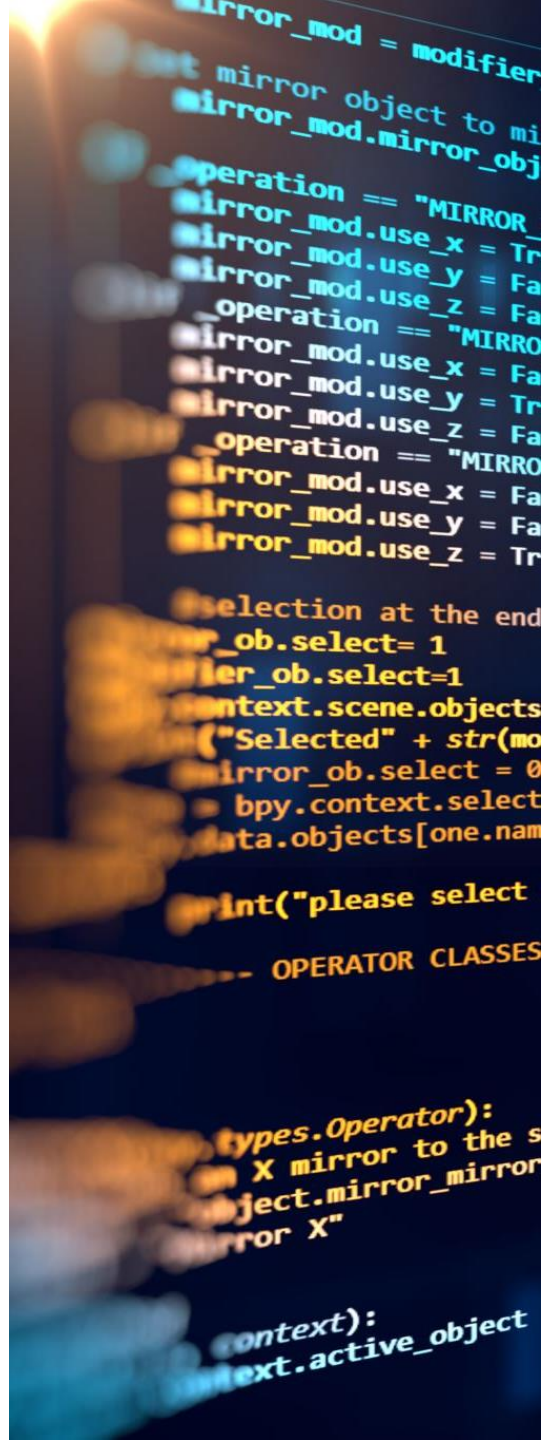
CS 1027
Fundamentals of Computer Science II

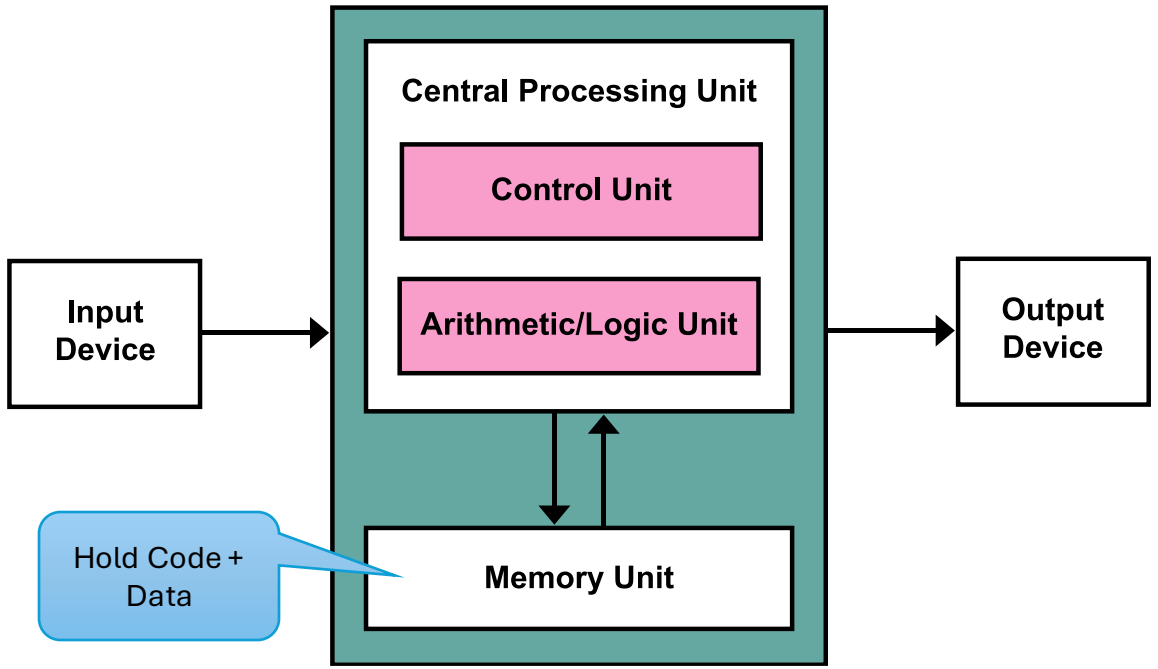
Object Oriented Design

Ahmed Ibrahim



Computer Model

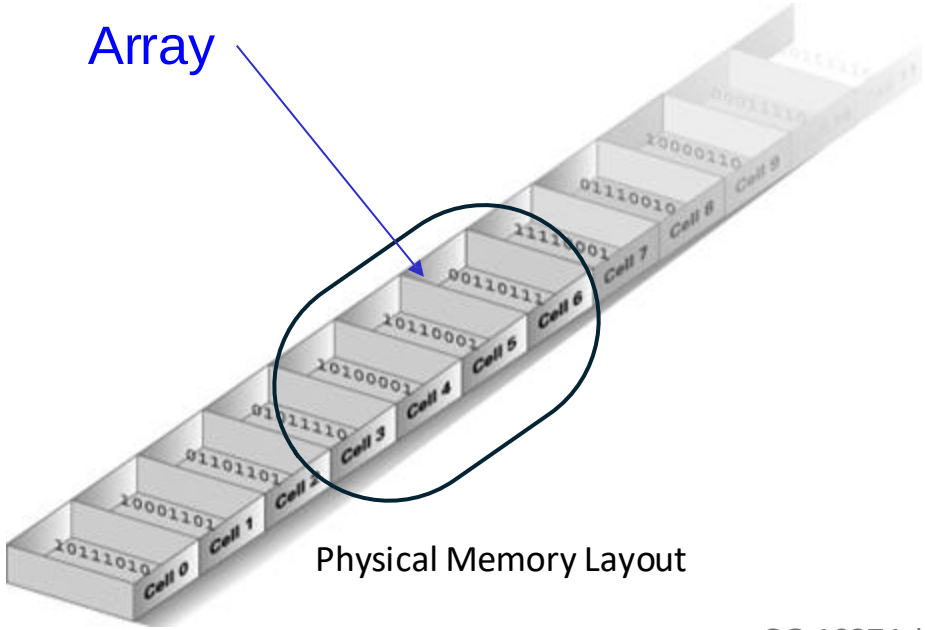
- To design a computer program, we need to understand how a computer works, stores information, and executes a program.



