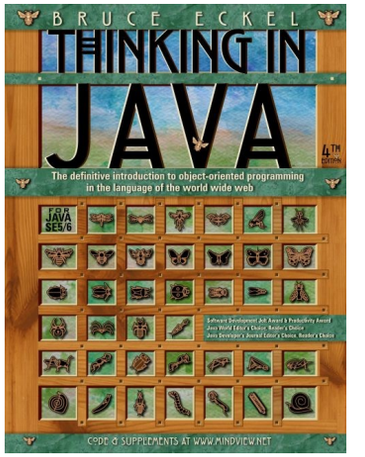
Introduction to Java and OOP

Hendrik Speleers

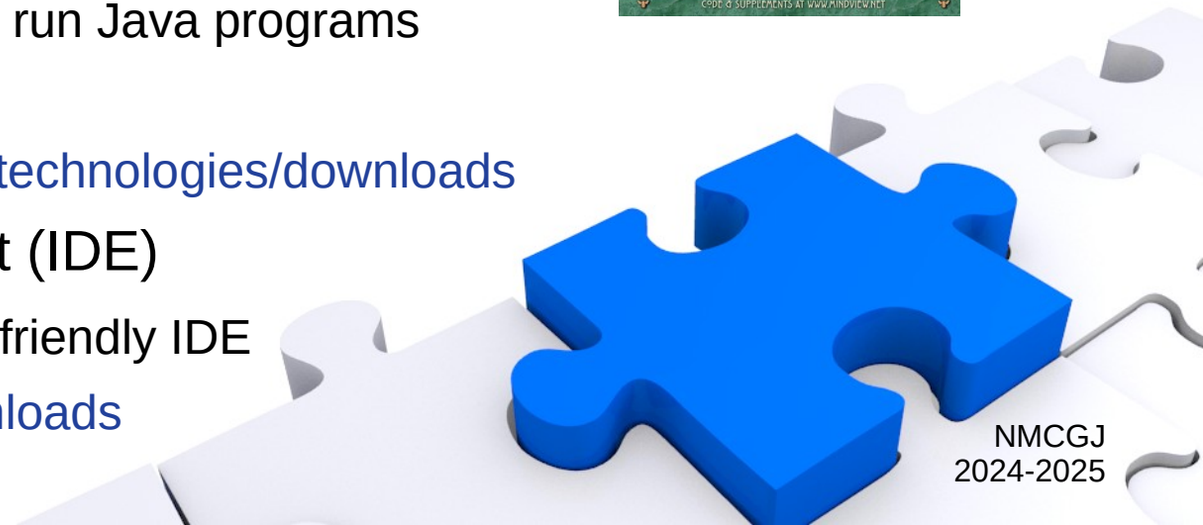
Introduction to Java

Università di Roma Tor Vergata

- Additional course material
  - “Thinking in JAVA” (4th edition) by Bruce Eckel
  - Free download: https://www.mindviewllc.com



