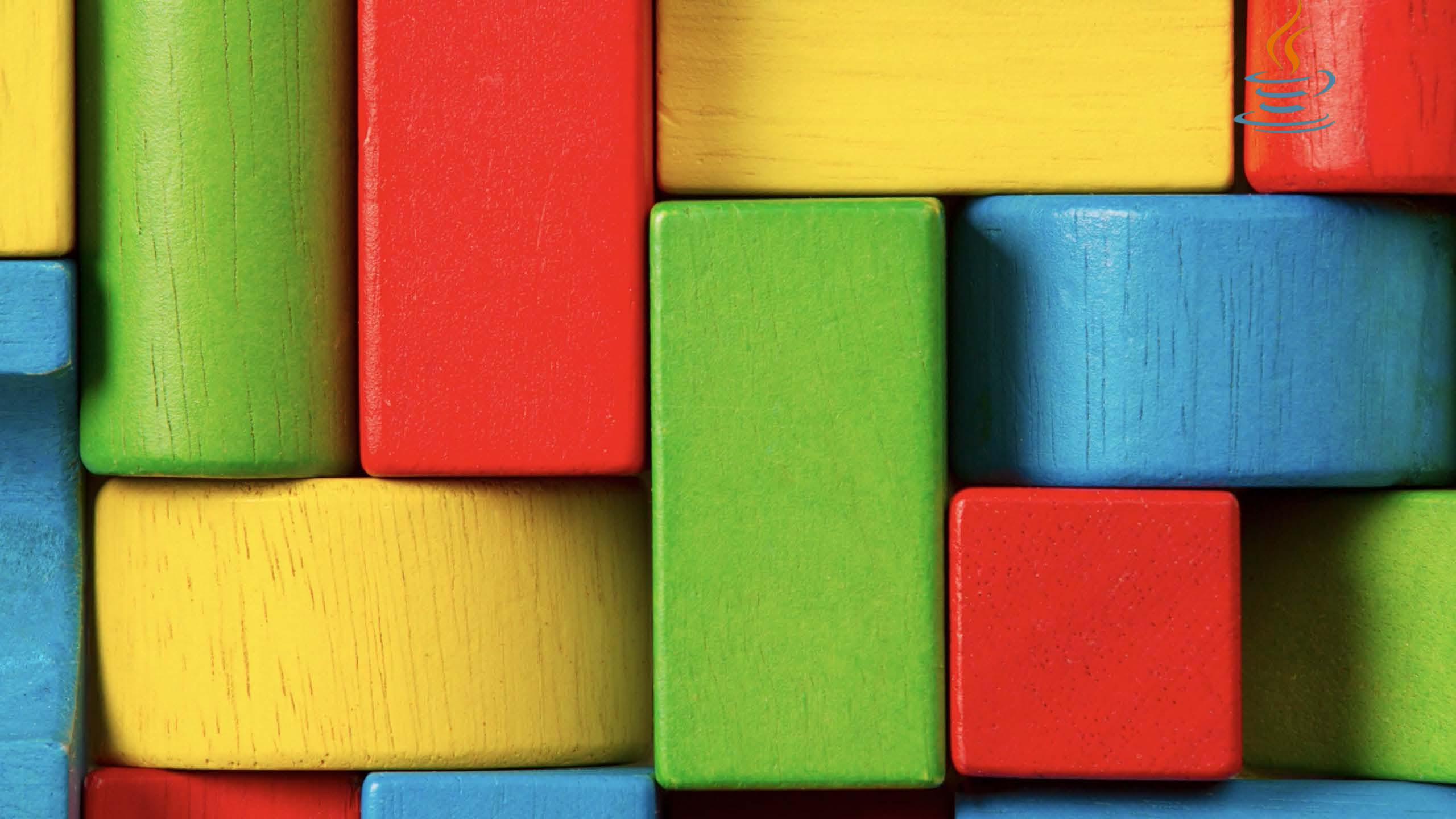
IF ELSE AND TERNARY

LOOPS

Clean Coding

"Any fool can write code that computers can understand.
Good programmers write code that humans can understand."

— Martin Fowler



CREATING METHODS

REFACTOR CALCULATOR

HTTPS://WWW.MTGPROFESSOR.COM//FORMULAS.HTM

- Finding and fixing errors
- Types of errors
- Common errors
- Debugging
- Packaging

Compile-time Errors

Run-time Errors

Object-oriented Programming

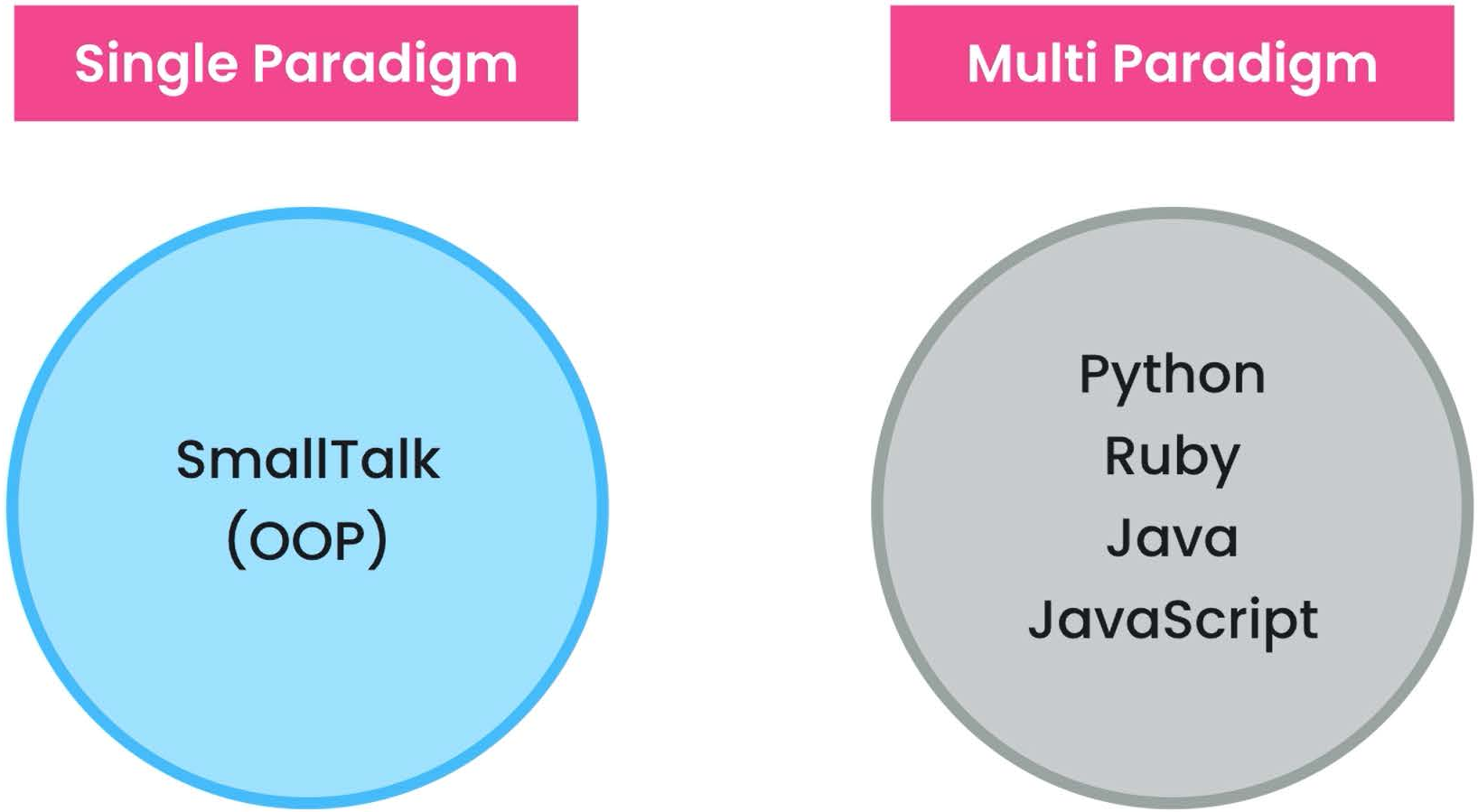
WHAT YOU MUST KNOW

- Variables
- Data types
- Methods
- Control flow

“PROGRAMMING PARADIGMS”

PROGRAMMING PARADIGMS

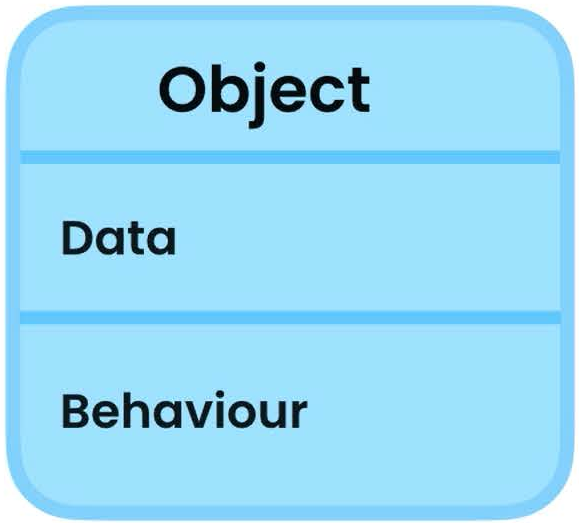
- Procedural
- Functional
- Object-oriented
- Event-driven
- Logic
- Aspect-oriented


Single Paradigm
SmallTalk
(OOP)
Multi Paradigm
Python
Ruby
Java
JavaScript

Data (State)

Methods (Behaviour)

