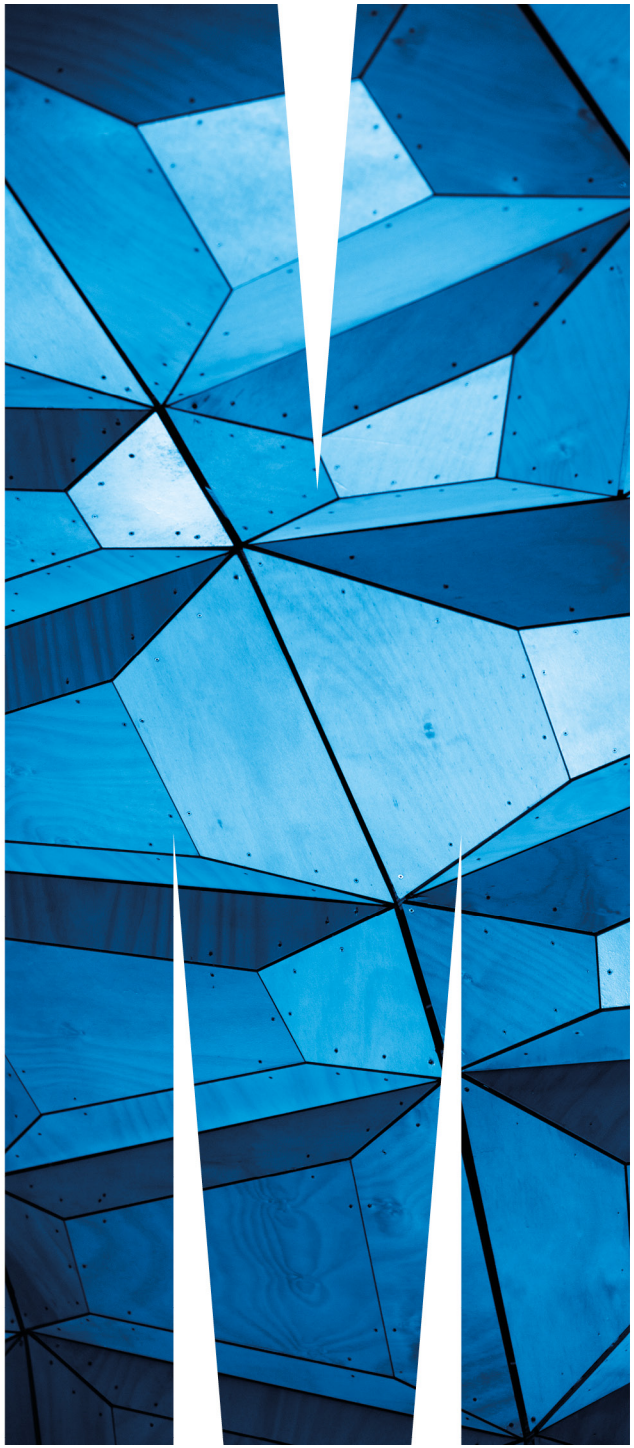
MONASH University

MONASH
INFORMATION
TECHNOLOGY



Introduction to Object-Orientation

FIT2099: Object-Oriented Design and Implementation

GROUP OF EIGHT AUSTRALIA

Lecture Outline

MONASH University

- What is OO?
- Why OO?
- OO languages
- What is Java
- A first Java program

What is a computer program?