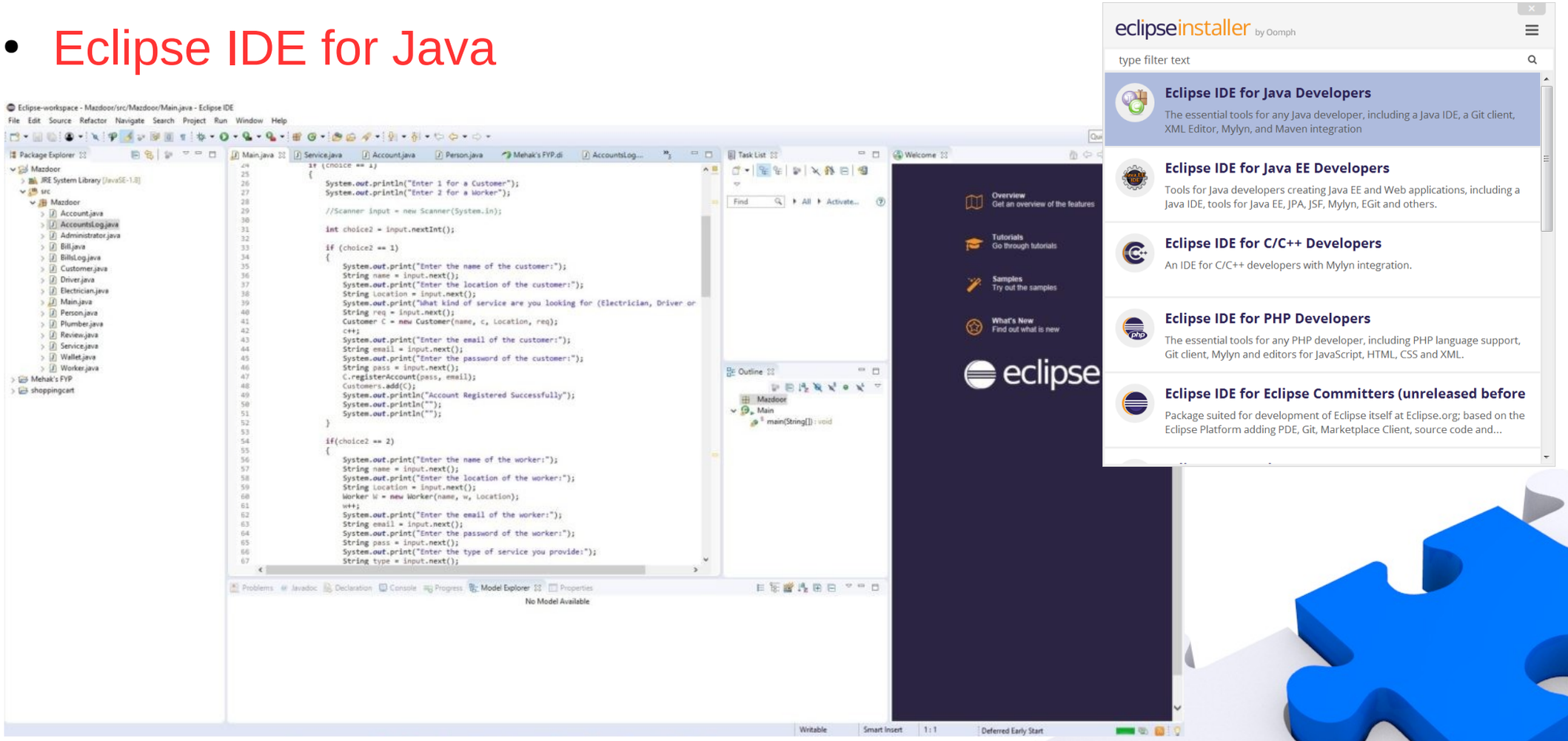
- Java programming
  - Java Development Kit (JDK) from Oracle
    - Includes Java Runtime Environment (JRE) to run Java programs
    - Includes tools for Java development
    - Free download: https://www.oracle.com/java/technologies/downloads
  - Java Integrated Development Environment (IDE)
    - Eclipse IDE for Java: very powerful and user friendly IDE
    - Free download: https://www.eclipse.org/downloads