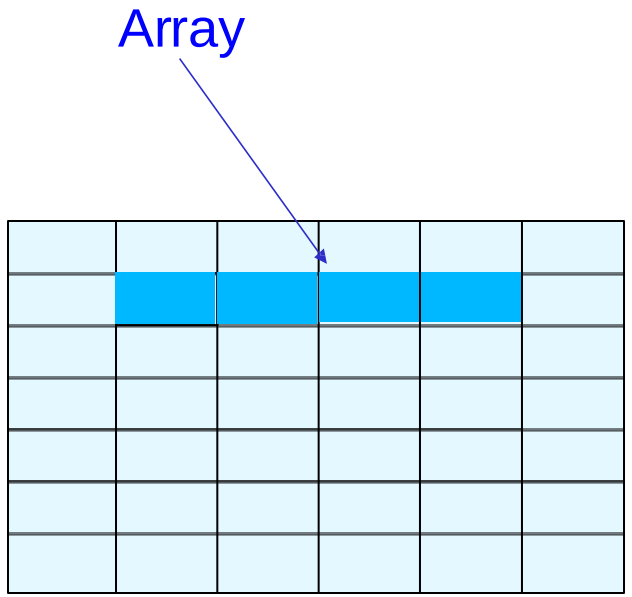
The simplest model of a computer is the Von Neumann Model

Recall: Computer Memory

- Computer memory consists of a set of cells, each with a unique address.
- Information is stored in memory in binary format (0s and 1s).
- A 0 or a 1 is a bit.
- A set of 8 bits is a byte.



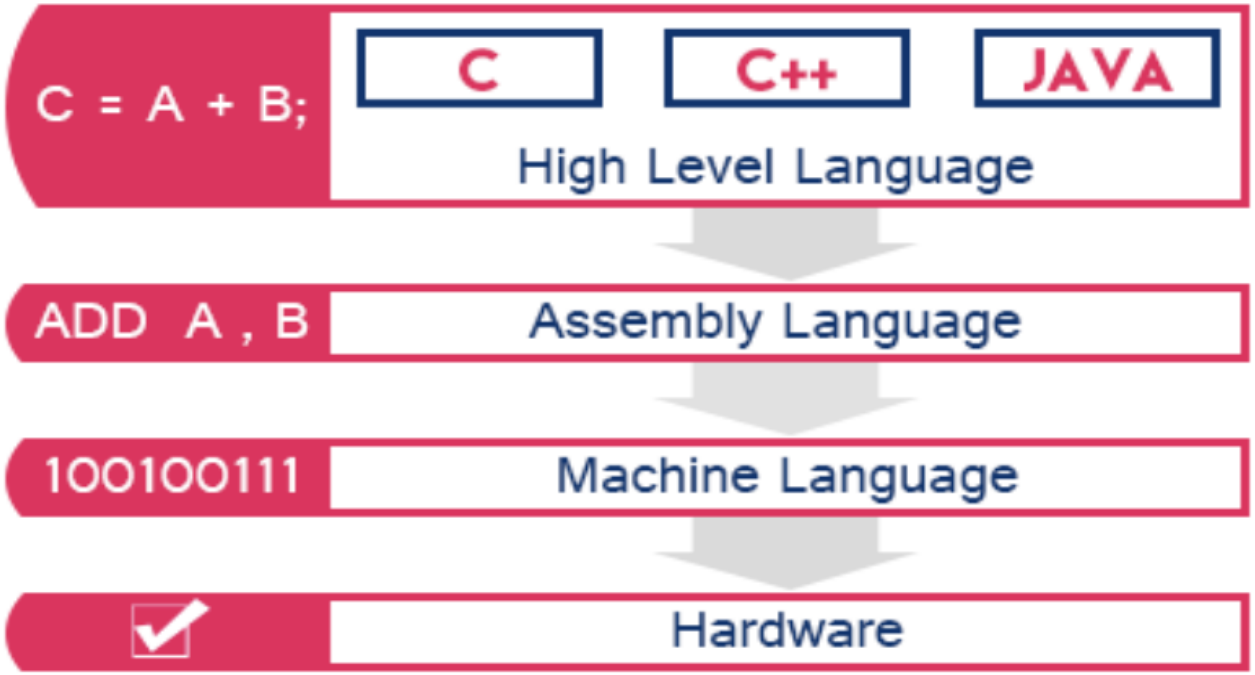
Physical Memory Layout



Logical 2D Structure

Machine Language

- A computer can only understand code written in a special language called machine code, machine language, or executable code.
- Machine language is binary.
- Each processor has its own instruction set, a collection of commands specific to its architecture that dictate how the machine code interacts with the hardware.



Programming Languages

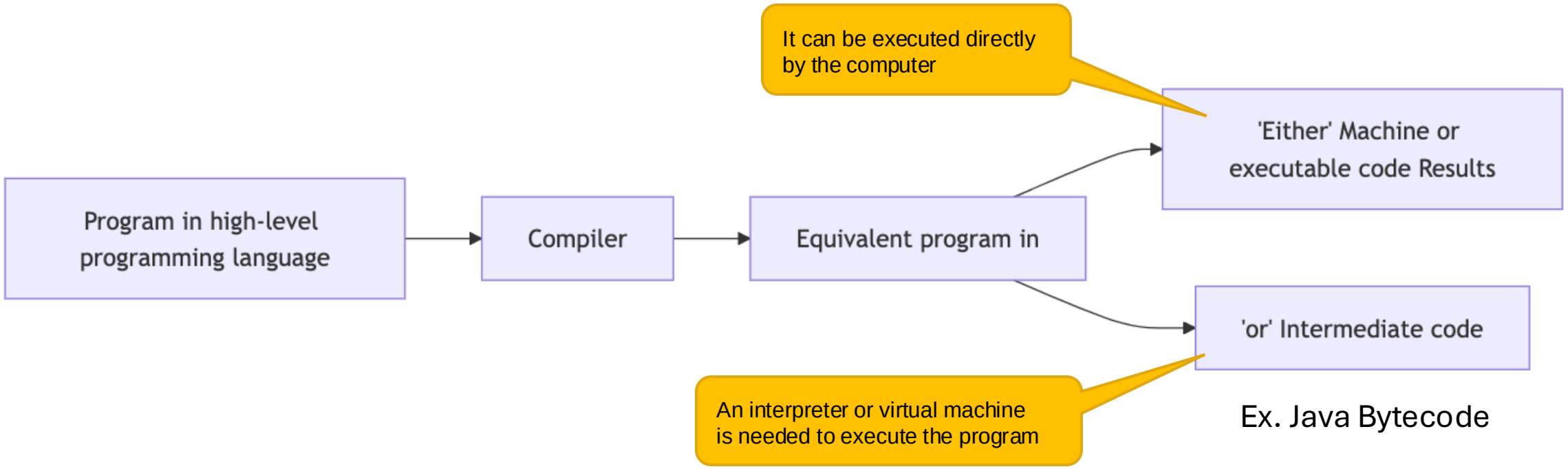
- Machine language is composed of binary code, which is difficult for humans to read or write. For instance, machine code on an Intel 8086 processor might appear as a long sequence of binary digits (Hello):