Object-oriented



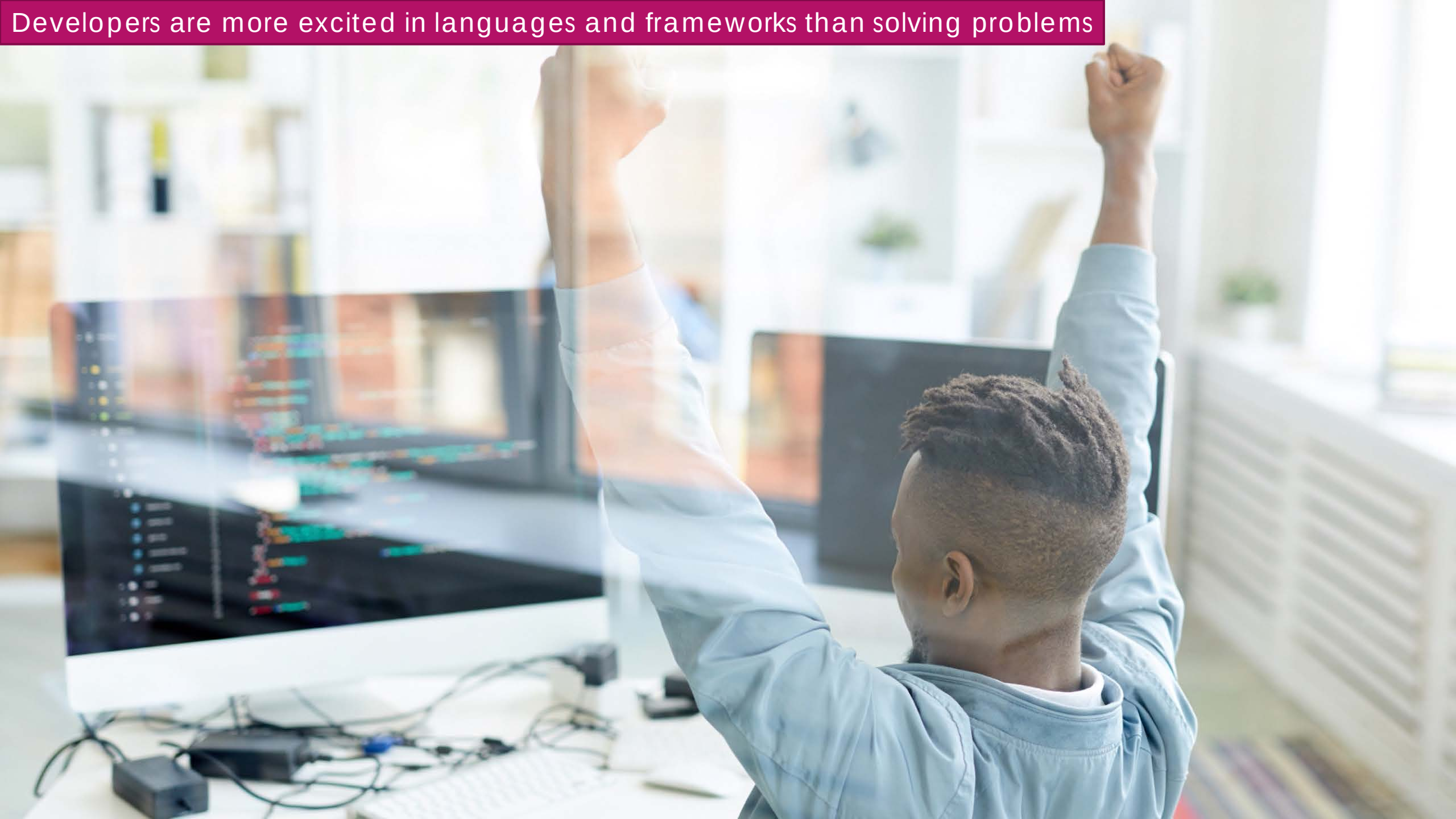
Functional


Difficult
?'s


DETERMINISM
LIGHTYEAR
DETERMINISM EVERYWHERE
imgflip.com

Object-oriented or Functional?

Developers are more excited in languages and frameworks than solving problems



Problem Solving

Process of **defining a problem**, identifying and comparing **different solutions**, and picking the one that best solves that problem with respect to the **context** and **constraints**.

PROGRAMMING PARADIGMS

Event-driven

Functional

Object-oriented

Depends on the

problem, **context** and **budget**

BENEFITS

- Reduced Complexity
- Easier Maintenance
- Code Reuse
- Faster Development

Most people thinks that in real world no application is with object oriented program, and no object replaced with another

Both object-oriented and functional programming are great

Classes

Refactoring Towards Object-oriented Code

Inheritance & Polymorphism

Interfaces

Encapsulation

Abstraction

Classes

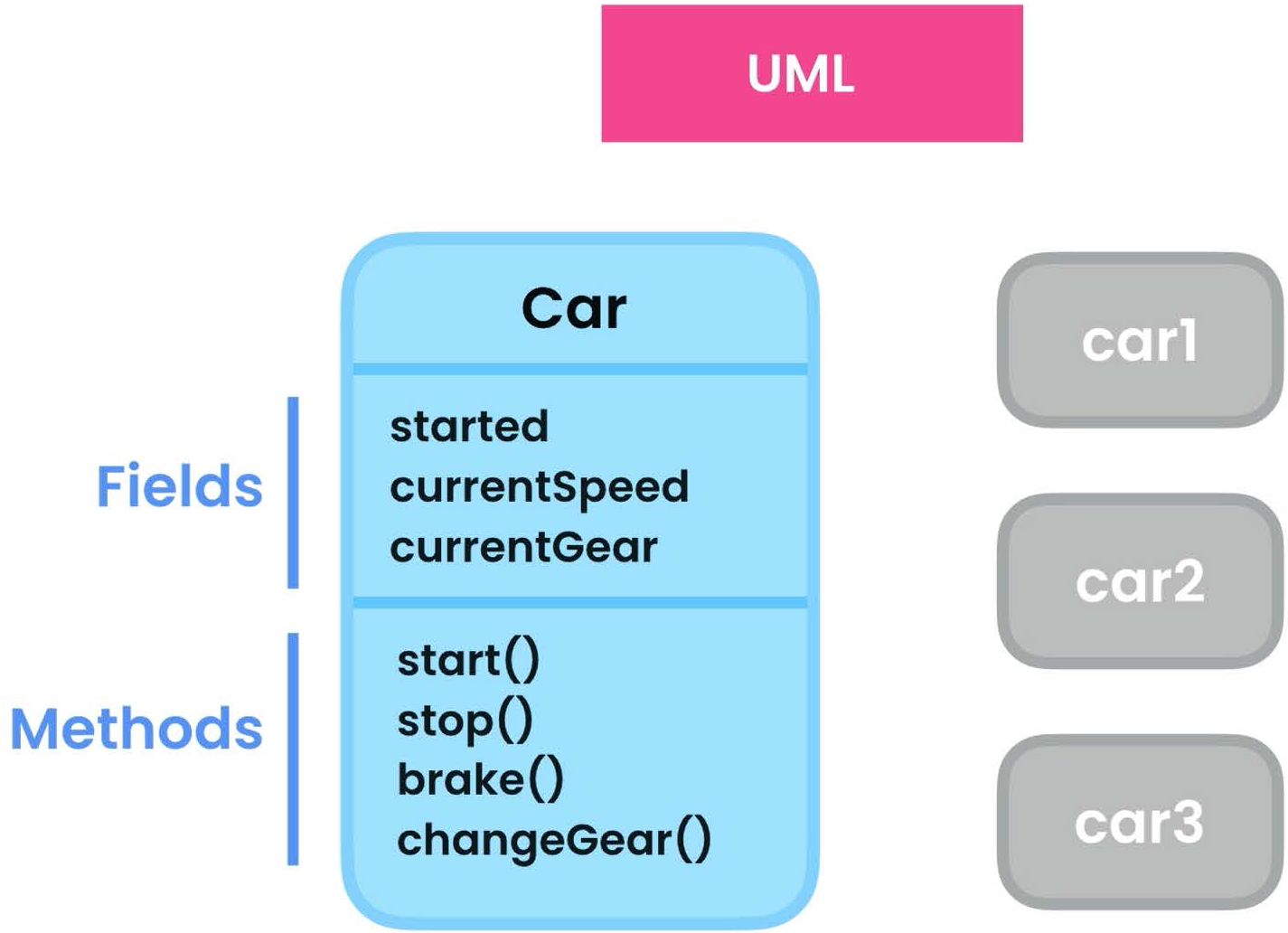
- Encapsulation
- Abstraction
- Constructors
- Getters / Setters
- Method Overloading