Introduction to Java

Università di Roma Tor Vergata

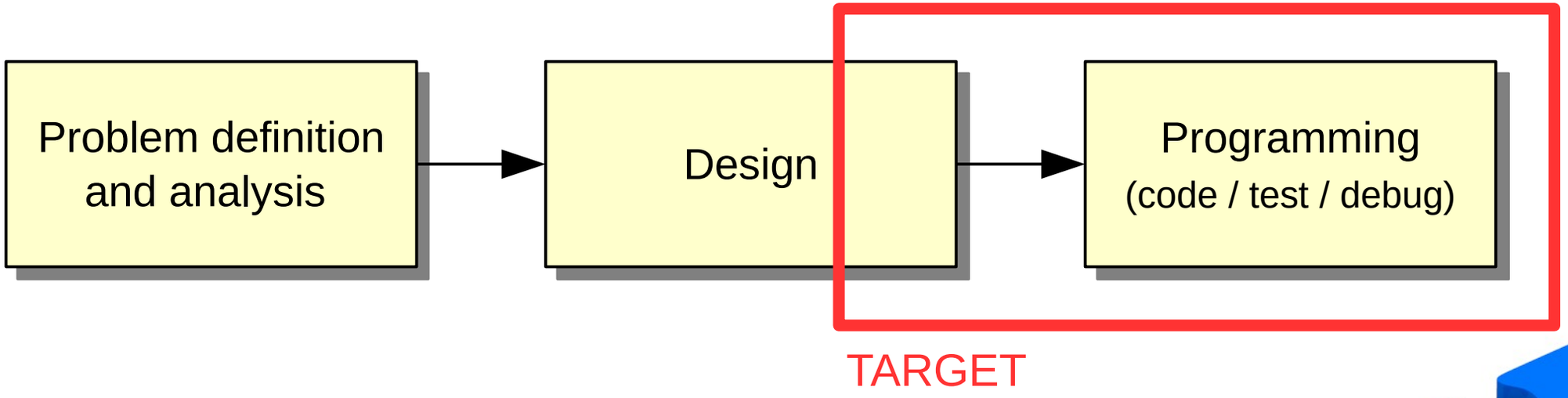
- Eclipse IDE for Java



Introduction to Java

Università di Roma Tor Vergata

- Programming
  - Part of the software development process

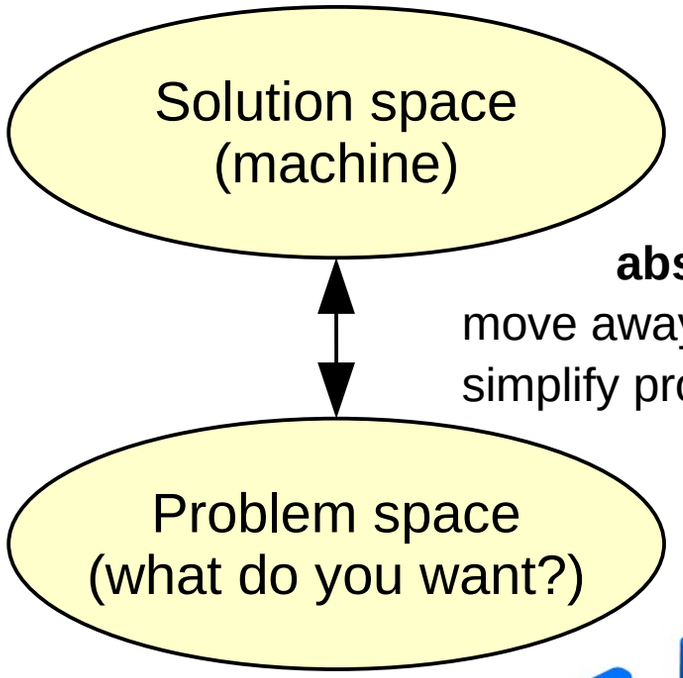


- Abstraction: problem space vs. solution space

Introduction to Java

Università di Roma Tor Vergata

- Programming languages: level of abstraction
  - Low-level programming language
    - Easy conversion to machine code
    - Relatively non-portable
  - High-level programming language
    - Higher abstraction: in need of compiler
    - More readable, more portable
  - Structured: Pascal, C, …
  - Object-oriented: C++, **Java**
    - Describe problem in terms of problem elements: OBJECTS