  01001000 01100101 01101100 01101100 01101111

- To make programming easier, we use high-level languages like Python or Java, which are more human-readable.
- However, since computers can only process machine code, high-level languages must be translated into machine language for execution.

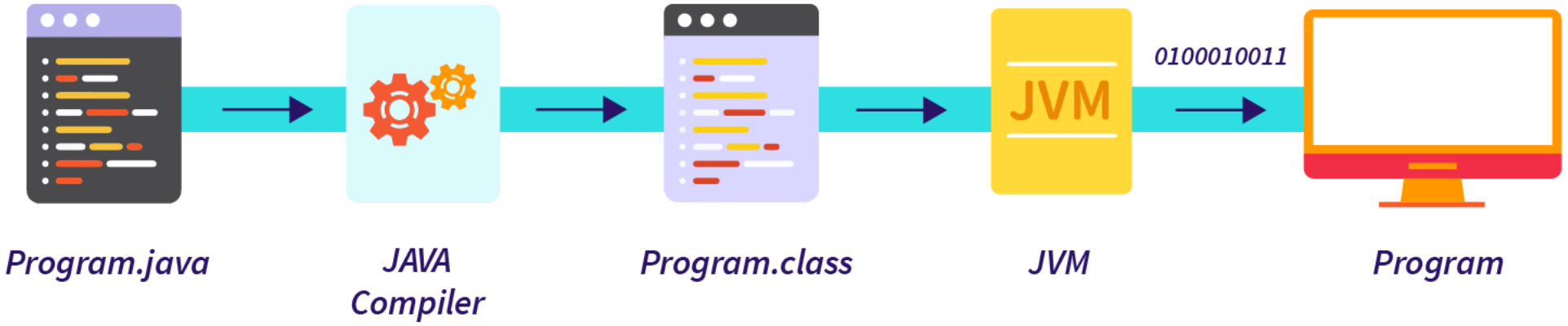
Compiler

- A compiler is a program that translates code in a programming language into another, simpler language that is easier to understand by a computer

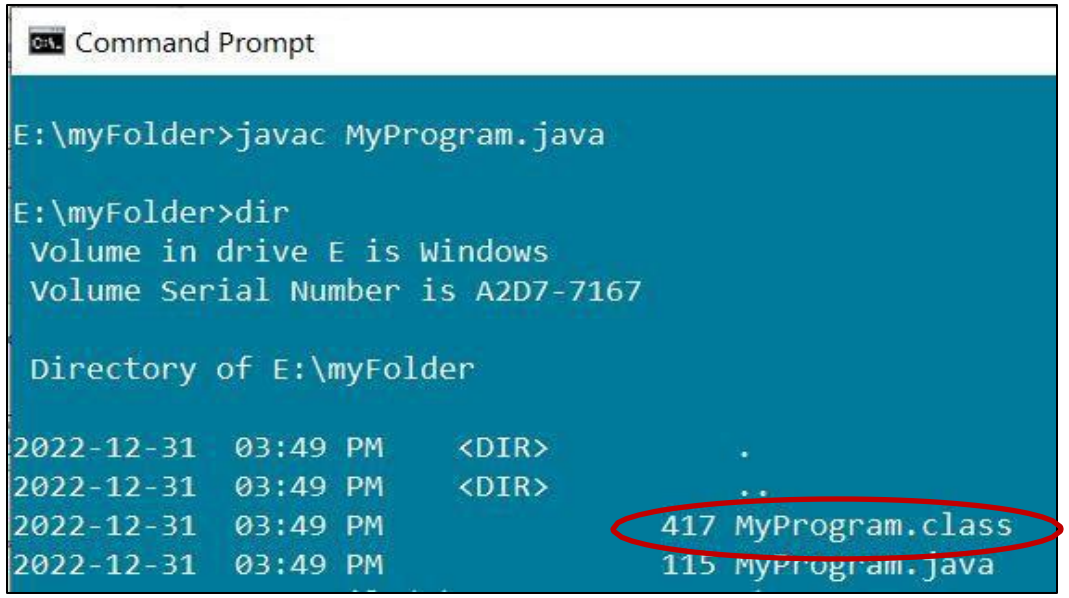


Executing a Java Program

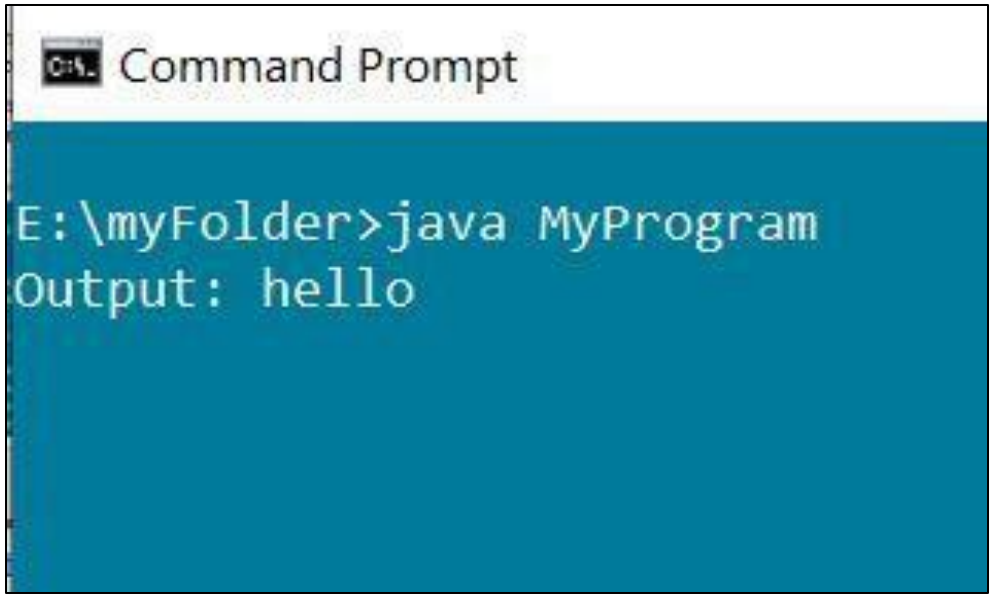
- Java IDE invokes the Java compiler automatically?!