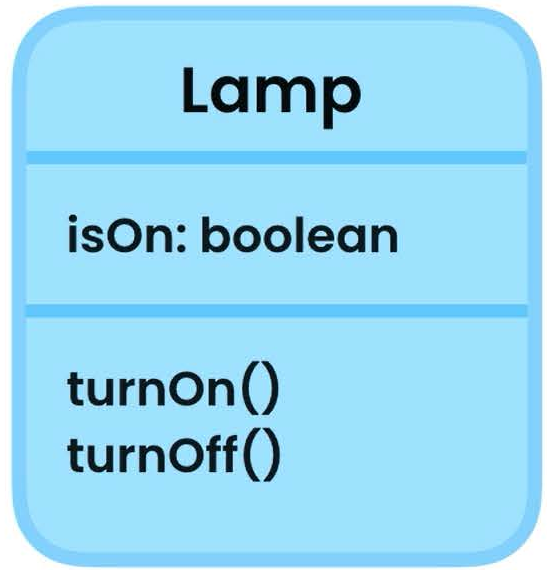
Class

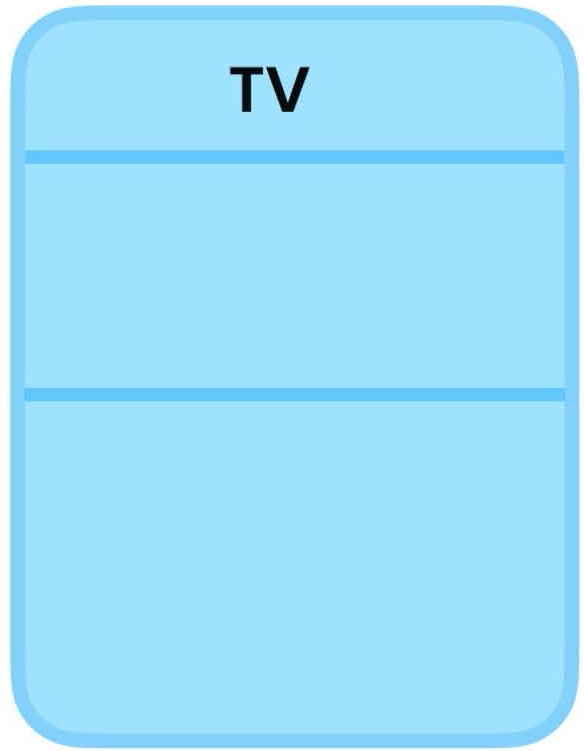
A blueprint for creating objects

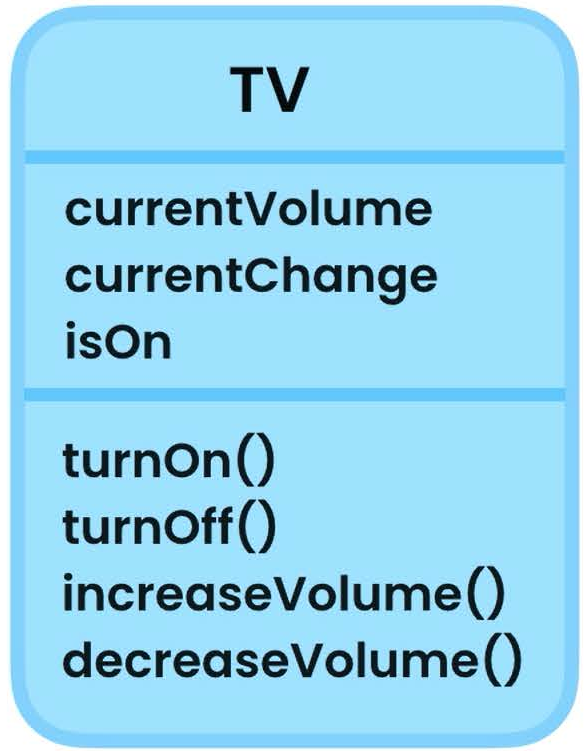
Object

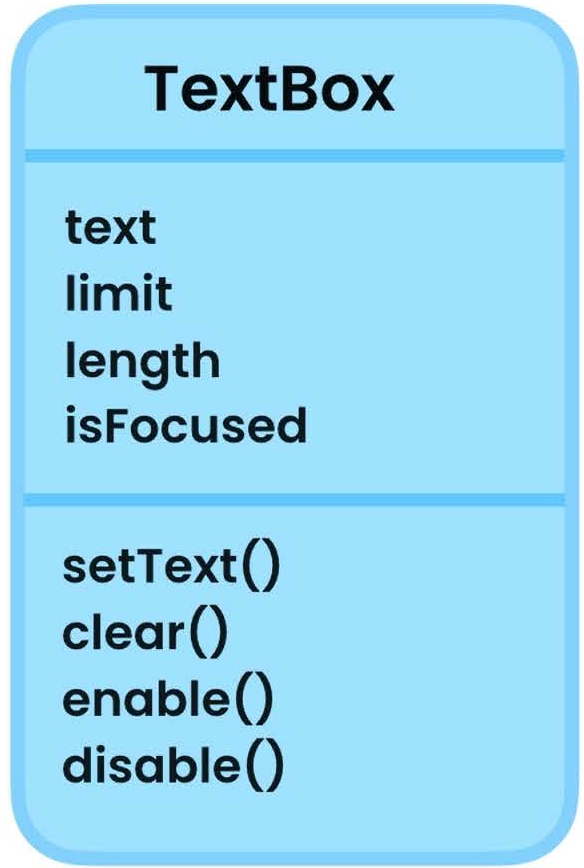
An instance of a class


UML
Car
started
currentSpeed
currentGear
start()
stop()
brake()
changeGear()
Fields
Methods
car1
car2
car3


Lamp
isOn: boolean
turnOn()
turnOff()


TV

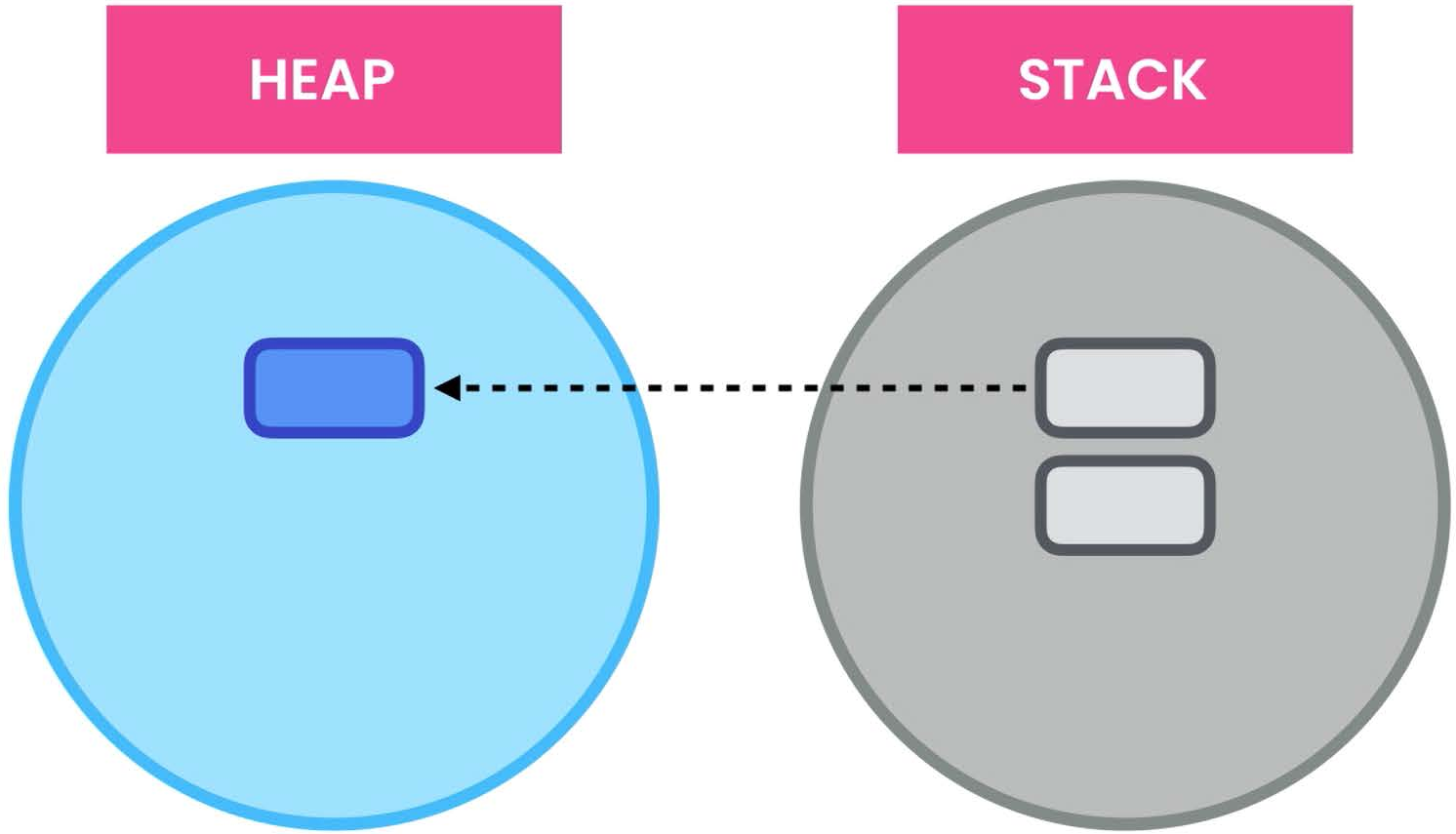

TV
currentVolume
currentChange
isOn
turnOn()
turnOff()
increaseVolume()
decreaseVolume()


TextBox
text
limit
length
isFocused
setText()
clear()
enable()
disable()

“

MEMORY

”



Objects

Primitives and
short-lived variables

“Memory Deallocation”

In JAVA & C
We don’t need to worry about “Memory Deallocation” it will automatically delete from stack once the execution is over.

“Garbage Collector”

Once the process is over, there is a software component called “Garbage Collector” in JAVA (JRE), it will automatically detect un-used objects from heap and it will remove it.

“PROCEDURAL PROGRAMMING (PATTERN)”

“ENCAPSULATION”

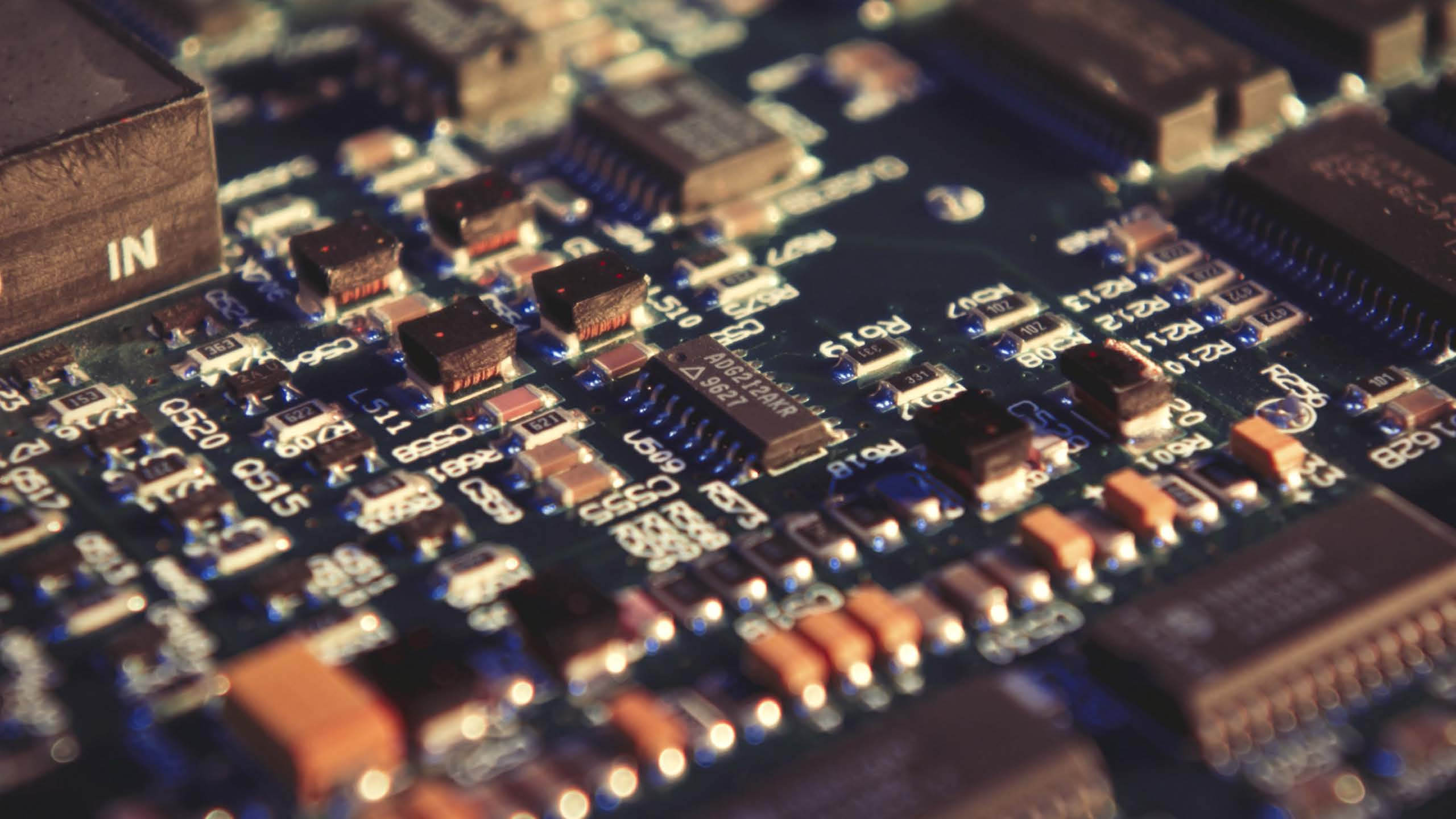
Bundle the data and methods that operate on the data in a single unit.