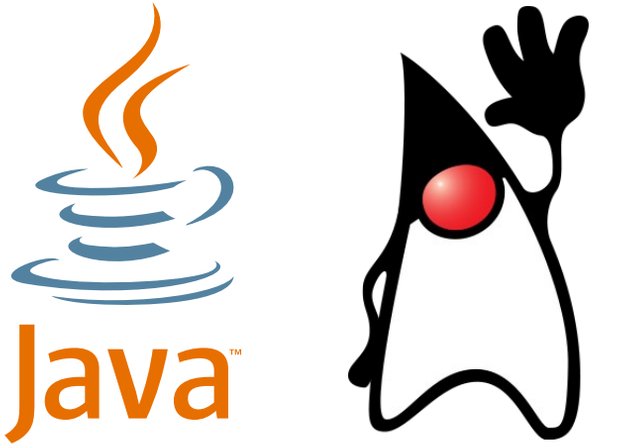


abstraction:
move away from machine
simplify problem description

Introduction to Java

Università di Roma Tor Vergata

- Java
  - Developed at Sun Microsystems (now Oracle), 1995
    - First intended for programs in small devices
    - Syntax based on C and C++
  - Two types of Java programs: applications – applets
  - Platform independent: highly portable
    - Java code (*.java) is compiled to byte code (*.class)
    - Byte code is executed in Java Virtual Machine (JVM)
  - More user-friendly language than C/C++
    - Memory management: garbage collector



Java's logo and mascot (Duke)

Introduction to Java

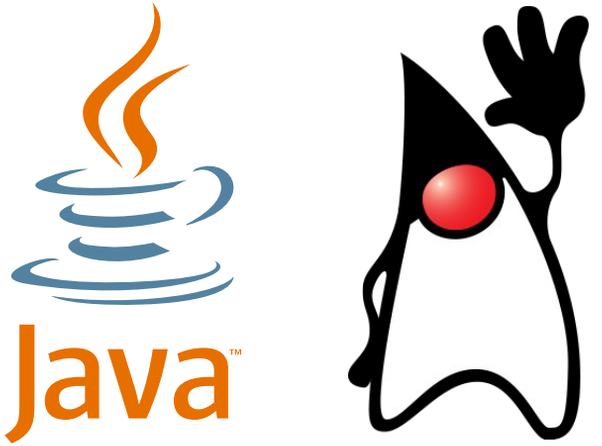
Università di Roma Tor Vergata

- Java
  - History

JDK 1.0	1996
JDK 1.1	1997
J2SE 1.2	1998
J2SE 1.3	2000
J2SE 1.4	2002
J2SE 5.0	2004

Java SE 6	2006
Java SE 7	2011
Java SE 8	2014
Java SE 9	2017
Java SE 10	2018
new every 6 months	