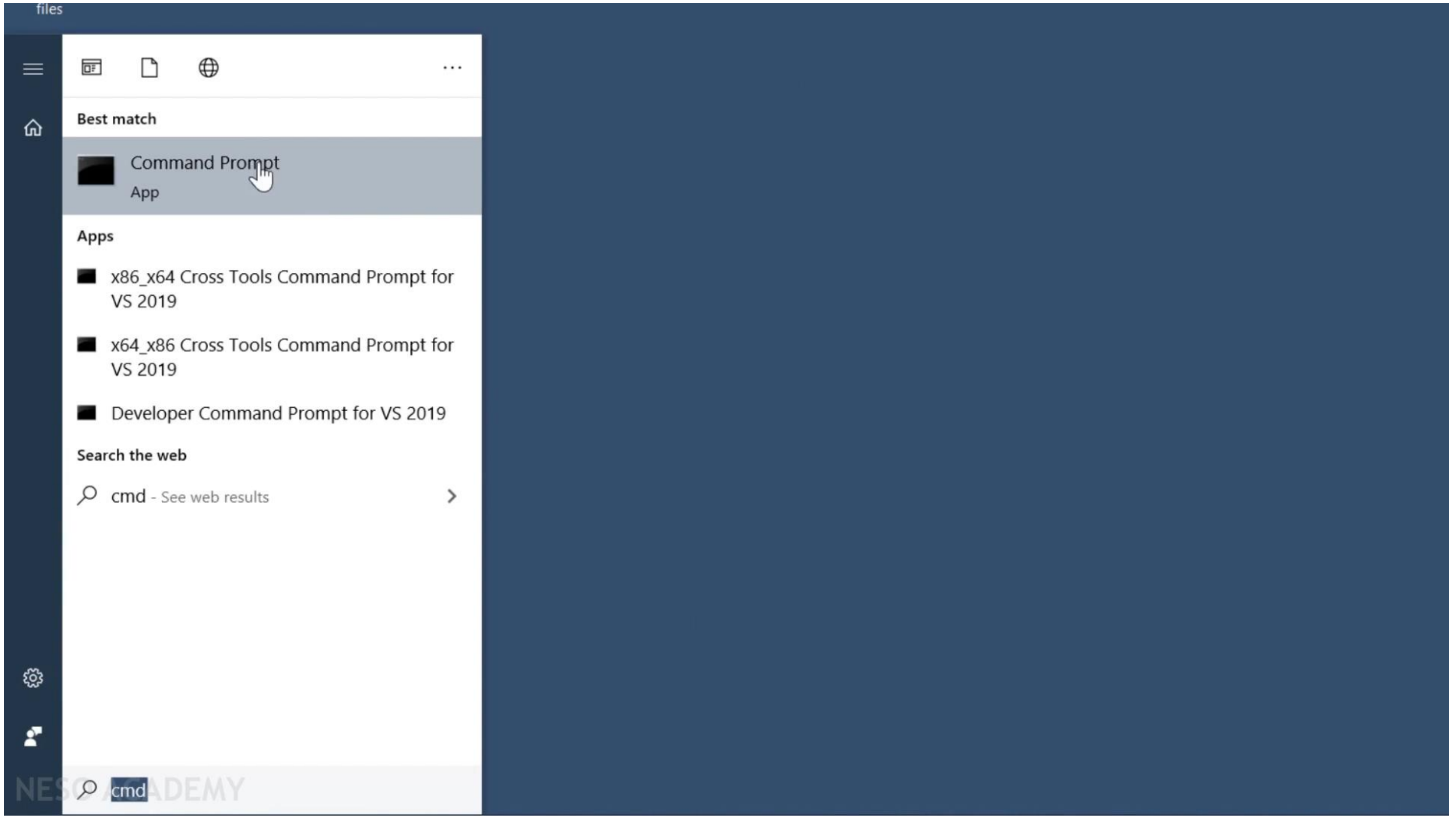
Compiling & Executing a Java Program



Compiling

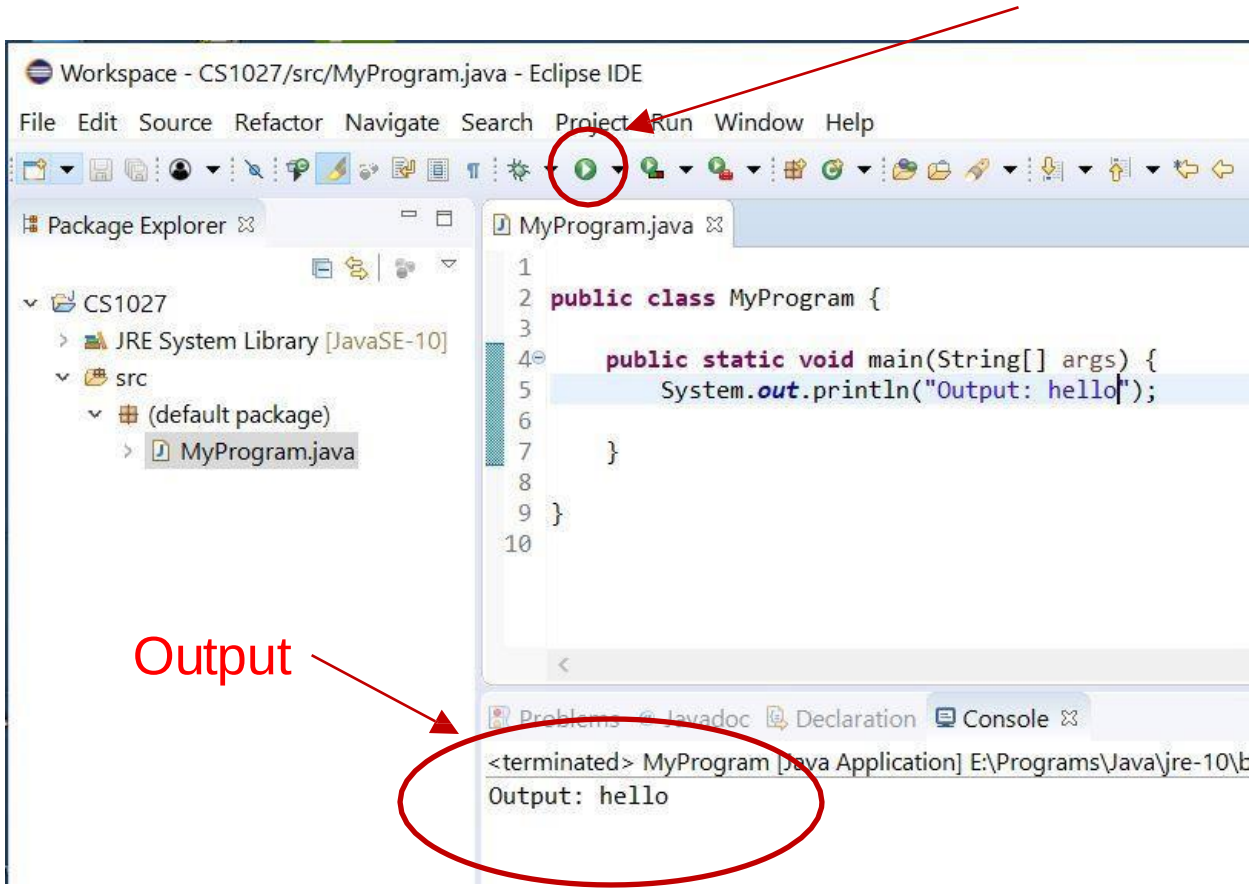


Executing


files
Best match
Command Prompt
App
Apps
x86_x64 Cross Tools Command Prompt for VS 2019
x64_x86 Cross Tools Command Prompt for VS 2019
Developer Command Prompt for VS 2019
Search the web
cmd - See web results
cmd

Executing a Java Program from Eclipse

- Eclipse invokes the Java compiler automatically as you type your Java code.
- To execute the program, click the “Run” button.



Compiler

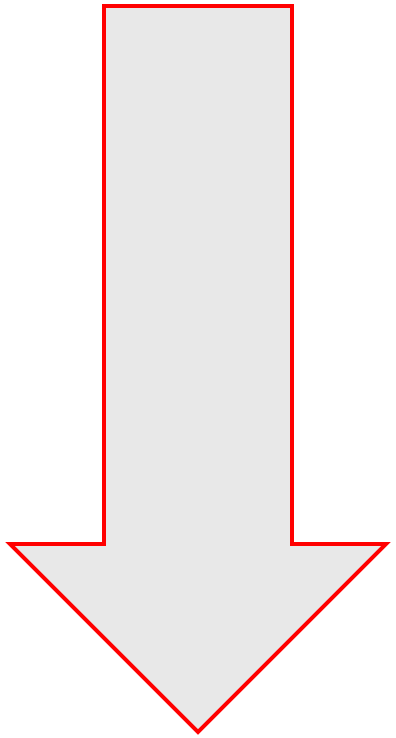
```
int b, c;
int a = 1;
if (foo() == 1) b = 2;
else b = “house”;
c = a + b;
```

The compiler will mark this error.

- The compiler also checks that the program is correctly written according to the syntax of the programming language

Designing a Program

- To create a program, there are several steps that we need to follow:
  - **Specification**–Understand what the program is required to do
  - **Design**–Determine the steps that the program needs to perform to satisfy its specifications.
  - **Implementation**–Translate the designed solution to a programming language
  - **Testing and Debugging**–Verify that the program works correctly and fix bugs or errors