GETTERS AND SETTERS

“ABSTRACTION”

Reduce complexity by hiding unnecessary details


AUDIO
SYNC MENU
RELATED SEARCH
TV PAUSE
TITLE LIST
REC
SOCIAL VIEW
FOOTBALL
APPS
SEN
DISCOVER
GUIDE
RETURN
1
4
7


IN
ADG212AKR
9627

Abstraction

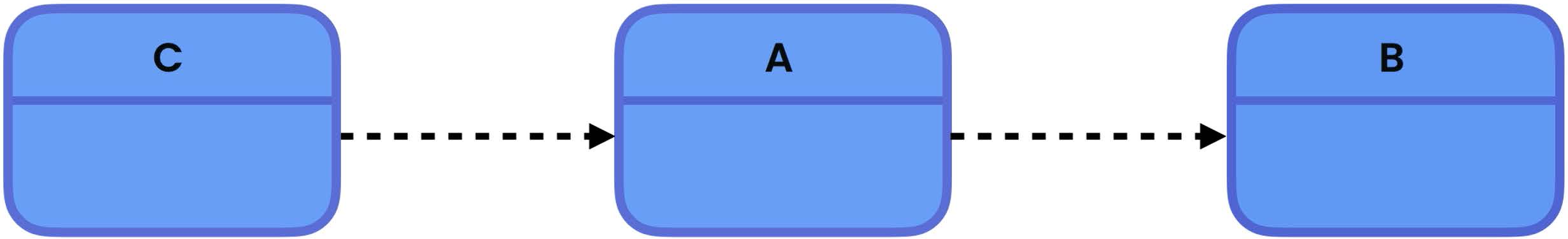

MENU
TEXT
APP
DISP
MEDIA
EXIT
INFO
CH
2 ABC
3 DEF
6 MNO

“COUPLING”

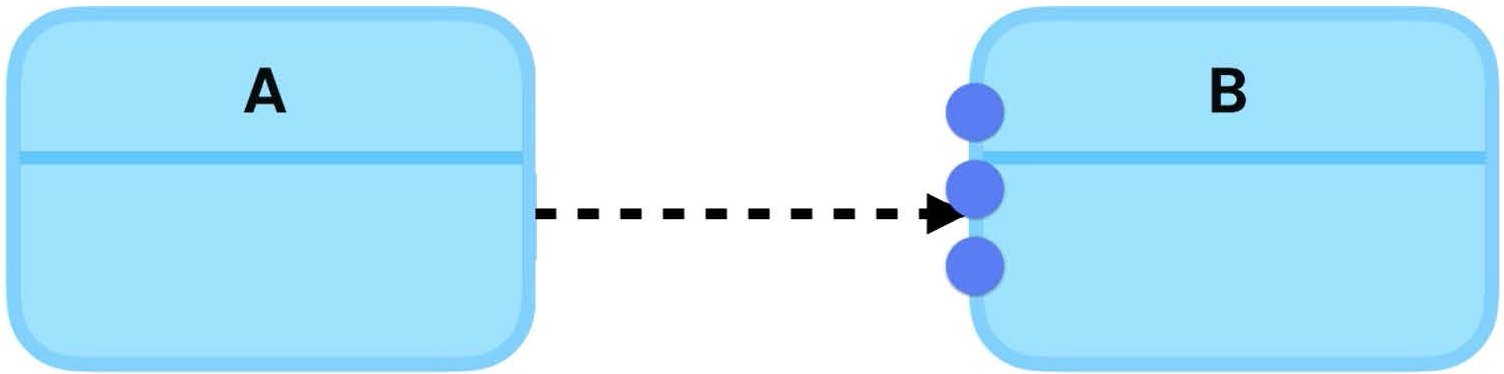

Coupling? Huh?