young and very active language

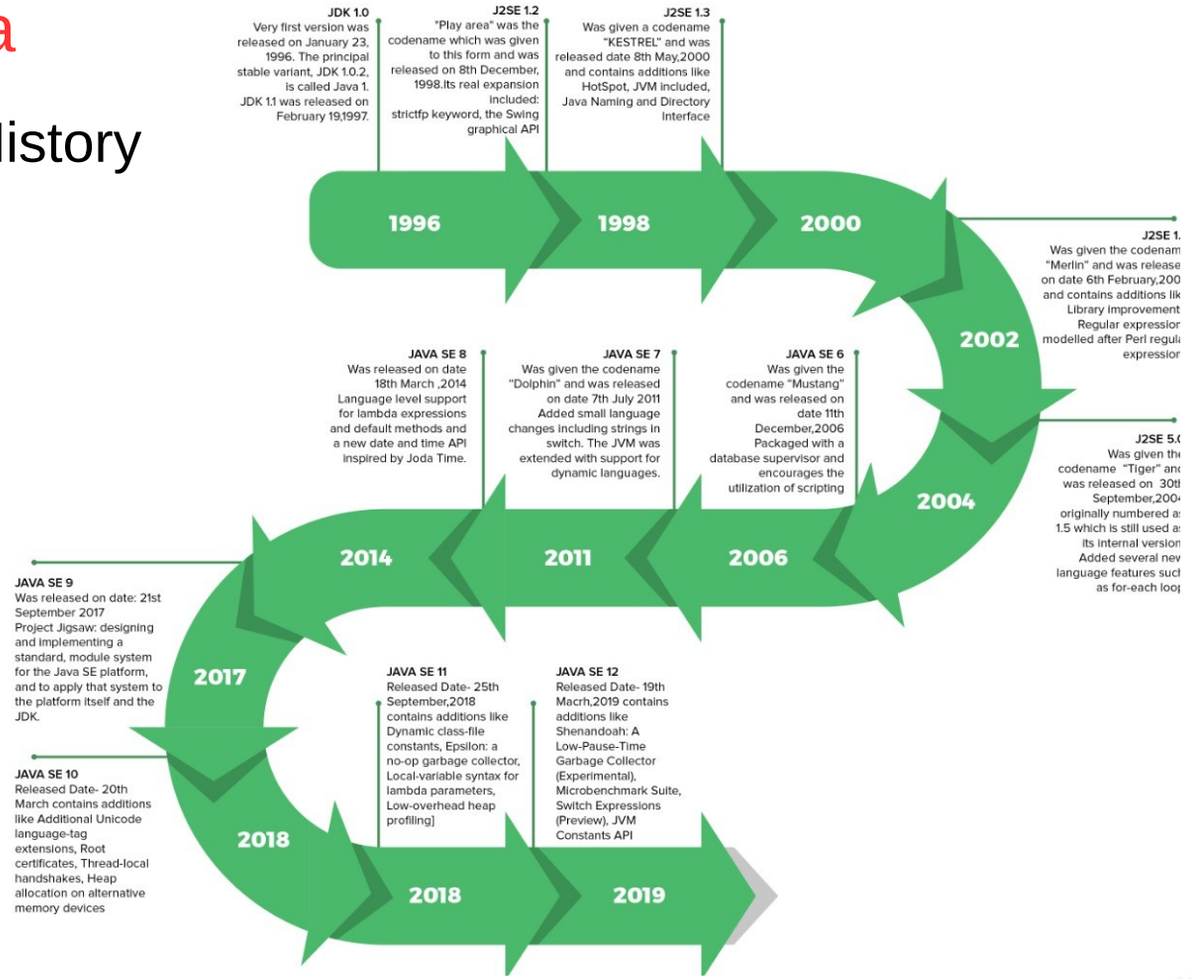


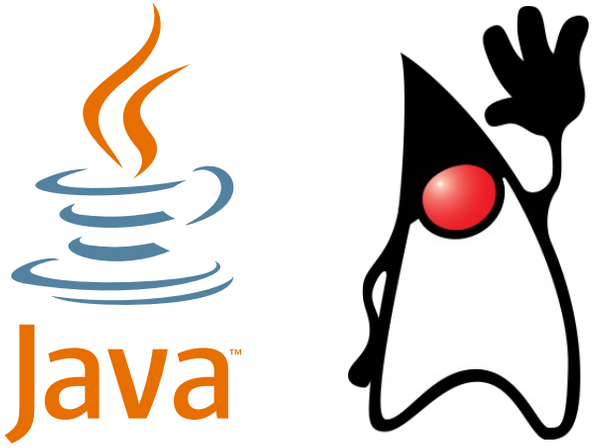
Java's logo and mascot (Duke)

Introduction to Java

Università di Roma Tor Vergata

- Java
  - History



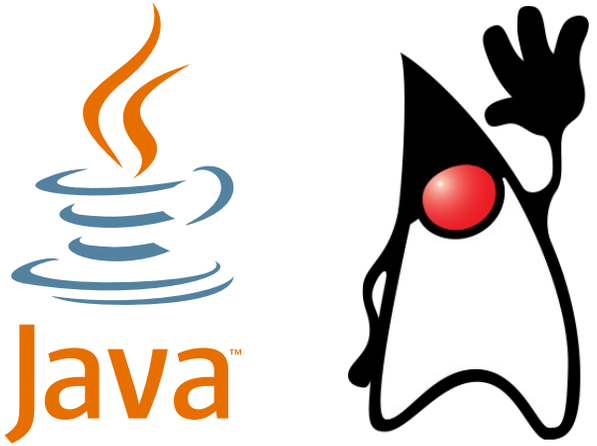


Java's logo and mascot (Duke)

Introduction to Java

Università di Roma Tor Vergata

- Java
  - Java in numbers (according to Oracle)
    - 95% of enterprise desktops run Java
    - 1 billion Java downloads each year
    - 9 million developers worldwide
    - #1 programming language
    - ...