  - **Verification**–Verify that the program works correctly and fix bugs or errors

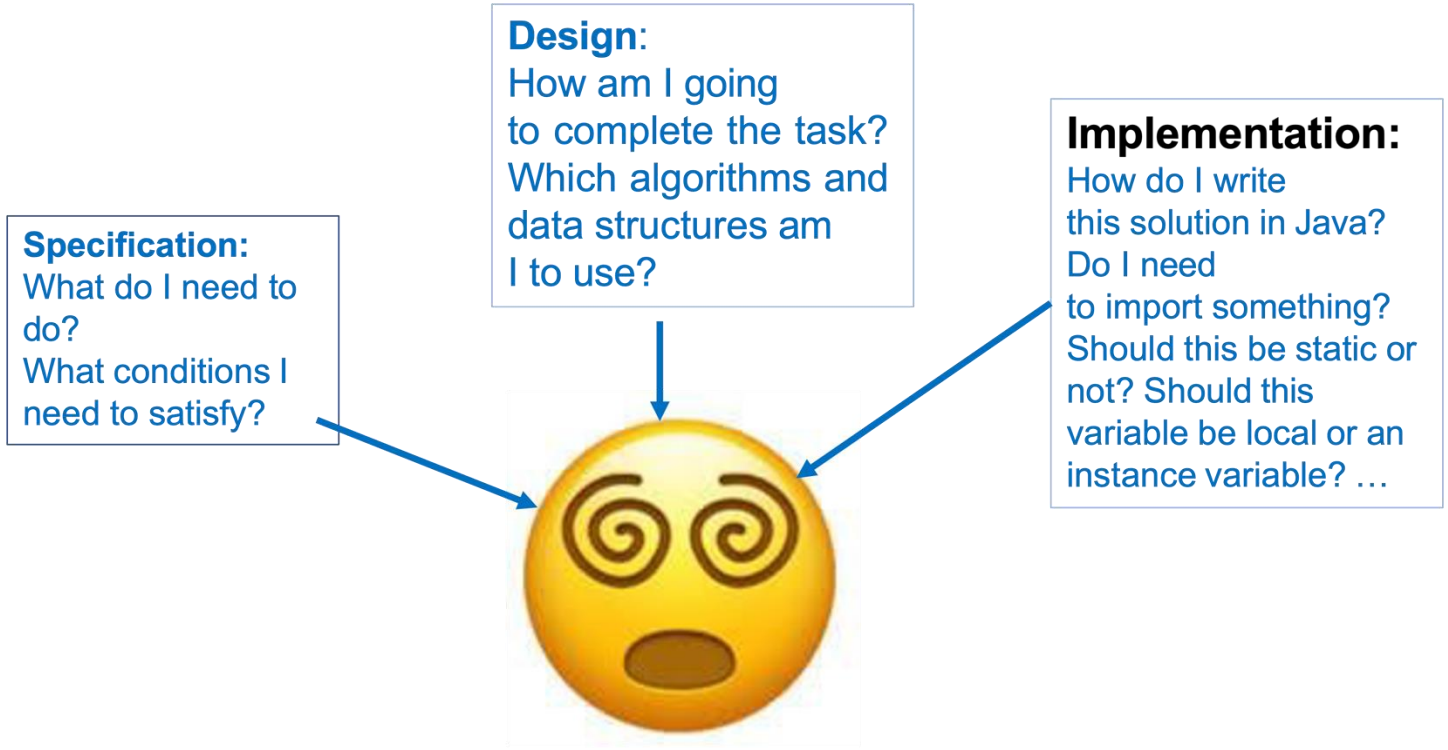


Design

- The most complex step is the **Design**.
- To design a program, you should **NOT** write Java code directly from the program's specification as then you need to simultaneously think about
  - how to solve the problem for which you are designing the program, and
  - how to express that solution in Java

Bad Design Approach

- Try to design the solution to a problem directly in Java from the program's specification.
- This requires you to think about many things at the same time:



Good Program Design

- Divide the task of writing a program into several simpler tasks:
  - Read the specifications of the assignment and understand what you need to do
  - Then, think about how to design your program.
  - Use paper and pencil to come up with a solution and write a detailed description of it using pseudocode. It's important to ensure that your **pseudocode** solution is correct, as this will save you time and effort in the coding phase. More about pseudocode later.
  - Now translate your pseudocode to Java
  - Test, and debug if needed.

