COMP122 Week 1

Module Introduction

Dr. Patrick Totzke
totzke@liverpool.ac.uk

https://liverpool.instructure.com/courses/59716

So what is object-oriented programming?

Historical Context – Imperative Programming Paradigms

- Universal Machines! (1940s –)
  - $\rightsquigarrow$ COMP124 (systems)
- High-level Programming languages
  - Compilers / Interpreters
  - Imperative / Functional / Logic
- Structured Programming (late 50s)
  - Code blocks (if-then-else; for-loops; etc.)
  - Dijkstra's – *go to statement considered harmful*
  - $\rightsquigarrow$ COMP109 (TCS, correctness proofs...)
- Procedural Programming (60s –)
  - User-definable procedures (=routines=functions)
- Modular Programming (late 60s and 70s)
  - information hiding and separation of concerns
- Object-Oriented Programming (80s –)
  - SIMULA, Smalltalk,...

Object-Oriented Programming

is based on the idea of interacting objects which contain both data and procedures and are an instances of a whole “class” of similar objects.

- *Encapsulation:* grouping data and code that acts on it into a single unit.
- *Abstraction:* hiding implementation details from users.
- *Inheritance:* using known classes of objects as blueprints for more specific ones.
- *Polymorphism:* different behaviour of subclasses by re-defining methods.

An Example of OO Modelling

Suppose we're building some application that deals with people's names, addresses, student IDs (for students), university position and salary (for lecturers), etc.

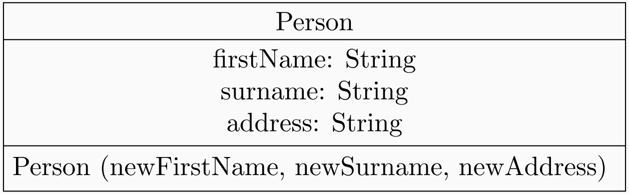
Firstly, a student or lecturer is a person, so let us first think of creating a `Person` class.

Each individual will have associated data such as their first and surname, and address, but we abstract from the remaining personal details.

An Example of OO Modelling (cont.)

This is an Class Diagram

A `Person` class may be represented in the following form.



Each individual will be an object (also an *instance*) of this class.

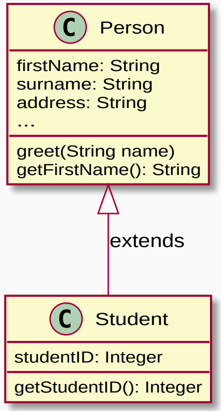
An Example of OO Modelling – Inheritance

A student is a person with *additional* associated data, such as a student ID.

We *extend* the `Person` object to add new attributes and behaviour to create a new subclass `Student`. This is inheritance.

The idea is that we have a well-defined (and well-tested) class, and can extend it to add more attributes and methods, *without modifying an already existing class*, and without modifying the code in the `Person` class.

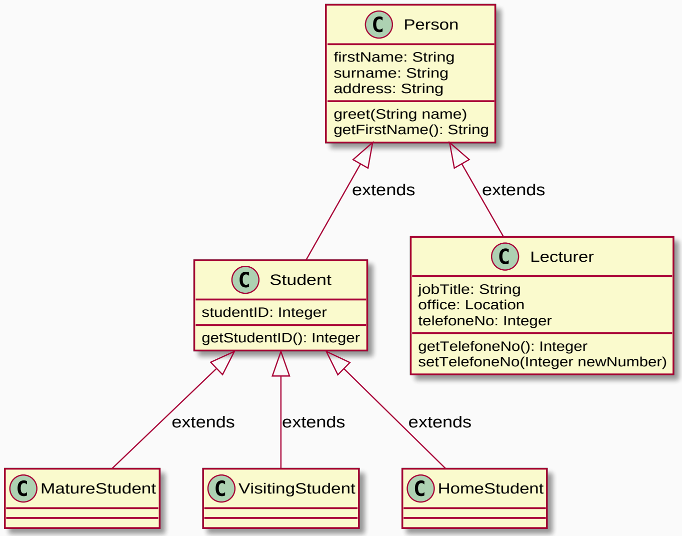
An Example of OO Modelling – Inheritance