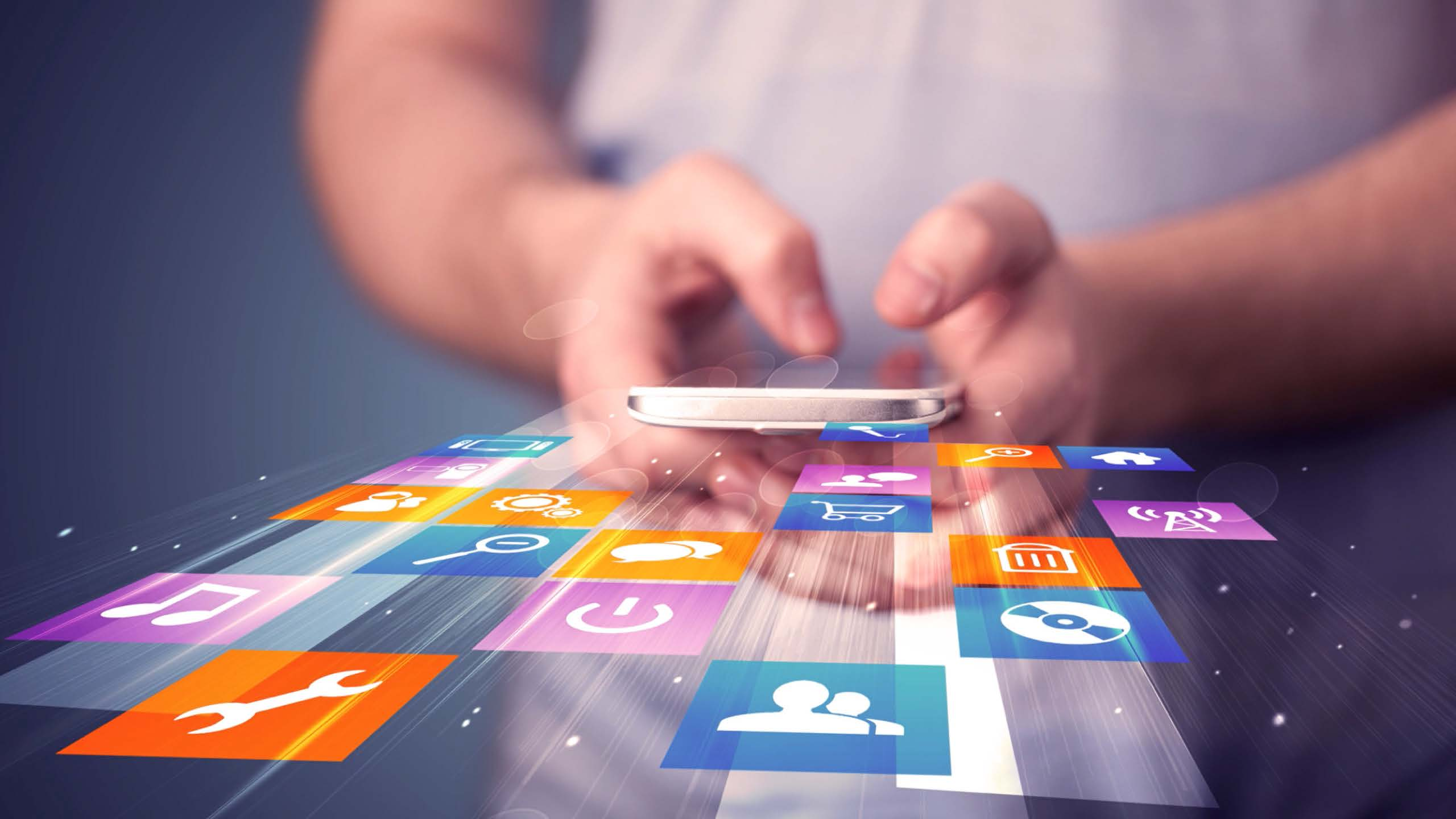
The level of dependency between classes

Coupling



Coupling



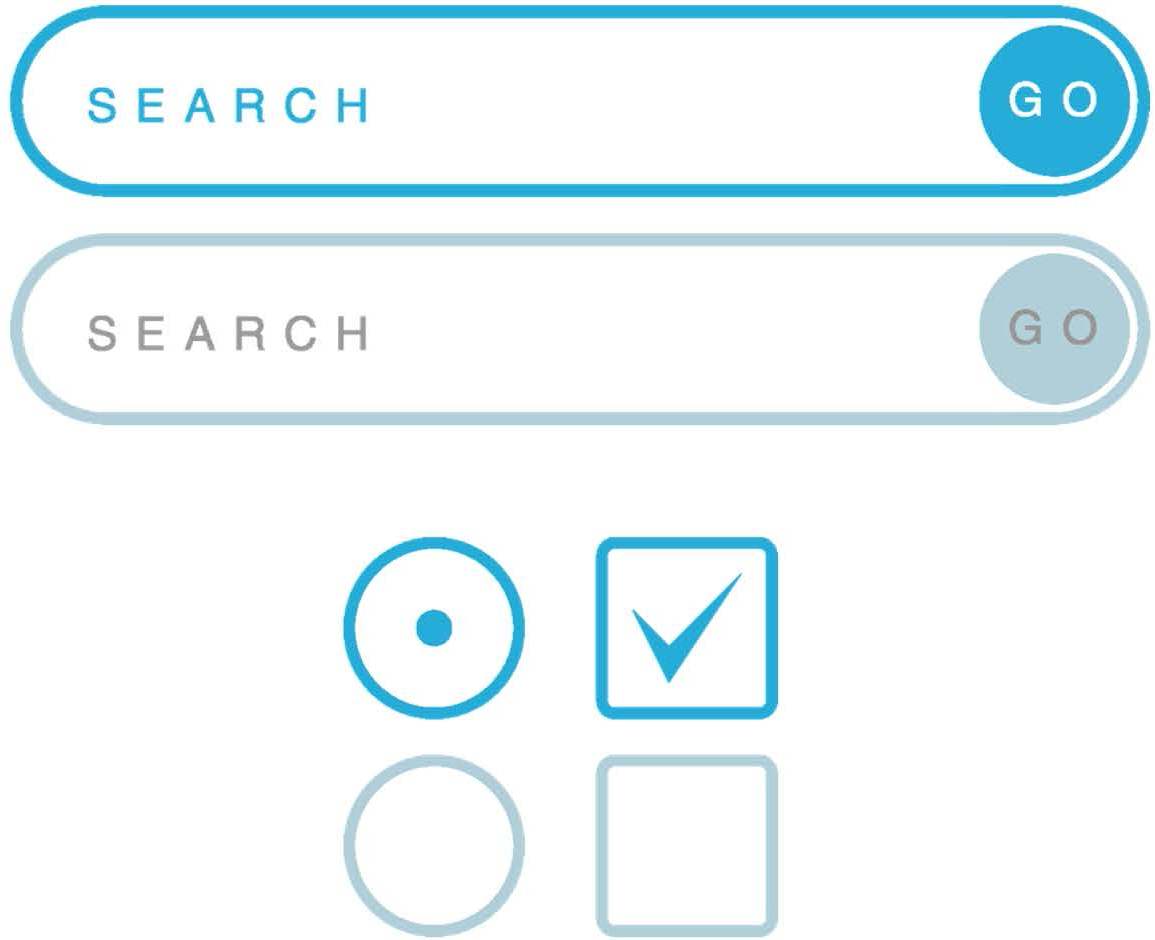


REDUCING COUPLING

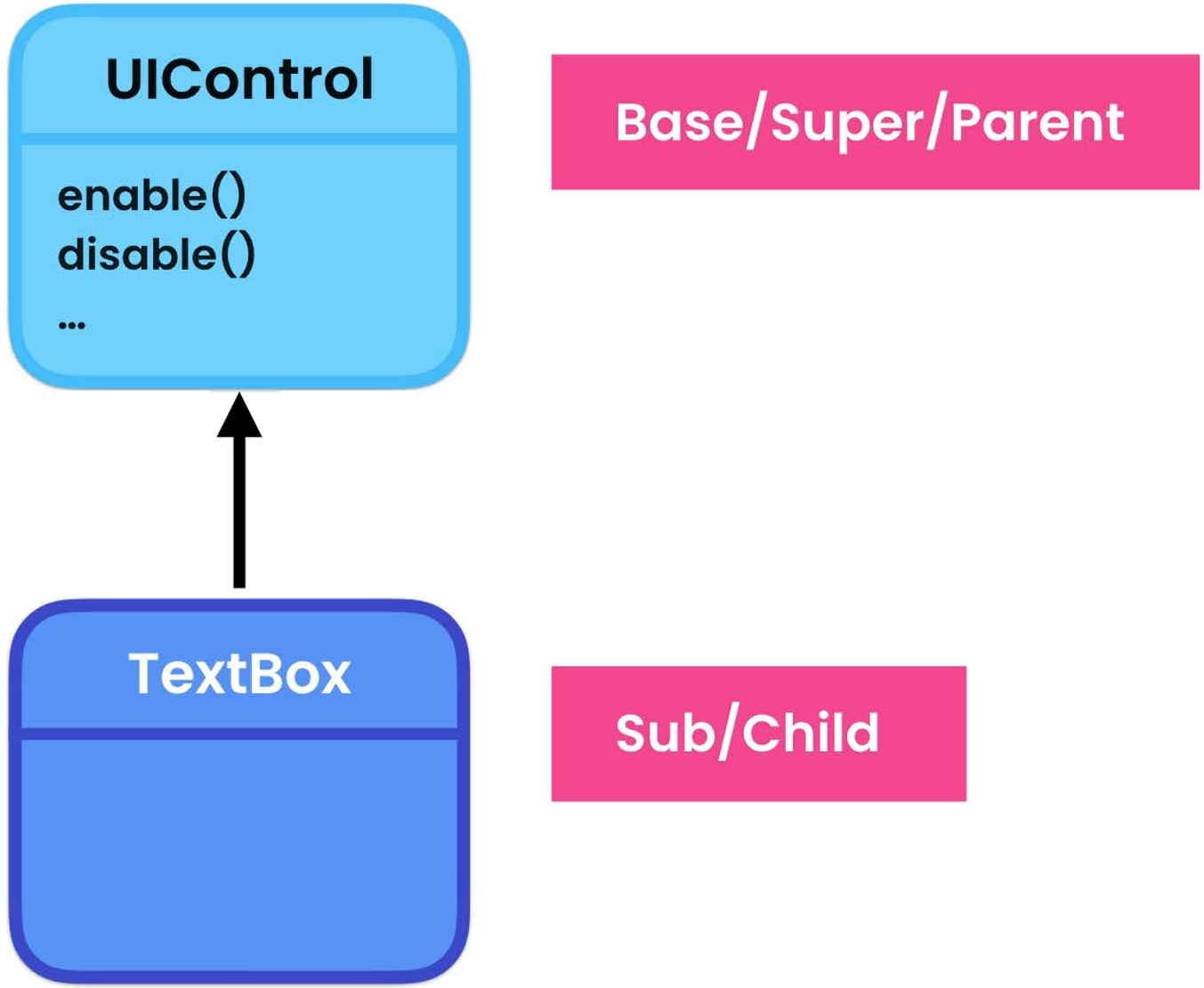
	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Inheritance

- Constructors
- Access Modifiers
- Overriding Methods
- Comparing Objects
- Polymorphism