Example

Problem Statement

Implement a C program that includes a function foo() , which returns the value 1. In the main() function, initialize the variable a to 1. Use an if-else statement to check the return value of foo(). If it returns 1, set the variable b to 2; otherwise, set b to 3. Then, calculate the value of c as the sum of a, b, and an additional 1. Finally, print the value of c. The expected output of the program is 4 when foo() returns 1.

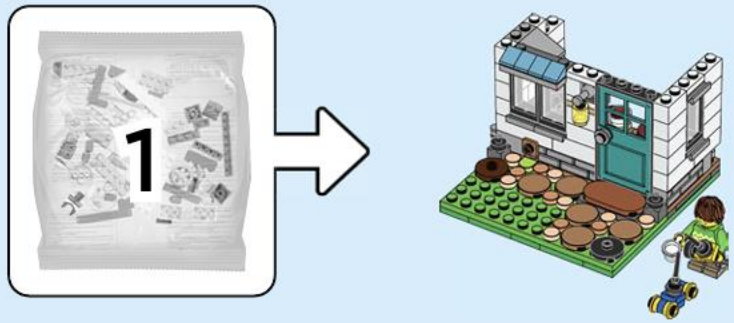
```
# pseudocode

Start
Function foo:
    Return 1

Main Function:
    Declare variable a and set it to 1
    Calculate the value of c as the sum of a, the result of foo(), and 10
    Print the value of c

End
```

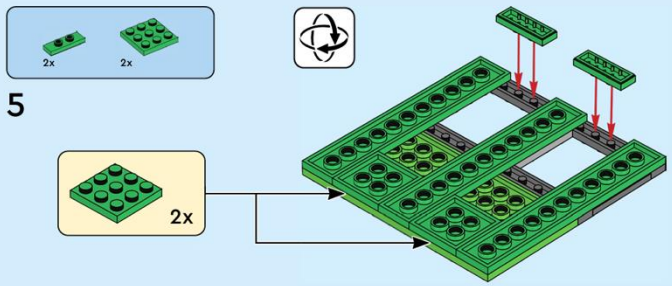
Object-Oriented programming principles

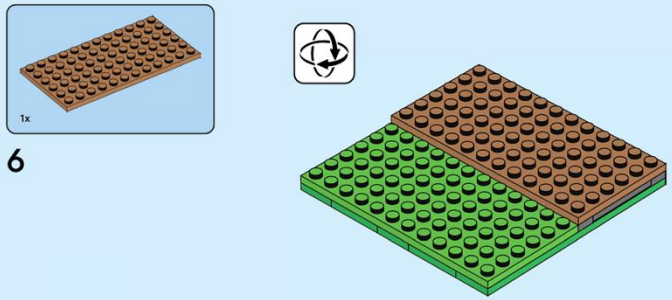
- There are different techniques to simplify the design of computer programs.
- This course will study program design techniques based on object-oriented programming principles.

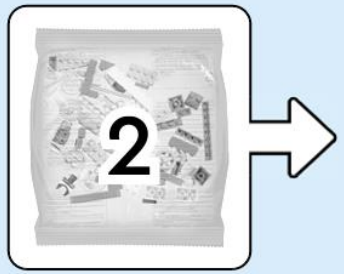
Program Design

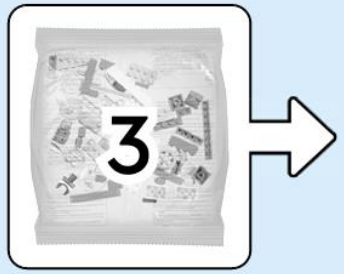
- Given a complex problem that we need to solve with a computer program:
  - we divide it into simpler, smaller sub-problems,
  - we design programs for the sub-problems and
  - we combine them to get the whole program