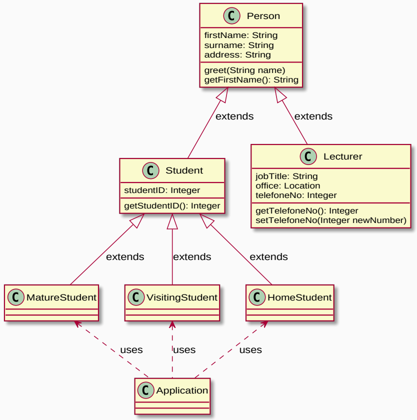
Each Student is a Person (but not vice-versa). Each `Student` will have all of the attributes and methods of a `Person`, with additional ones

Similarly, we can *extend* the `Person` to create an `Lecturer` class by adding new attributes such as `office` and `telefoneNo`, and new methods to access/change these attributes.

An Example of OO Modelling – Inheritance



An Example of OO Modelling – Polymorphism



Imagine an application that creates many *instances* of type `HomeStudent`, `VisitingStudent`, and `MatureStudent` and stores them in a list of `Student`.

It can then use a `for` loop to greet on each member without knowing the specific class.

```
for student in studentList:
    name = student.getName()
    student.greet(name)
```

This is an example of Polymorphism.

Module Overview

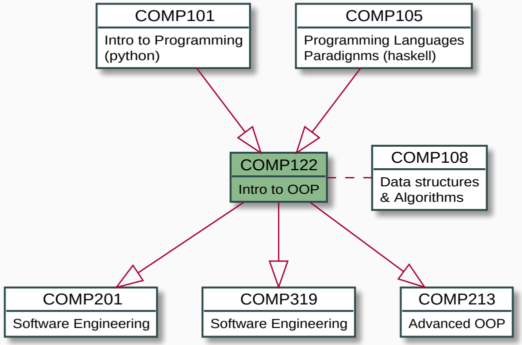
COMP122 presents a conceptual and practical introduction to object oriented programming, exemplified by the high-level programming language *Java*.

Learning Outcomes

- Class Hierarchies
- Polymorphism
- Design Patterns
- Unit Testing
- Event-driven GUIs

- Java Programming
- Version Control with `git`

Relations to other modules



Schedule

Weeks 1-4: Imperative programming with Java

- Variables & Types, Loops, Methods
- Assignment due 24/02

Weeks 5-7: Object-Oriented Programming

- Objects, Classes, Inheritance, Polymorphism, Interfaces
- Assignment due 24/03

Weeks 8-11: OOP Design Patterns

- Collections, Iterators, Streams
- Assignment due 05/05

Your Week in COMP122

Activities

1. **Self-study** ($\geq$ 1h)
   Watch videos, read stuff, take a quiz, ask questions.
2. **Lectures** ($2 \times$ 1h)
   one presenting new material,
   one for Q&A, demos, assignment prep
3. **Programming Exercises** (2h)
   1-3 exercises per week
   at home or in supervised labs.
   submit by Friday 5pm

Programming Labs

... allow you to practice with Java programming in a supervised environment.

- You have been allocated a lab slot.
- Attendance at lab sessions is optional but recommended.
- Labs start in week 2. Check your timetable!

Assessment

Your module grade is determined by

75%	take-home assignments (3x25%)
10%	lab exercises
15%	final Canvas quiz

Note that

- All assignments are individual pieces of work.
- Submissions are electronically on Canvas
- University policy for late submissions apply.
- Individual feedback will be released on Canvas
- The passing grade is 40%. There will be a resit assignment after the term.

Former Students on COMP122

“Assessments where to long, and difficult at first.”

“Assignments were hard but actually taught me how to program.”