SEARCH
GO
SEARCH
GO

Inheritance


UIControl
enable()
disable()
...
TextBox
Base/Super/Parent
Sub/Child

Upcasting

Casting an object to one of its **super** types

Downcasting

Casting an object to one of its **sub** types

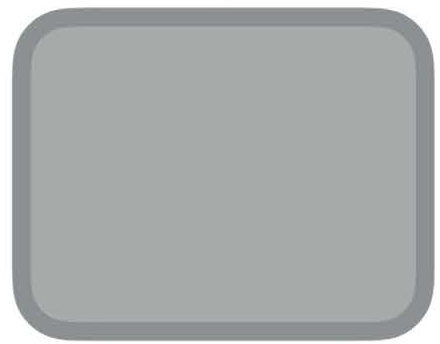
Object-oriented Programming

Encapsulation Abstraction Inheritance

Object-oriented Programming

Encapsulation

Abstraction

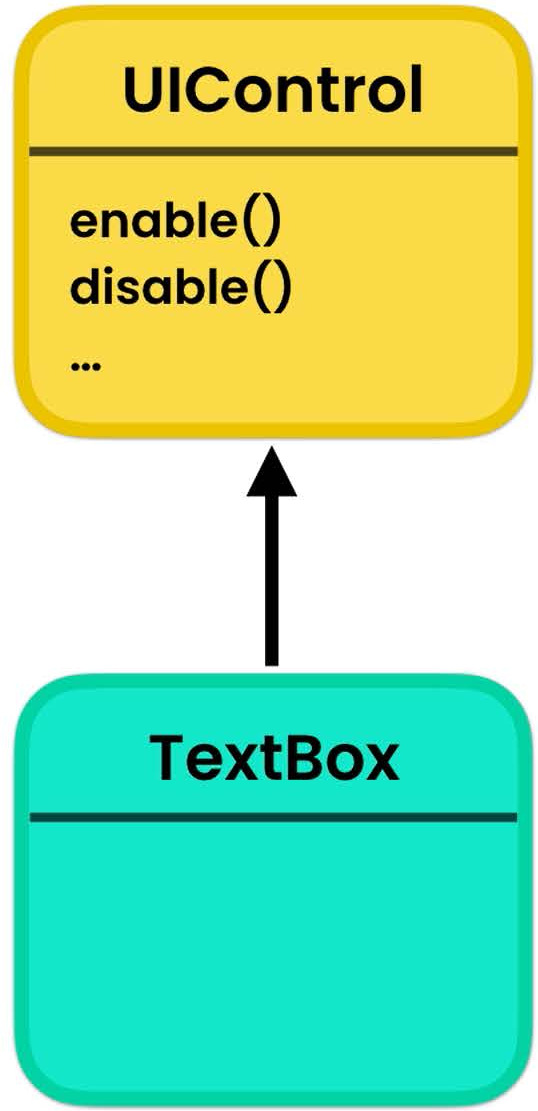


Inheritance

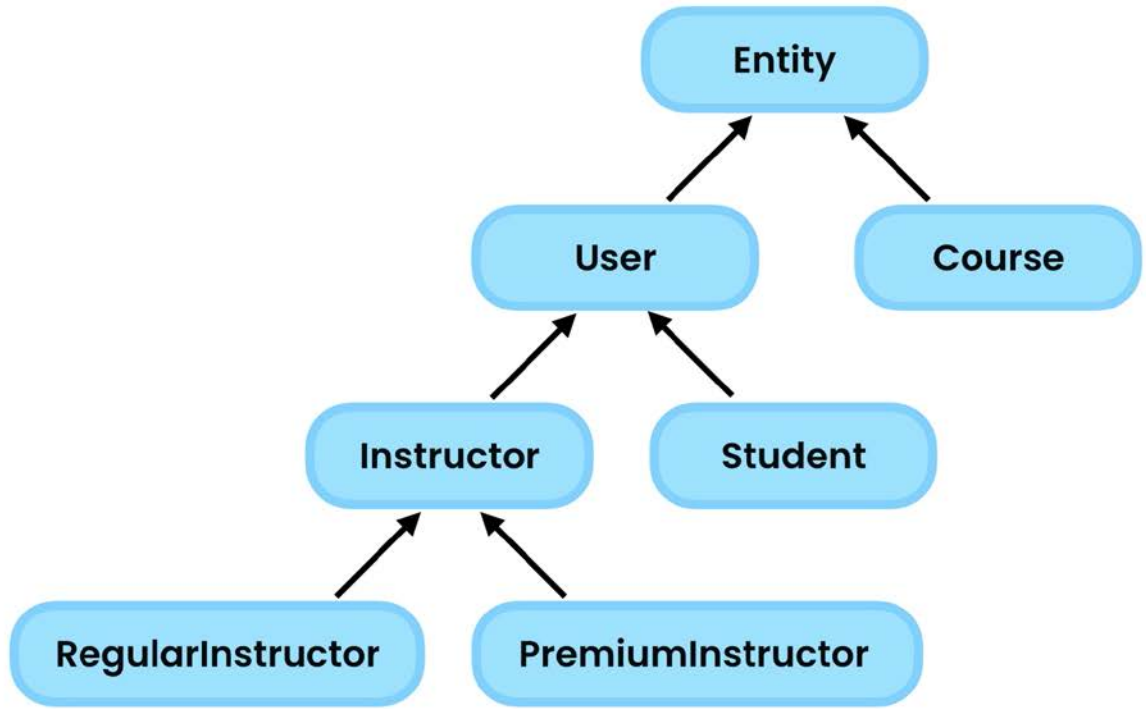
Polymorphism

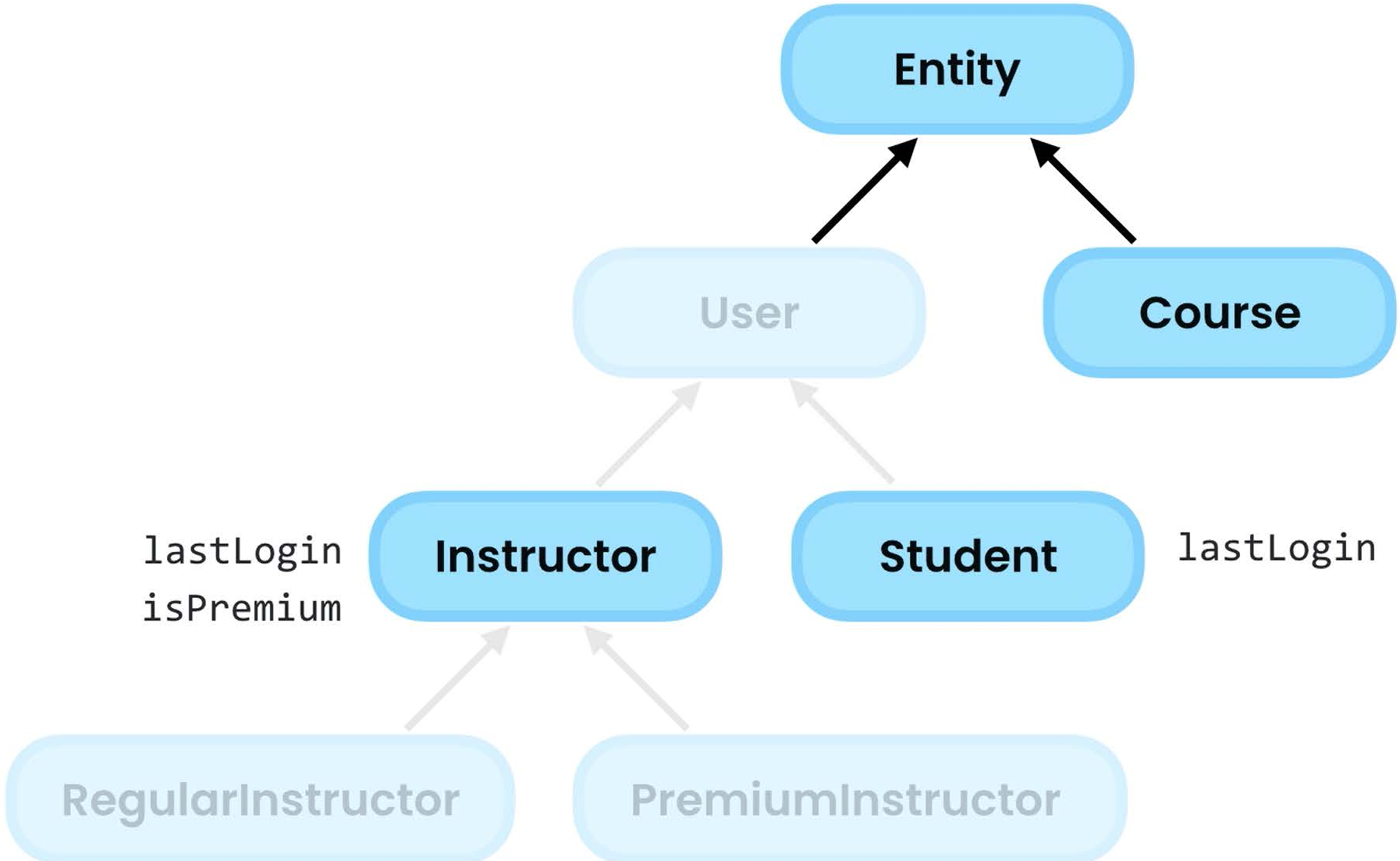
Poly-morph-ism

Many Form

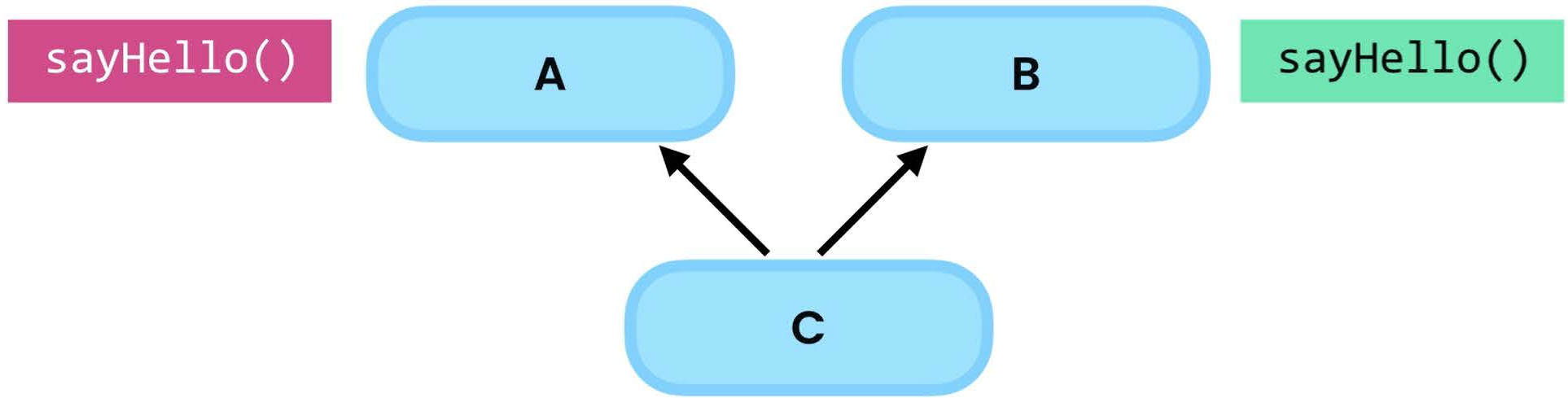

UIControl
enable()
disable()
...
TextBox

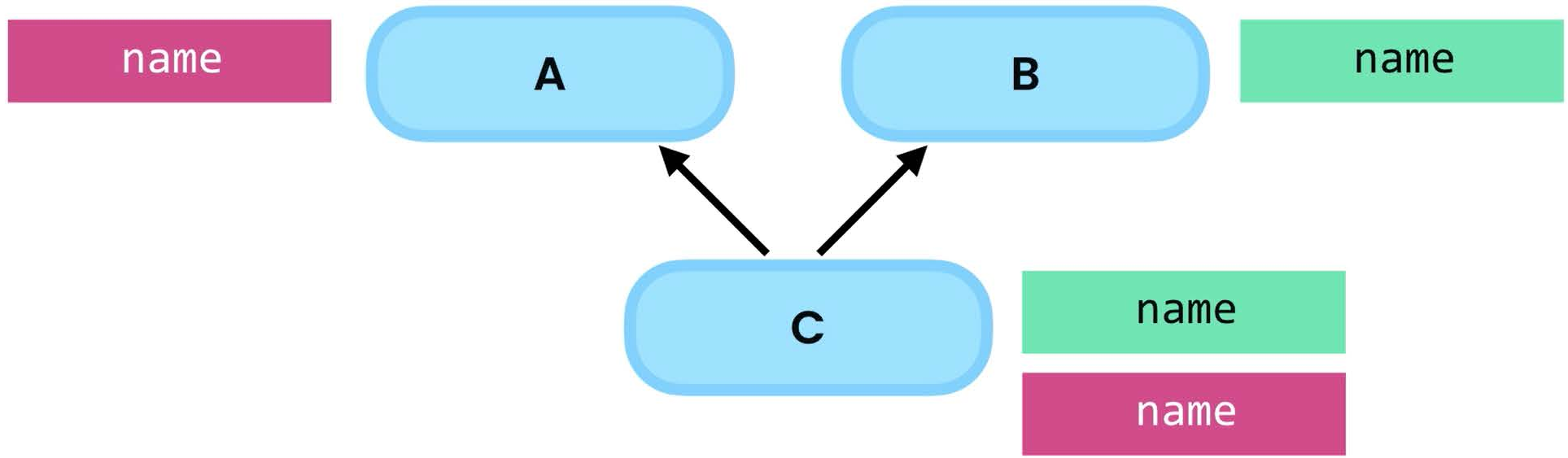
Don’t create deep inheritance hierarchies