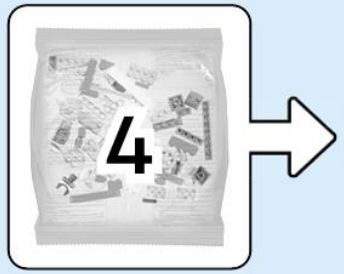

1


2x
2x
5
2x


1x
6

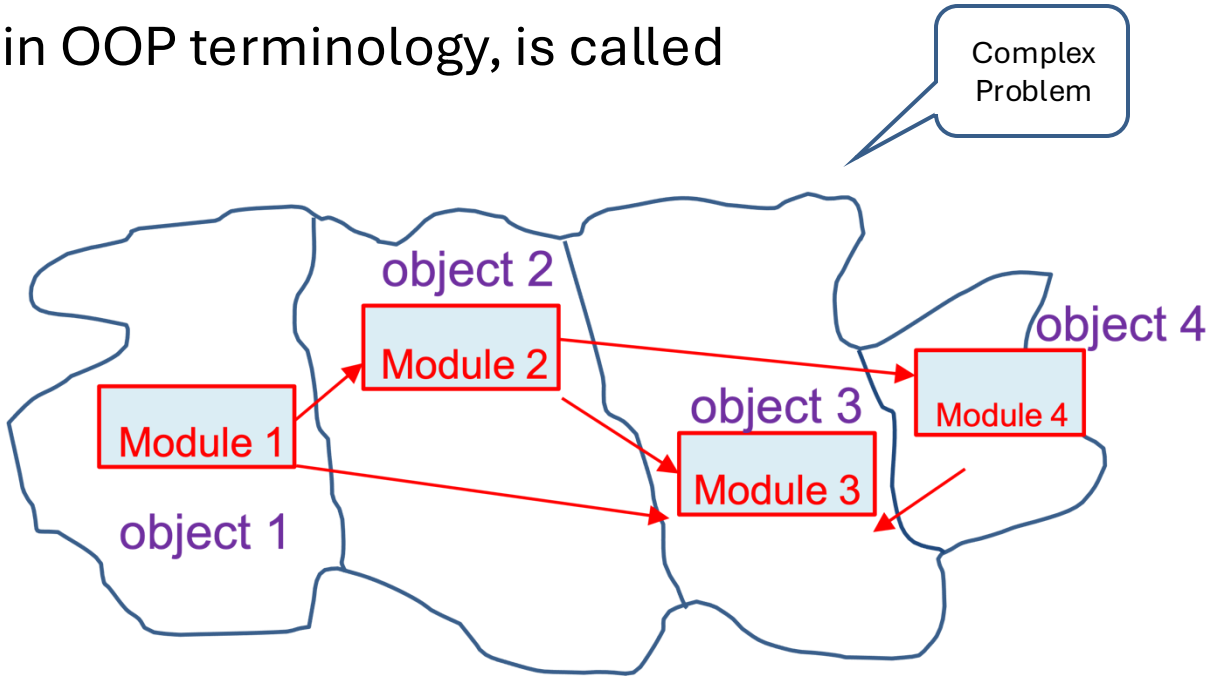

2


3


4

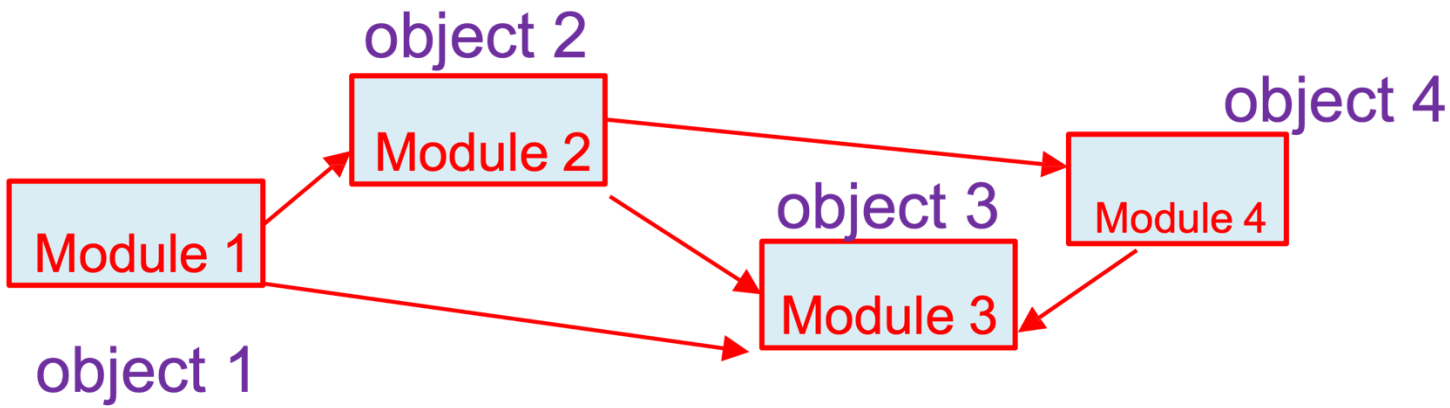
Modularity

- Each program designed for the sub-problems is called a **module**, or in object-oriented terminology, an **object**.
- Dividing a program into modules is what, in OOP terminology, is called **modularity**



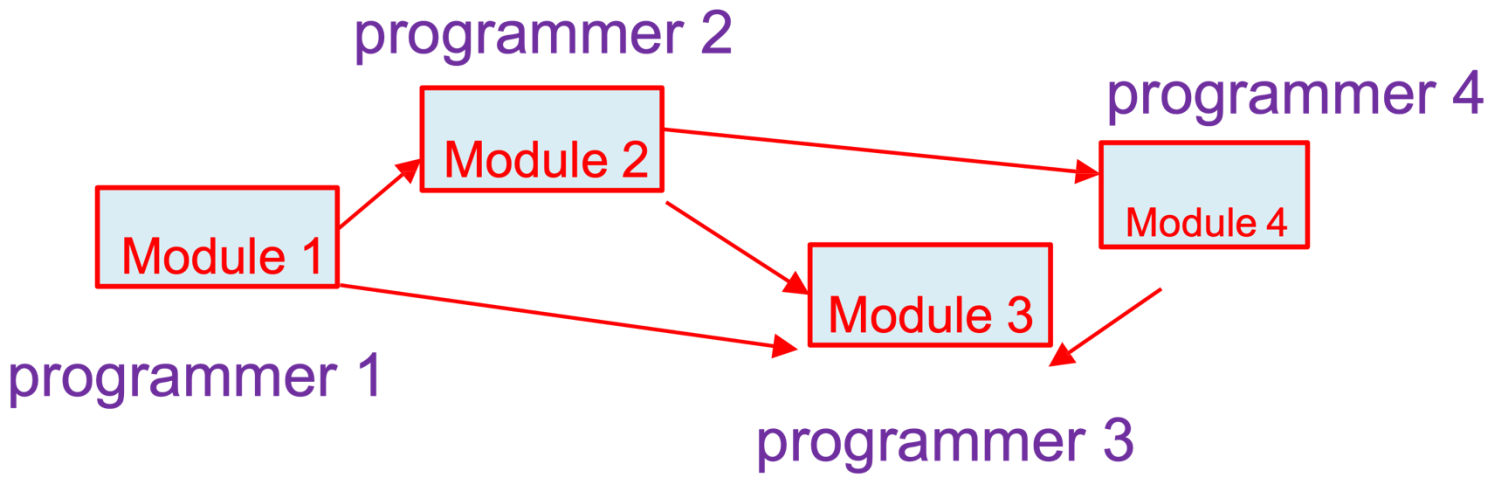
Module Design

- To simplify their design, we should design the modules to be independent of each other.
- To help achieve this, we use **encapsulation** and information-hiding



Module Design

- Modularity, encapsulation, and information hiding allow different programmers to be assigned a large software project. These programmers can then work simultaneously and independently on the part of the project assigned to them.

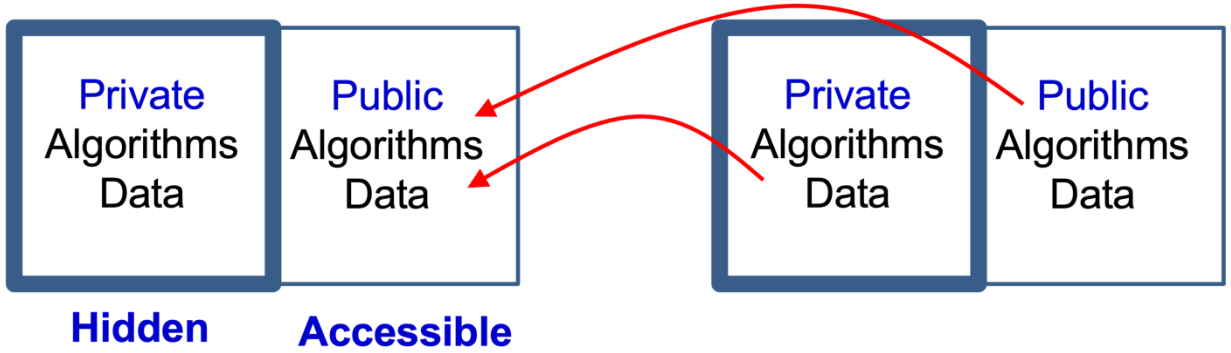


Encapsulation

- **Encapsulation**: Each module includes its own algorithms and data
- **Information hiding**: Details of the design of a module should be hidden from other modules to avoid complex module interactions and to keep the design of each module as independent from each other as possible.