“It’s a hard module. I struggled a lot with the assignments. I’d liken it to being thrown into cold water and being told how to swim. It’s a real struggle to do two java assignments at the same time without really knowing the language.”

How to Succeed?

Keep Calm and Code On!

Remember that “success” means gaining $\geq 40\%$. Even if you do not get there, there will be a resit, and no Y1 grade counts towards your degree grade. You’ve got this! Also,

- Keep up with the material and attend the labs.
- Seek help if you are stuck! Ask your peers or TAs or post on Canvas.
- Take your time for courseworks; don’t “hack it together” just before the deadline. Carefully consider the stated requirements and submission guidelines.
- Take advantage of automarker feedback!

How to get help

Topical Questions

Ask during labs or lectures, or on the Canvas discussion board.

Administrative issues

Extenuating circumstances need to be reported to the student office (`csstudy@liverpool.ac.uk`) in case you cannot submit assignments on time. Also contact them regarding timetabling issues, resits etc.

My Email/Office hour

For personal queries you can contact me by email (`totzke@liverpool.ac.uk`) or drop in to my office on Tuesdays from 1-12am.

TODO Now

1. Check your timetable for your lab slot
2. Find the Canvas Course page
3. Start reading the materials for weeks 1&2
4. Set up a convenient programming environment

Your Questions?

Schedule for Thursday

1. Java Compiler/ Interpreter
2. Hello.java
3. GitHub.com
4. CS50 labs / IDE
5. Coding on Windows
6. CodeGrade

Your Questions?

Q1

Q: I need help *now!* Can I contact you on Teams?

A: Please don't.

How to get help

Topical Questions

Ask during labs or lectures, or on the Canvas discussion board.
There is also a teams channel for live chat among yourselves.

Administrative issues

Extensions need to be requested in the student office (`csstudy@liverpool.ac.uk`).
Also contact them regarding timetabling issues, resits etc.

My Email/Office hour

For personal queries you can contact me by email (`totzke@liverpool.ac.uk`) or drop in to my office on Mondays, 10-11am.

Q2

Q: Can I swap my assigned lap slot?

A: No (for now). Find s/o to swap with?

Q3

Q: Is procedurally generated, or AI assisted code considered a threat to academic integrity? Or should a CS student be encouraged to use all modern tools available to develop solutions to problems?

A: yes

Q4

Q: Why do we not use Visual Studio Code?

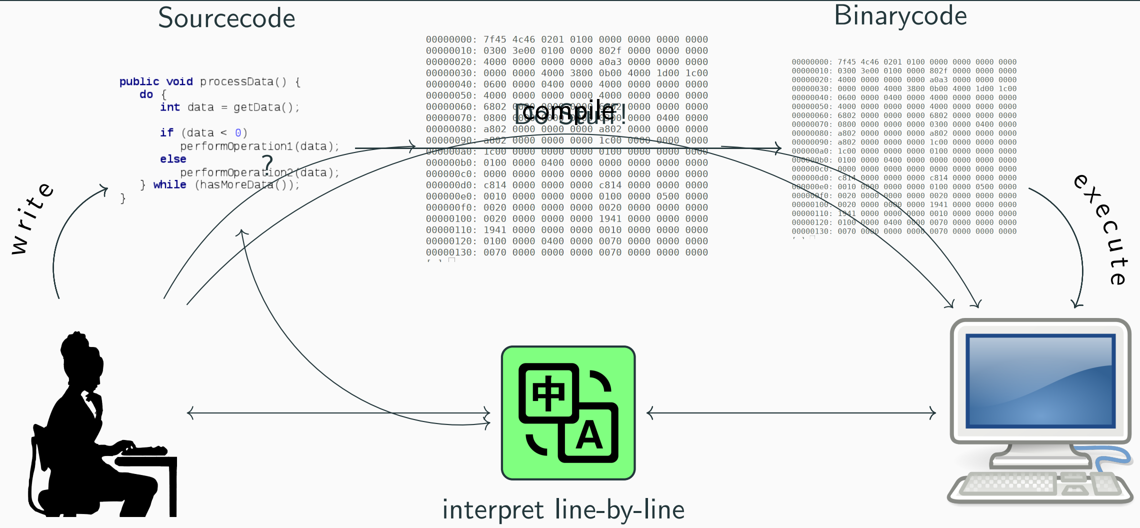
What is is the best IDE?