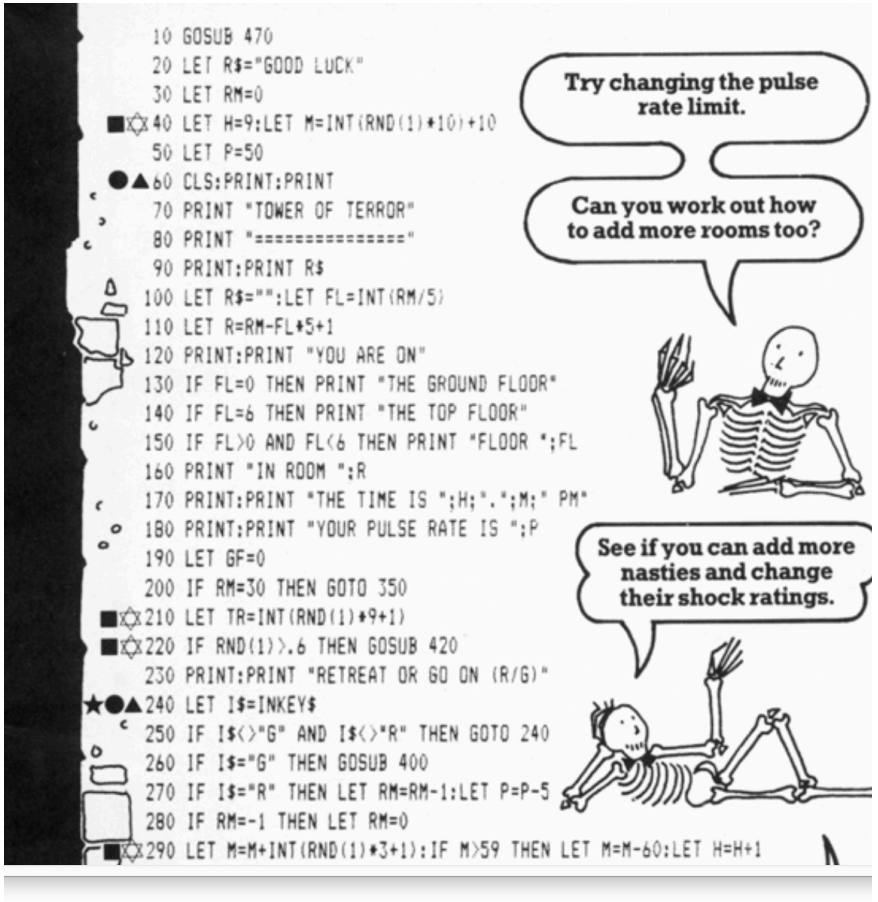
MONASH University

- A computer program is a set of instructions executed by a computer in order perform a particular task
  - At the most basic level, this is machine code executed by a CPU
- Higher level languages use different models to understand an construct programs. There are several programing paradigms, e.g.
  - Imperative
  - Functional
  - Procedural
  - Object-Oriented
- No matter what higher level language you write in, it still ends up as machine code eventually – but the model helps programmers to design and reason about their programs

The Dark Ages

MONASH University

- Languages such as Assembler, BASIC, FORTRAN
- Program was an unstructured list of statements.
- GOTO let you jump from any statement to any other statement
- All variables had global scope
- Nightmare to debug and modify



From: Weird Computer Games, Usborne Books
https://usborne.com/browse-books/features/computer-and-coding-books/

Go To Considered Harmful

MONASH University

- In 1968, Dijkstra published one of the first “software engineering” articles. It is now considered a classic

Go To Statement Considered Harmful

Key Words and Phrases: go to statement, jump instruction, branch instruction, conditional clause, alternative clause, repetitive clause, program intelligibility, program sequencing
CR Categories: 4.22, 5.23, 5.24

EDITOR:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such disastrous effects, and I became convinced that the **go to** statement should be abolished from all “higher level” programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

Dijkstra, E. (March 1968).“Go To Statement Considered Harmful”, Communications of the ACM 11 (3): 147–148. https://doi-org.ezproxy.lib.monash.edu.au/10.1145/362929.362947