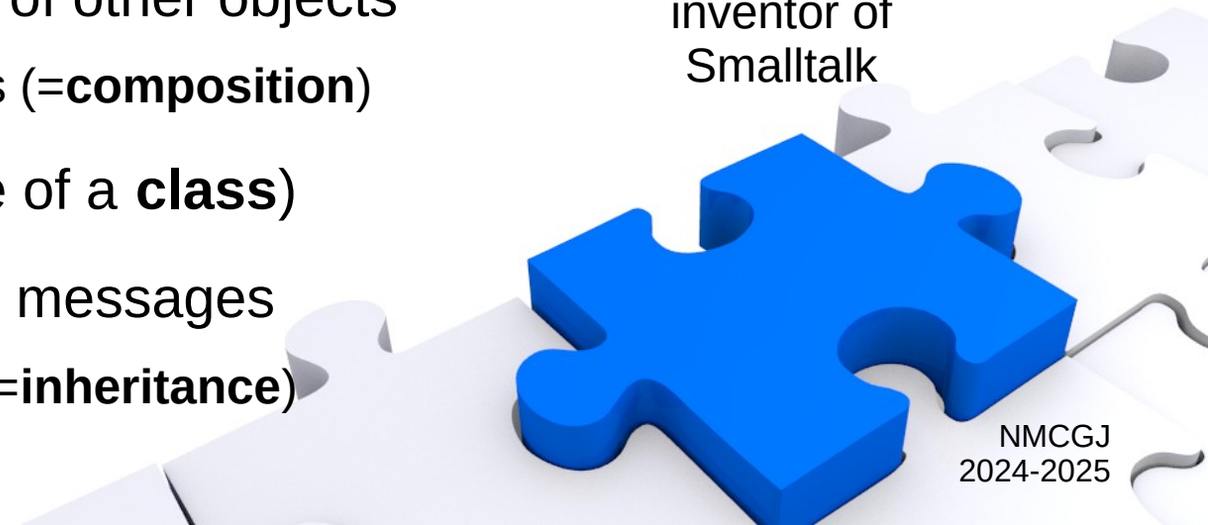
Java's logo and mascot (Duke)

Objects and Classes

Università di Roma Tor Vergata

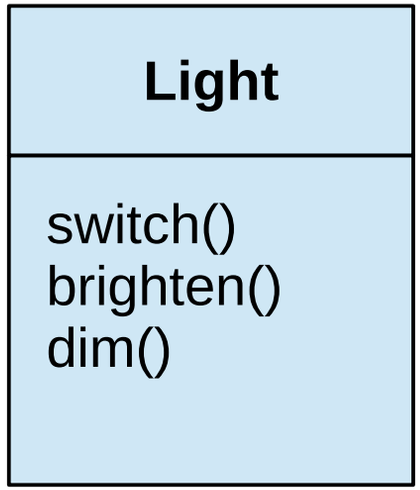
- Alan Kay's 5 rules for Object-Oriented Programming (OOP)
  - Everything is an **object**
    - An object is a fancy variable (storing data) + can perform operations
  - A program is a bunch of communicating objects
    - Objects are communicating by sending **messages**
  - Each object has its own memory made up of other objects
    - Hiding complexity behind simplicity of objects (=**composition**)
  - Every object has a type (= it is an instance of a **class**)
  - All objects of same type can receive same messages
    - Families of types can be under a base type (=**inheritance**)

Alan Kay, inventor of Smalltalk



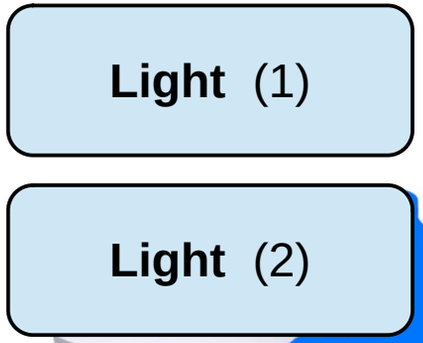
Objects and Classes

- Programming with objects: the interface
  - Class name
  - Interface



Class diagram
according to UML standard
(Unified Modeling Language)