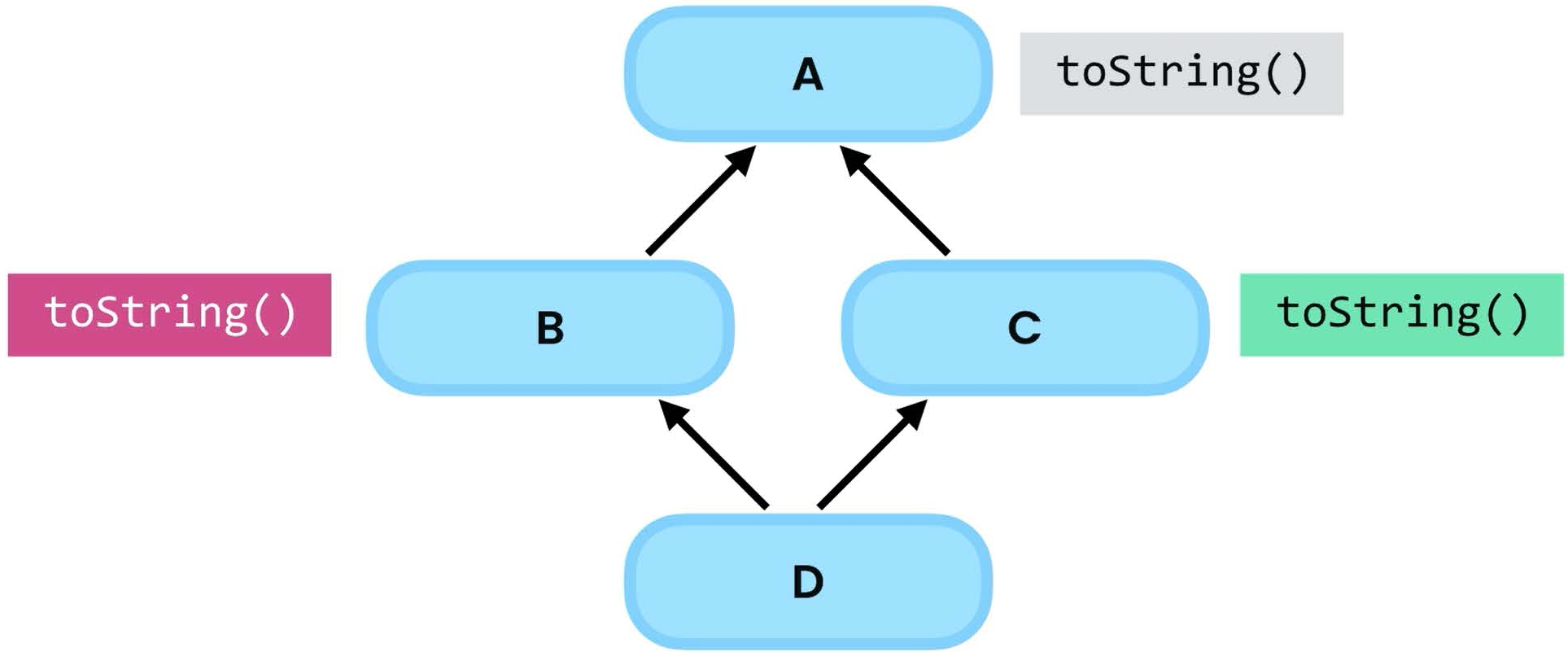

Entity
User
Course
Instructor
Student
RegularInstructor
PremiumInstructor

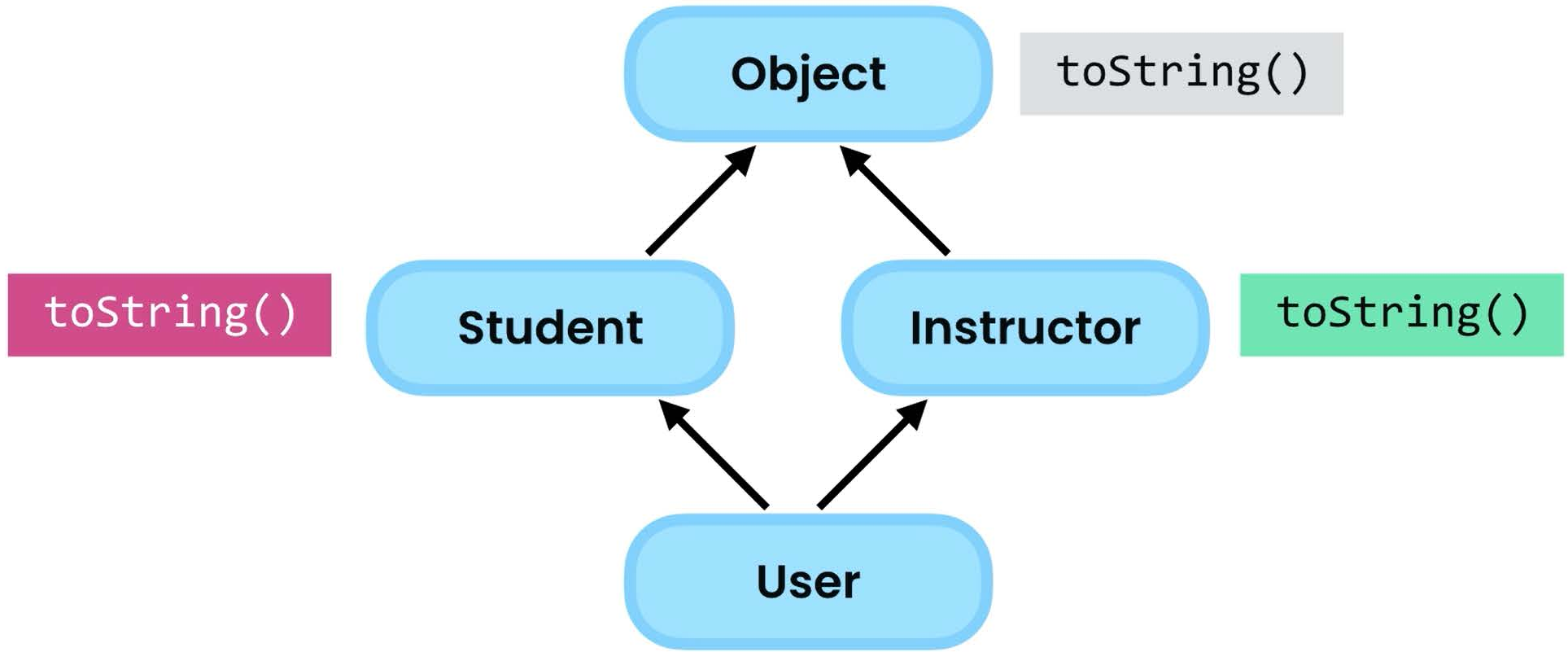

Entity
User
Course
lastLogin
isPremium
Instructor
Student
lastLogin
RegularInstructor
PremiumInstructor

MULTIPLE INHERITANCE


sayHello()
A
B
sayHello()
C


name
A
B
name
C
name
name


A
toString()
toString()
B
C
toString()
D


Object
toString()
toString()
Student
Instructor
toString()
User

YAGNI

You **a**ren't **g**onna **n**eed **i**t

Inheritance & Polymorphism

“INTERFACES”

Interfaces

What interfaces are

Why we need them

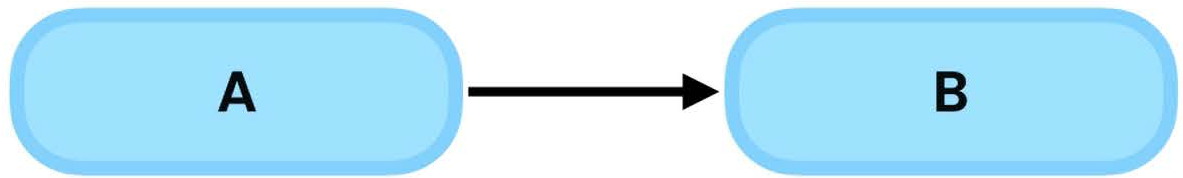
How to use them “properly”

Dependency injection

We use interface to build loosely-coupled, extensible, testable applications.

Abstraction

Hide the implementation details


A
B


A
B

```
public interface Draggable {
      void drag();
}
```


A
Interface
B

Interface

What should be done

- Data compression
- Encryption
- Sorting
- Searching

Classes

How it should be done

We use interface to build loosely-coupled, extensible, testable applications.

“DEPENDENCY INJECTION”

Our Classes should not instantiate their dependencies.

DEPENDENCY INJECTION

- Constructor Injection
- Setter Injection
- Method Injection

INTERFACE SEGREGATION PRINCIPLE

Divide big interfaces into smaller ones

Interfaces

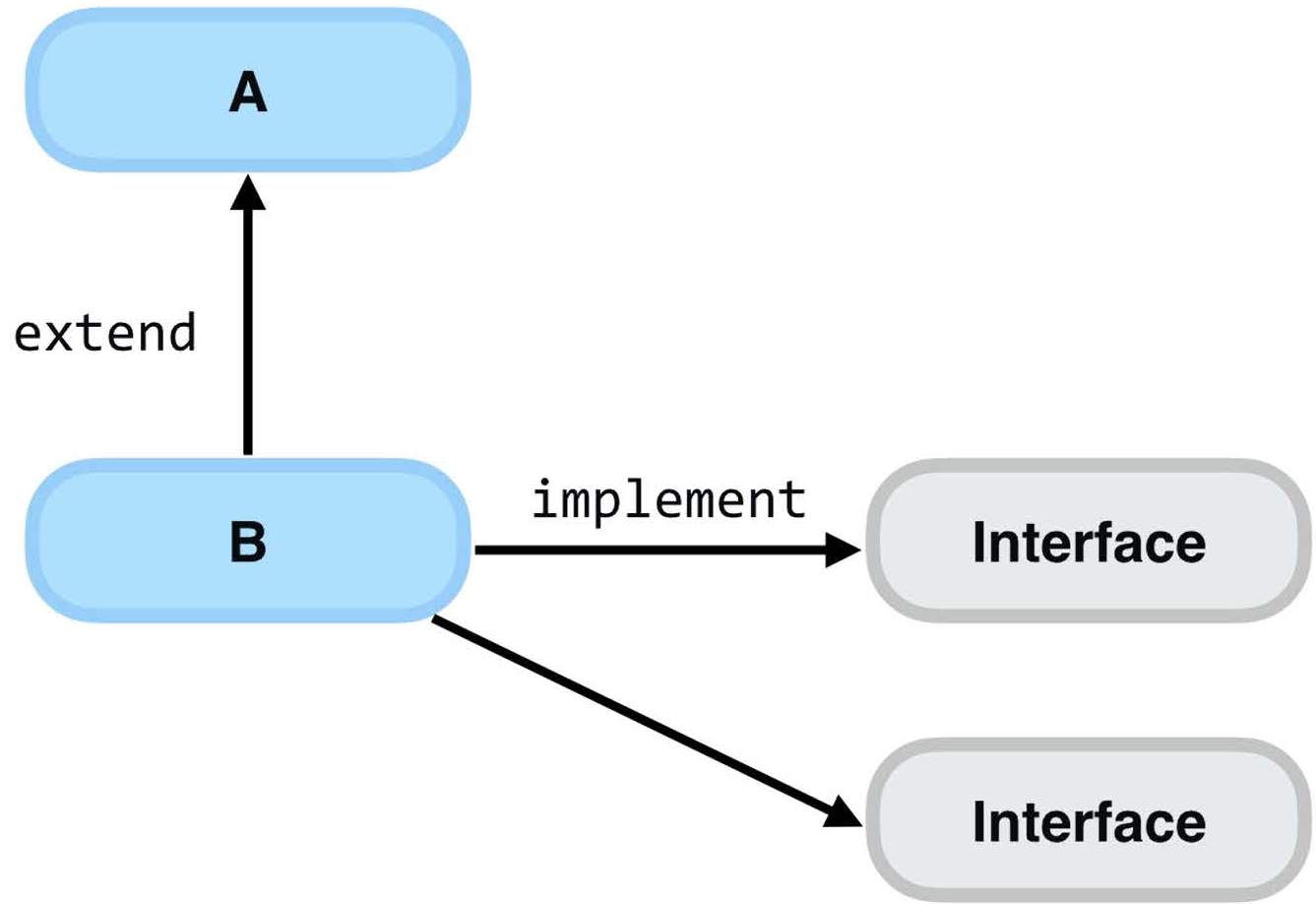
(Contracts)