Procedural programming

MONASH University

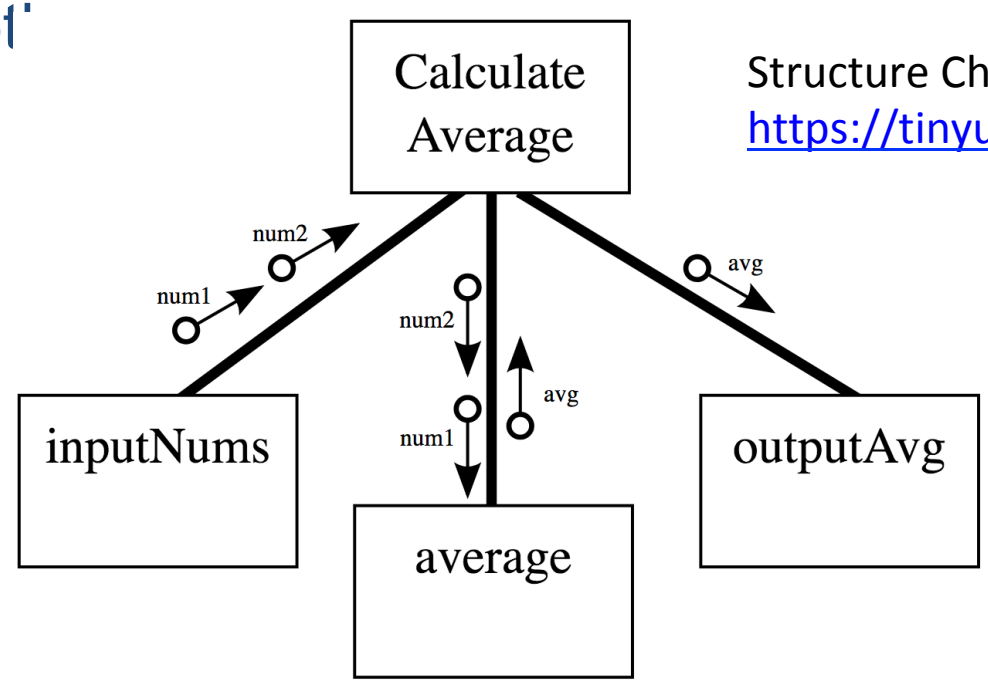
- Programs are collections of procedures
  - Each procedure is a little mini-program
- Each procedure has an *interface* (inputs and return values).
  - Can in theory change implementation without changing interface
- May support defining data types
- Makes it much easier to write/ test/debug/extend programs than spaghetti code
- Key abstraction is the *action*

```
    VAR
        Day: DayType;               (* Scanner. *)
        Found: boolean;             (* Tell if a match was found. *)
    BEGIN
        Found := FALSE;
        Day := Sun;
        WHILE (Day < BadDay) AND NOT Found DO
            BEGIN
                IF DayMap[Day] = S THEN
                    Found := TRUE
                ELSE
                    Day := succ(Day)
            END;

        MapToDay := Day
    END;

{ Read one character, but do not read past the end of line.  Just
  return a space.
    Pre: InFile is open for reading.
    Post: If InFile was at eoln, Ch is set to ' ', and InFile is
          unchanged.  Otherwise, one character is read from InFile to Ch. }
PROCEDURE ReadOnLine(VAR InFile: TEXT; VAR Ch: Char);
    BEGIN
        IF eoln(InFile) THEN
            Ch := ' '
        ELSE
            read(InFile, Ch)
    END;
```

Pascal program by Thomas Bennett
http://sandbox.mc.edu/~bennet/cs404/doc/pasdex.html