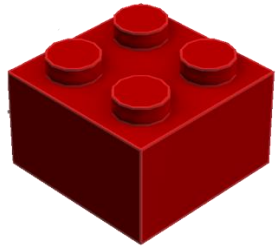
Abstraction

- Modules cannot be completely independent, as they need to cooperate to form a whole program
- Abstraction: Allows module interaction by making some algorithms and data of a module public, so they are accessible to other modules; other algorithms and data are private and inaccessible to other modules



Abstraction

- Recall that in OOP terminology, a module is called an object.
- The data of an object is called its properties, attributes, fields, or instance variables
- The algorithms of an object are called actions or behaviors, and in Java, they are implemented as methods or instance methods

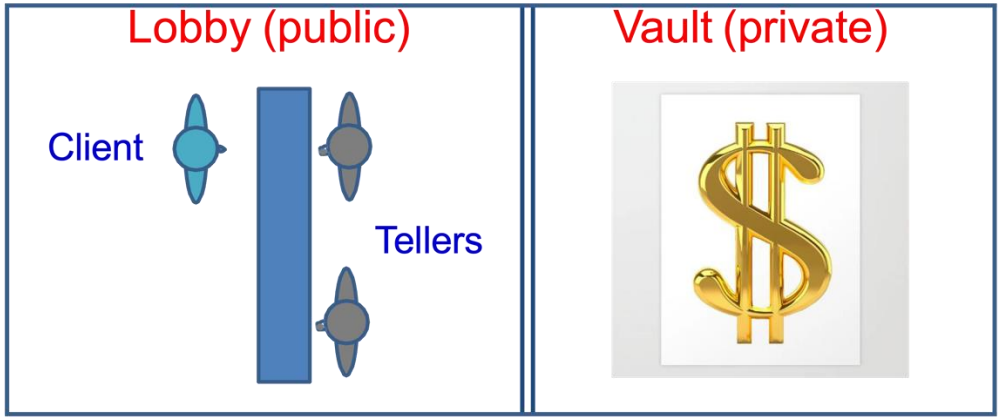


Object

Public	**Private**
Data (instance variables) Algorithms (methods)	Data (instance variables) Algorithms (methods)

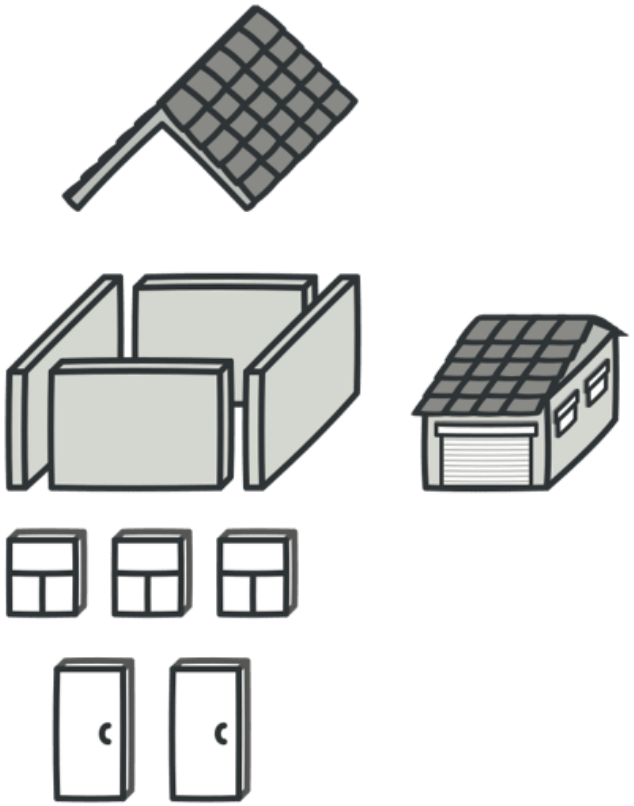
Private vs Public

- To understand the notions of private and public, consider how a bank works:
  - **Private**: the vault where money is stored
  - **Public**: the lobby where clients can interact with the bank tellers. Through the interaction with the tellers (public section of a bank) a client can have indirect access to the vault (private section of the bank)



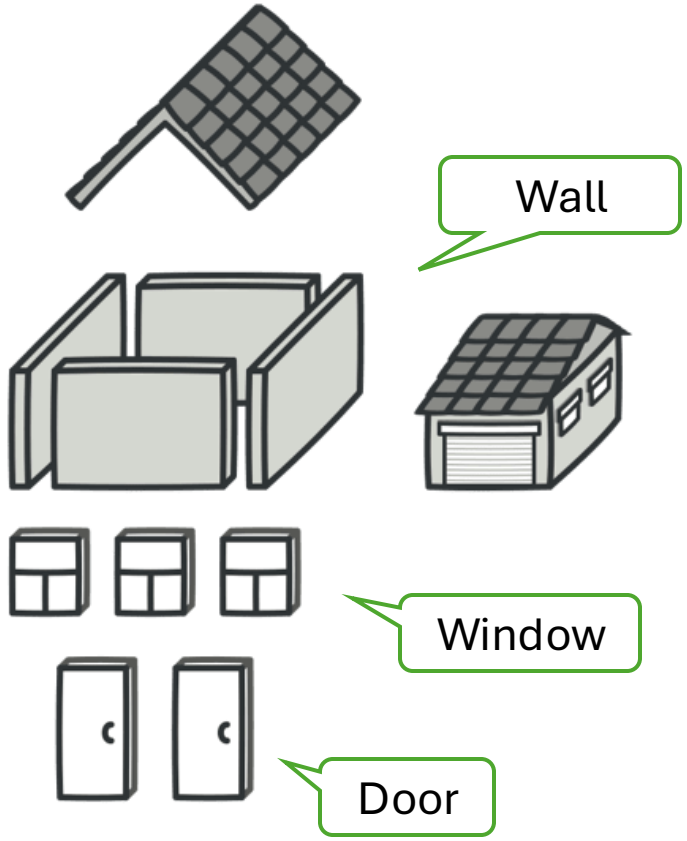
Objects and Classes

- A program can have several objects of the same class.
- We can think of a class as being a **template, pattern**, model, or definition for objects of that class
- In Java, a class definition must be stored in a file with the same name as the class and a .java extension.



Objects and Classes

- Every object belongs to a **specific class**
- A class specifies the **instance variables** and **methods** of an object, so a class definition consists of
  1. Instance variable declarations (also known as **fields** or **attributes**) determine its status.
  2. Method definitions (behavior)




thank you