To build loosely-coupled, extensible, testable applications

Abstract Classes

(Partially-completed Classes)

To share code


A
extend
B
implement
Interface
Interface

Benefits of Interfaces

Swap Implementations

Extend Your Applications

Test Your Classes in Isolation

WHAT YOU'LL LEARN

- Exceptions
- Generics
- Collections
- Lambda Expressions & Functional Interfaces
- Streams
- Multi-threading
- Asynchronous Programming

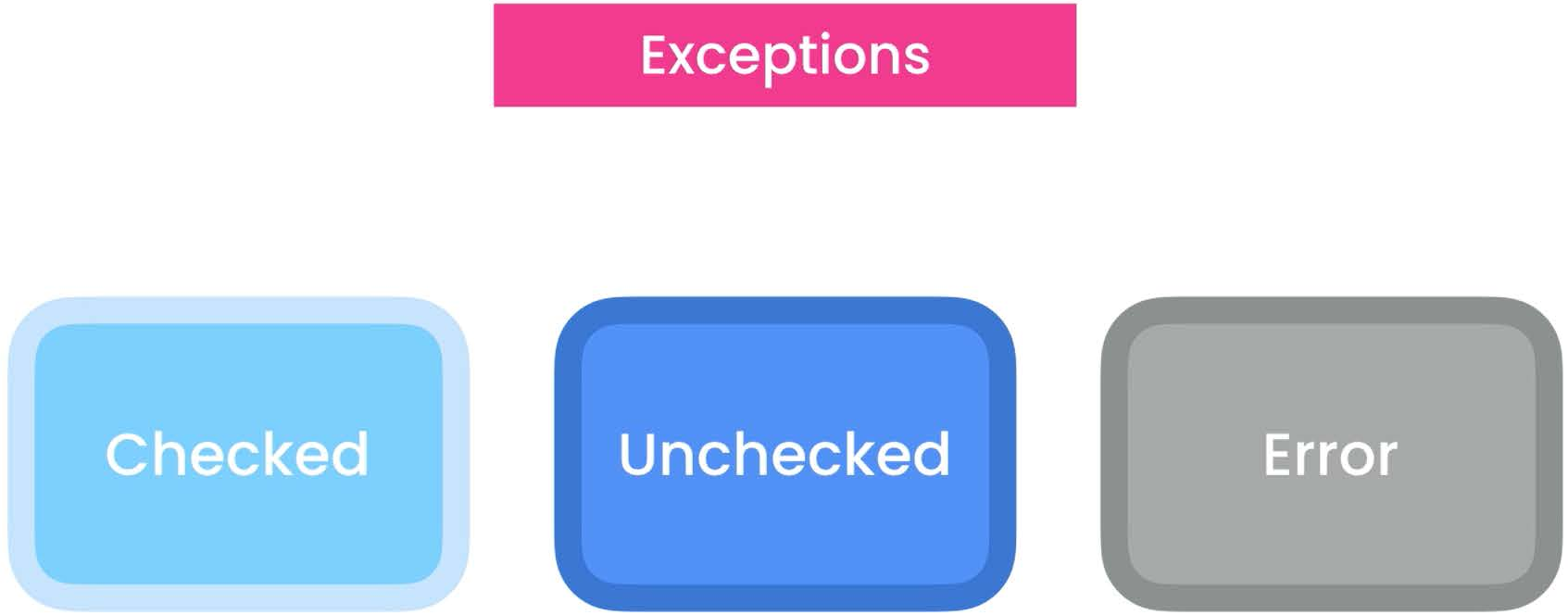
WHAT YOU SHOULD KNOW

- Classes
- Interfaces
- Object-oriented Programming
- Inheritance
- Polymorphism

Exceptions

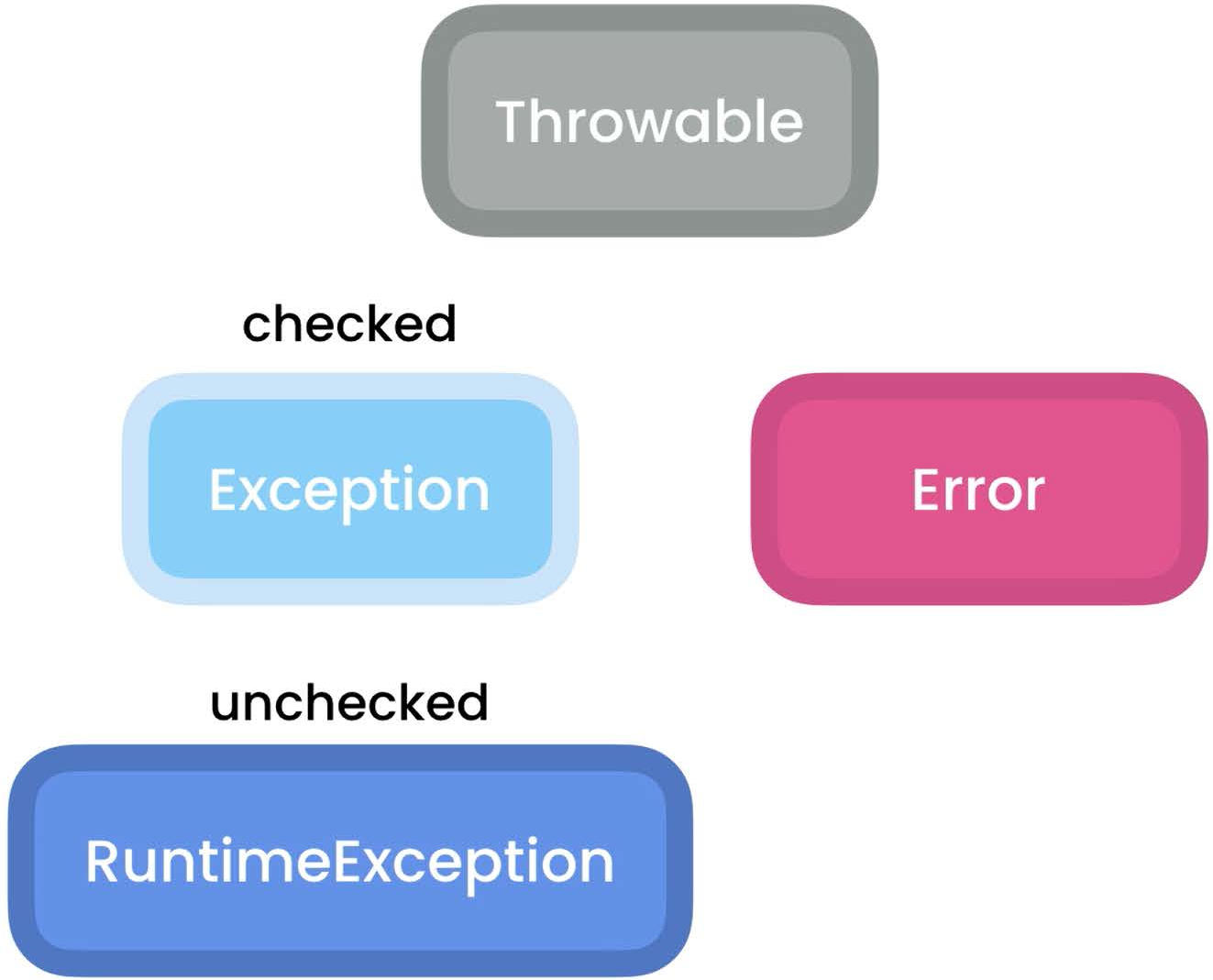
IN THIS SECTION

- Exceptions
- Types of Exceptions
- Handling Exceptions
- Custom Exceptions
- Chaining Exceptions


Exceptions
Checked
Unchecked
Error

Runtime Exceptions

- NullPointerException
- ArithmeticException
- IllegalArgumentException
- IndexOutOfBoundsException
- IllegalStateException


Throwable
checked
Exception
Error
unchecked
RuntimeException

SUMMARY

- Exceptions (checked, runtime, error)
- Try/catch blocks
- The throw keyword
- Custom exceptions
- Chaining exceptions

Generics

Streams

STREAMS

- Overview of Streams
- Mapping
- Filtering
- Slicing
- Sorting
- Reducing
- Collectors

PARADIGMS

- Declarative
- Functional
- Object-oriented
- Event-driven
- ...

Imperative

How

Declarative

What

```sql
SELECT *
FROM movies
WHERE genre = 1
ORDER BY name
```

Streams

To process a collection of data in a declarative way


Collection
Stream

CREATING STREAMS

- From collections
- From arrays
- From an arbitrary number of objects
- Infinite/finite streams

MAPPING

- map()
- flatMap()

Intermediate

- map()
- filter()

Terminal

- forEach()

SLICING STREAMS

- limit(n)
- skip(n)
- takeWhile(predicate)
- dropWhile(predicate)

INTERMEDIATE

- map() / flatMap()
- filter()
- limit() / skip()
- sorted()
- distinct()
- peek()

REDUCERS

- count()
- anyMatch(predicate)
- allMatch(predicate)
- noneMatch(predicate)
- findFirst()
- findAny()
- max(comparator)
- min(comparator)

PRIMITIVE TYPE STREAMS

- IntStream
- LongStream
- DoubleStream

SUMMARY

- Streams
- Mapping
- Filtering
- Slicing
- Sorting
- Reducing
- Collectors