MONASH University

- Programs consist of procedures (actions) that pass data to each of

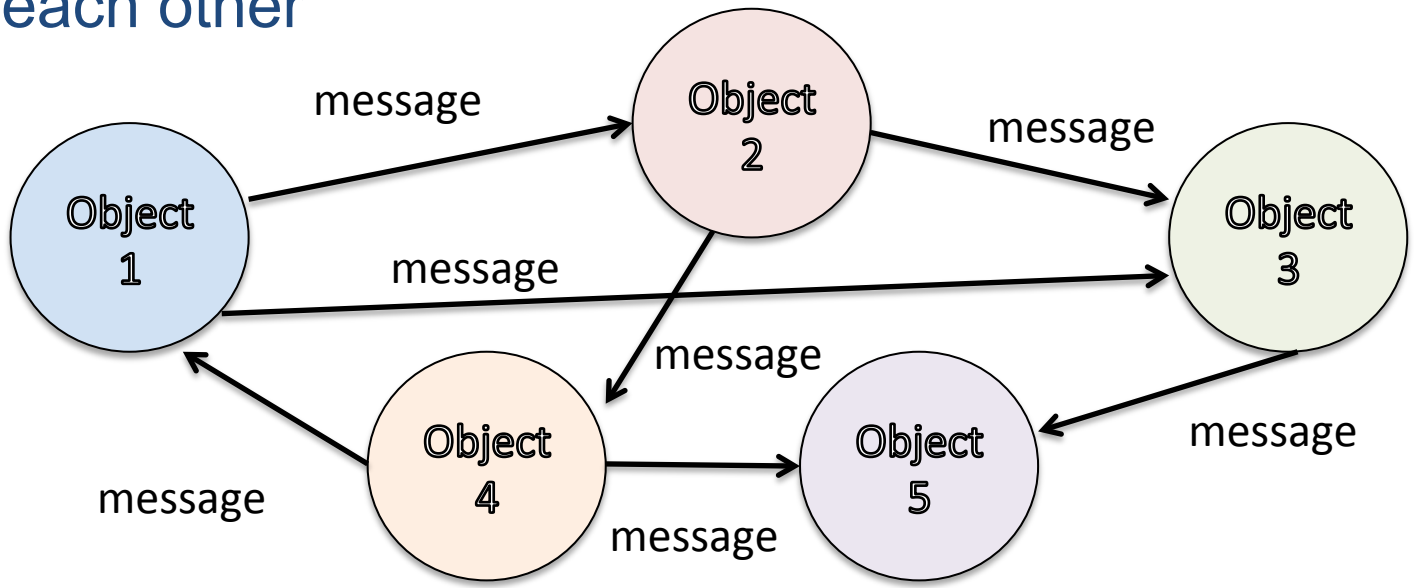


Structure Chart Example
https://tinyurl.com/y7ysyhdk

- Procedures were a big improvement on spaghetti code, BUT…
  - Need a larger-scale abstraction for big systems
  - Procedural programming makes the *action* the primary unit of organisation – *data* is secondary.

MONASH University

- Object-oriented programming flips this around
  - Unit of organization is the *object*
    - Objects are instances of *classes*: a type of thing (an abstract *data* type)
    - A class defines an interface: the set of *messages* that its objects can receive and what it promises to do in response (*actions*).
- Programs consist of objects (data) that send messages to each other



Why might Object-Orientation be better?

MONASH University

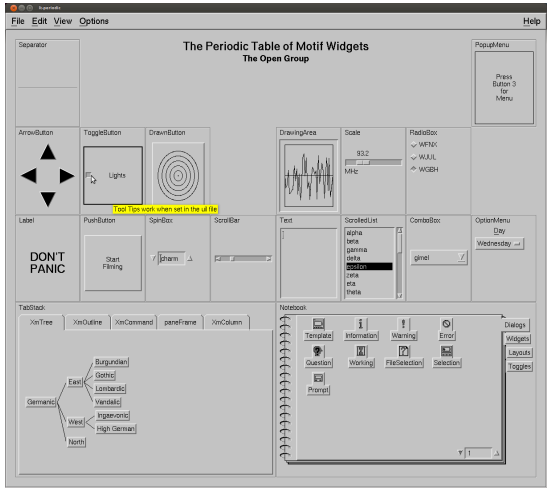
- A (perhaps "The") key design principle:
  - *Reduce Dependencies* As Much As Possible
  - You will hear it again and again in this unit and others
- The corollary is:

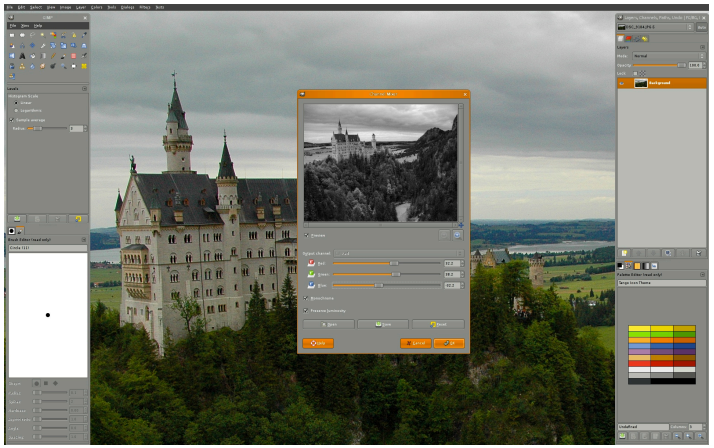
  Group Things That *Must* Depend On Each Other Together (inside an encapsulation boundary – more on that later)
  - You will hear this one a lot too ☺
- Other the actions that access or modify are certain type of data *must* depend on that data, and on each other
  - It thus makes sense to group them together in a class
  - Many changes will thus be limited to a single class