What do you use?

A: Use whichever text editor you like best.

Do shop around! I use vim.

Compilers vs Interpreters



```
public void processData() {
    do {
        int data = getData();

        if (data < 0)
            performOperation1(data);
        else
            performOperation2(data);
    } while (hasMoreData());
}
```

```
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000
00000010: 0300 3e00 0100 0000 802f 0000 0000 0000
00000020: 4000 0000 0000 0000 a0a3 0000 0000 0000
00000030: 0000 0000 4000 3800 0b00 4000 1d00 1c00
00000040: 0600 0000 0400 0000 4000 0000 0000 0000
00000050: 4000 0000 0000 0000 4000 0000 0000 0000
00000060: 6802 0000 0000 0000 6802 0000 0000 0000
00000070: 0800 0000 0000 0000 0300 0000 0400 0000
00000080: a802 0000 0000 0000 a802 0000 0000 0000
00000090: a802 0000 0000 0000 1c00 0000 0000 0000
000000a0: 1c00 0000 0000 0000 0100 0000 0000 0000
000000b0: 0100 0000 0400 0000 0000 0000 0000 0000
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000
000000d0: c814 0000 0000 0000 c814 0000 0000 0000
000000e0: 0010 0000 0000 0000 0100 0000 0500 0000
000000f0: 0020 0000 0000 0000 0020 0000 0000 0000
00000100: 0020 0000 0000 0000 1941 0000 0000 0000
00000110: 1941 0000 0000 0000 0010 0000 0000 0000
00000120: 0100 0000 0400 0000 0070 0000 0000 0000
00000130: 0070 0000 0000 0000 0070 0000 0000 0000
```

```
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000
00000010: 0300 3e00 0100 0000 802f 0000 0000 0000
00000020: 4000 0000 0000 0000 a0a3 0000 0000 0000
00000030: 0000 0000 4000 3800 0b00 4000 1d00 1c00
00000040: 0600 0000 0400 0000 4000 0000 0000 0000
00000050: 4000 0000 0000 0000 4000 0000 0000 0000
00000060: 6802 0000 0000 0000 6802 0000 0000 0000
00000070: 0800 0000 0000 0000 0300 0000 0400 0000
00000080: a802 0000 0000 0000 a802 0000 0000 0000
00000090: a802 0000 0000 0000 1c00 0000 0000 0000
000000a0: 1c00 0000 0000 0000 0100 0000 0000 0000
000000b0: 0100 0000 0400 0000 0000 0000 0000 0000
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000
000000d0: c814 0000 0000 0000 c814 0000 0000 0000
000000e0: 0010 0000 0000 0000 0100 0000 0500 0000
000000f0: 0020 0000 0000 0000 0020 0000 0000 0000
00000100: 0020 0000 0000 0000 1941 0000 0000 0000
00000110: 1941 0000 0000 0000 0010 0000 0000 0000
00000120: 0100 0000 0400 0000 0070 0000 0000 0000
00000130: 0070 0000 0000 0000 0070 0000 0000 0000
```