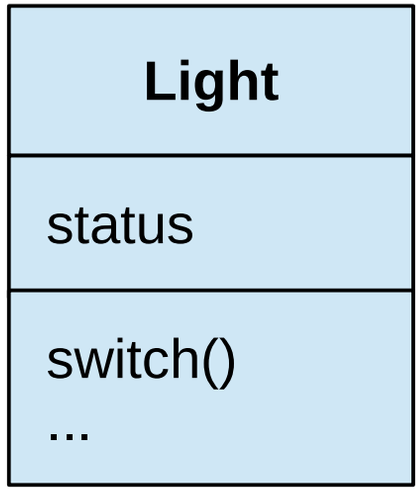
  - Object `Light light1 = new Light();`
  - Message `light1.switch();`



Objects and Classes

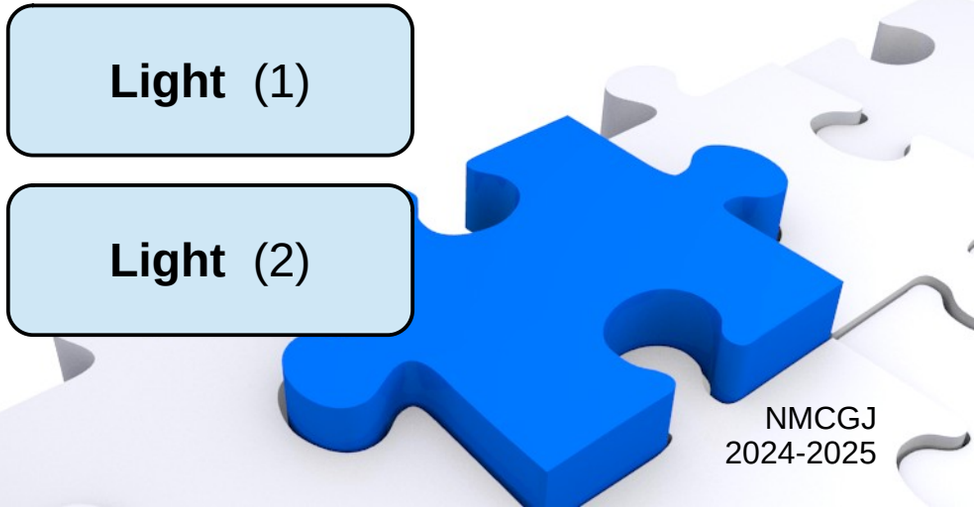
Università di Roma Tor Vergata

- Programming with objects: the interface
  - Class name
  - Data
  - Interface



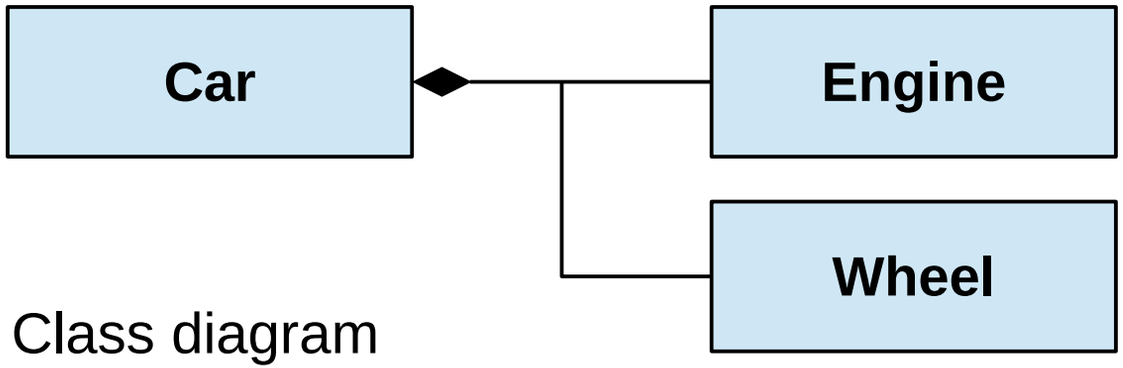
Class diagram
according to UML standard
(Unified Modeling Language)