Object-oriented languages

MONASH University

- Object-orientation is a philosophy of program design more than anything
- You can implement an object-oriented design in any language
- Much easier to use a language that directly supports OO concepts
- Many languages claim to be “OO”, but OO features supported differ





Notable OO languages

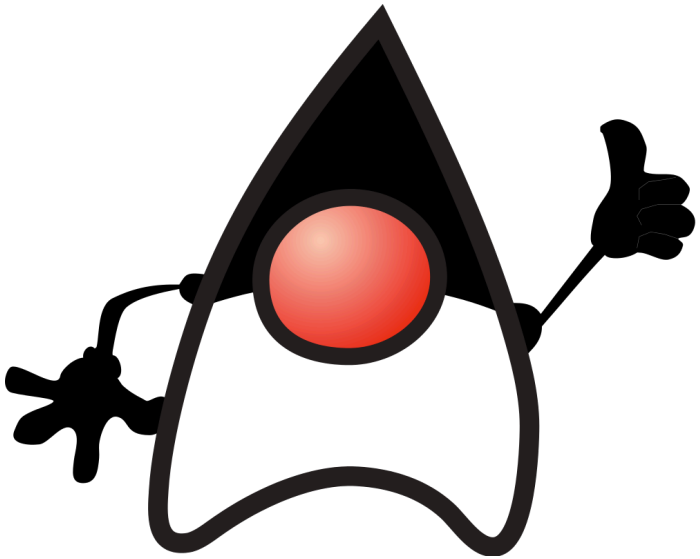
MONASH University

- The grandparent:
  - SIMULA 67
- Parents:
  - Smalltalk
  - Eiffel
- Hitting the big time:
  - C++
  - Java
  - Objective-C
  - Python
  - C#

Java

MONASH University

- Developed by James Gosling at Sun Microsystems.
  - Originally designed for Internet of Things.
    - Applets
- Inspired by C++, Smalltalk, Eiffel and other languages.
- Design goals:
- Object-oriented
  - Familiar for C/C++ programmers.
  - Robust
  - Secure
  - Fast
  - Portable
  - Threadable



Duke, the Java mascot
https://commons.wikimedia.org/wiki/File:ThumbsUp.svg

Why Java?

MONASH University

- As a teaching language:
  - Fairly pure OO language
    - Also keywords match very well with the key OO concepts
      - `extends`, `implements`, `public`, `private`, etc.
  - Strong, static typing
  - Memory safety (garbage collection)
  - Relatively clean design (low “cruft” levels)
  - Widely used in industry (see tables in last lecture)
  - Free tools available on all major platforms (MS Windows, MacOS, Linux,…)
- As an industry language:
  - Widely used for enterprise systems
  - Supports large systems well
  - High importance of reliability, maintainability, security
- Default language for Android development

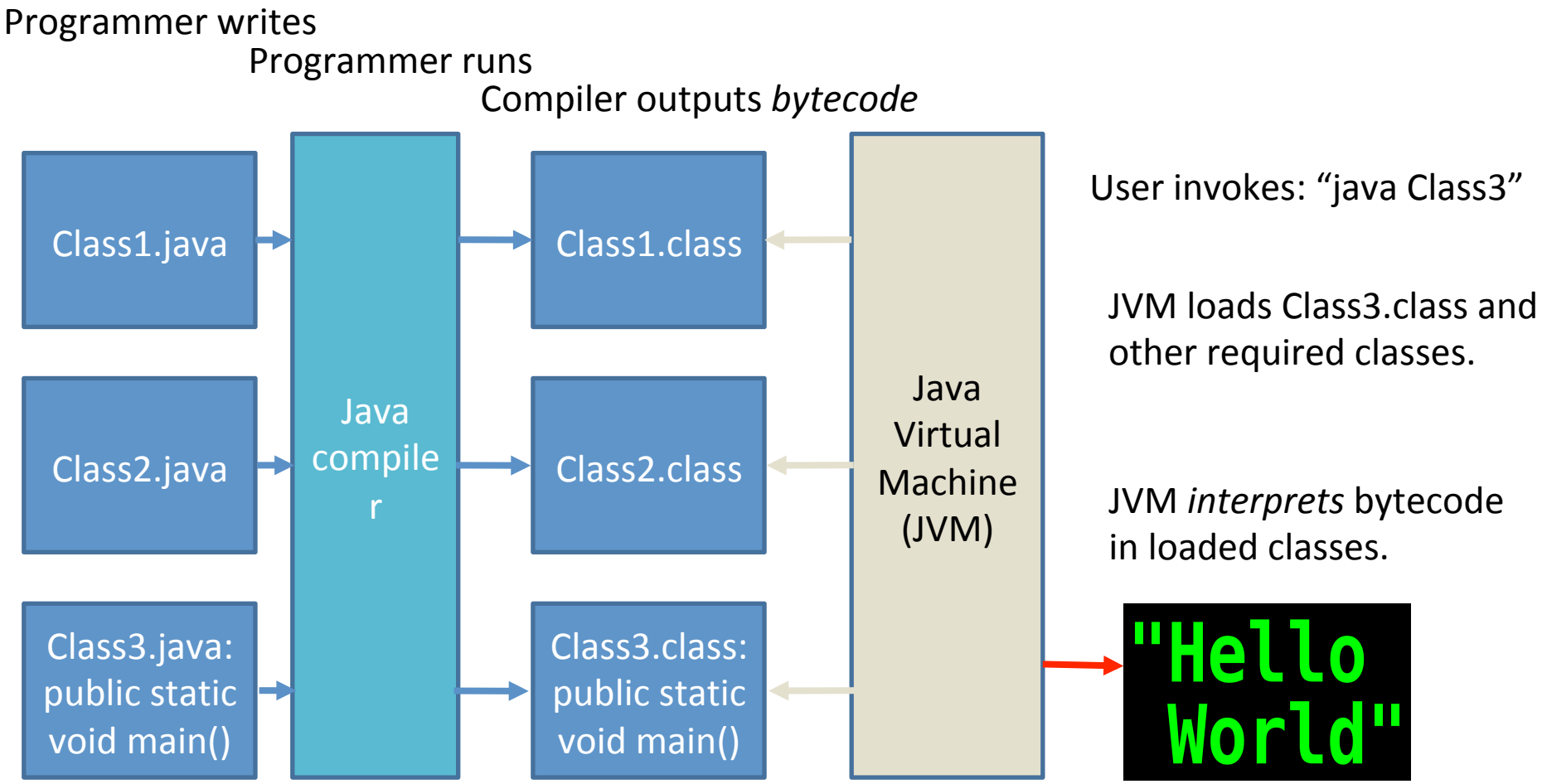
Why not Java?

MONASH University

- Verbose
  - Partly inherent to being an industrial-scale OO language
  - Partly due to design
  - Tedious to write without IDE support
- Harder to use platform-specific toolkits (particularly UI)
  - If you want to target Windows, C# is easier
  - If you want to target MacOS, Objective-C or Swift are easier
  - If you want to target browser front-ends, Javascript is easier
- Supplied class libraries not newbie-friendly
- Slower than C/C++ in some circumstances
- Not for systems programming/bit twiddling