- Object `Light light1 = new Light();`
- Message `light1.switch();`



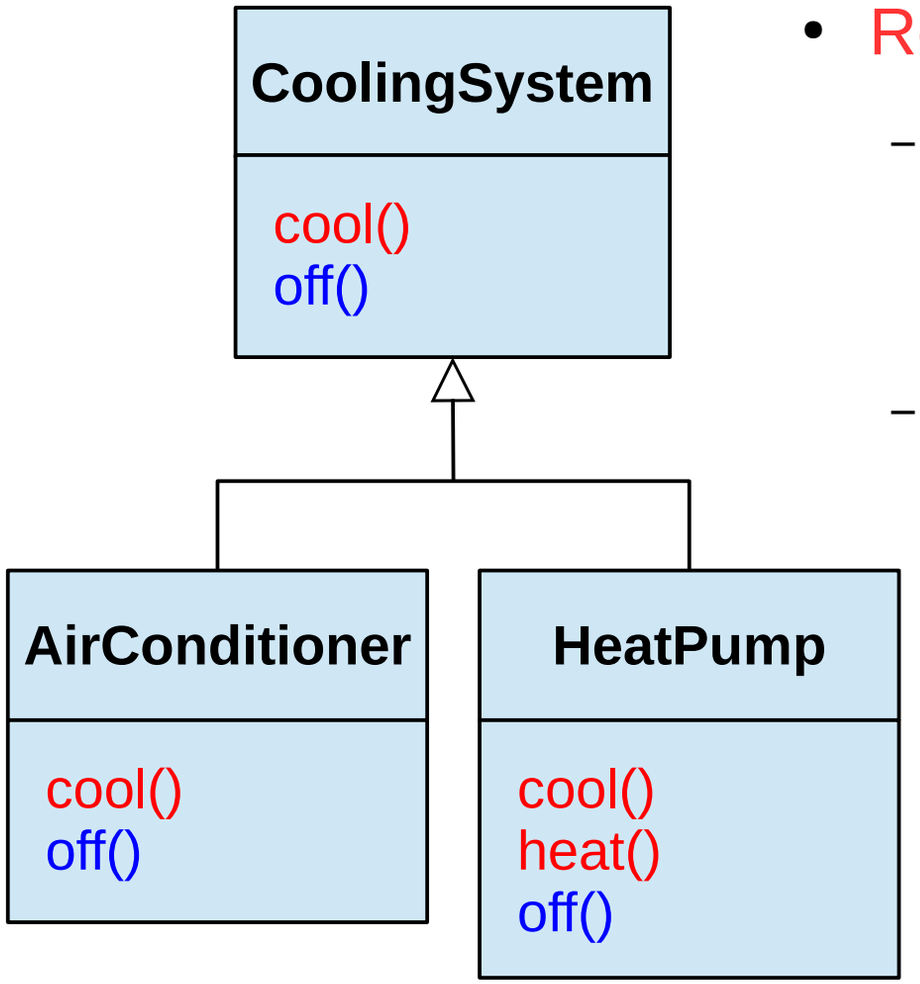
Objects and Classes

- Reusing the implementation
  - Good design: reuse of classes, once created and tested
  - Simplest way: creating member objects of a class
  - Composing a new class from existing classes
    - Composition is as a “has-a” relationship



Class diagram

Objects and Classes



- Reusing the interface
  - **Inheritance**: derive functionality from a parent class
    - An “is-a” relationship: override parent class functions
    - An “is-like-a” relationship: override + add new functions
  - **Polymorphism**
    - Code assumes parent class, but not specific child class (upcasting)
    - Add new child classes without effort
    - Method call determined at run-time

Objects and Classes

Università di Roma Tor Vergata