Digression: How your Java program runs

MONASH University



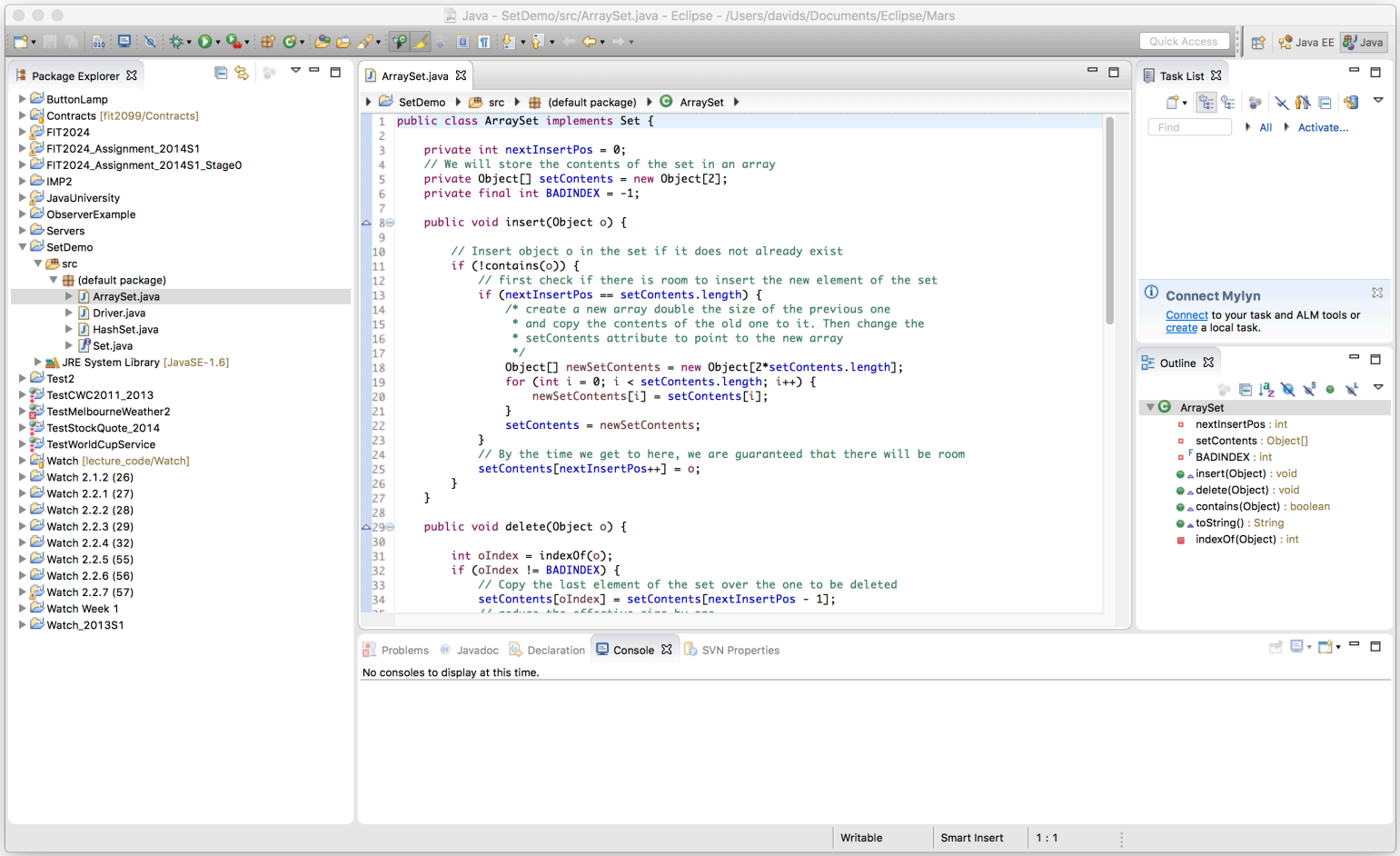
Working with Java: the Eclipse IDE

MONASH University

- It's possible to
  - write Java programs in a text editor
  - compile by running the compiler at the command line
  - run by invoking the JVM
- This can be tedious
- Integrated Development Environments (IDEs) take most of the tedium out
- Several IDEs for Java in wide use:
  - Eclipse https://www.eclipse.org/ide/
  - Netbeans https://netbeans.org/
  - IntelliJ IDEA https://www.jetbrains.com/idea/
- You can use any tools you like for FIT2099
- We will be using and supporting the Eclipse IDE

Using Eclipse: Hello World!

MONASH University



Where to next?

MONASH University

- Readings: There are **set readings** every week **that you must read**. They prepare you for the lectures and labs, and are **examinable**
  - see Moodle
- Lectures: Learn fundamental OO Concepts, Design Principles, and UML Basics via code-along with FIT2099 teaching staff
- Labs: Put OO Concepts and Design Principles from lectures into practice in Java, after sketching design in UML
  - Week 1 unmarked
  - Week 2-6 **marked**