Compilers vs Interpreters

Compiler

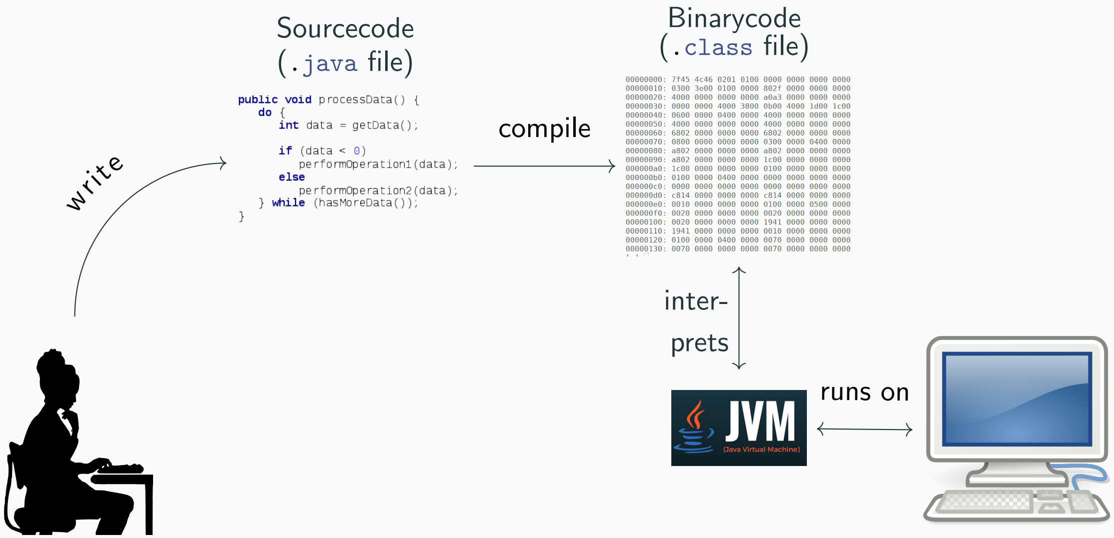
- translates sourcecode into executable binary code all at once
- full code analysis and optimization before code is executed.
- compiled binary code is specific to an architecture and OS.
- Examples: Cobol, Fortran, C, Haskell

Interpreters

- translates and executes sourcecode one command at a time
- slower and no static code analysis before execution.
- sourcecode is runs directly on any system with an interpreter.
- Examples: Perl, Bash, Javascript

Java is a hybrid between the two.

Running your Java code



```
public void processData() {
    do {
        int data = getData();

        if (data < 0)
            performOperation1(data);
        else
            performOperation2(data);
    } while (hasMoreData());
}
```

```
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000
00000010: 0300 3e00 0100 0000 802f 0000 0000 0000
00000020: 4000 0000 0000 0000 a0a3 0000 0000 0000
00000030: 0000 0000 4000 3800 0b00 4000 1d00 1c00
00000040: 0600 0000 0400 0000 4000 0000 0000 0000
00000050: 4000 0000 0000 0000 4000 0000 0000 0000
00000060: 6802 0000 0000 0000 6802 0000 0000 0000
00000070: 0800 0000 0000 0000 0300 0000 0400 0000
00000080: a802 0000 0000 0000 a802 0000 0000 0000
00000090: a802 0000 0000 0000 1c00 0000 0000 0000
000000a0: 1c00 0000 0000 0000 0100 0000 0000 0000
000000b0: 0100 0000 0400 0000 0000 0000 0000 0000
000000c0: 0000 0000 0000 0000 0000 0000 0000 0000
000000d0: c814 0000 0000 0000 c814 0000 0000 0000
000000e0: 0010 0000 0000 0000 0100 0000 0500 0000
000000f0: 0020 0000 0000 0000 0020 0000 0000 0000
00000100: 0020 0000 0000 0000 1941 0000 0000 0000
00000110: 1941 0000 0000 0000 0010 0000 0000 0000
00000120: 0100 0000 0400 0000 0070 0000 0000 0000
00000130: 0070 0000 0000 0000 0070 0000 0000 0000
```

Running your Java code

1. Compile a sourcecode file `Hello.java` with `javac Hello.java`. This will create a bytecode file `Hello.class`.
2. Start the JVM and run `Hello.class` with `java Hello`.

Example: Hello.java

```java
/**
 * Author: Patrick Totzke
 * The HelloWorld class implements an application that
 * prints out "Hello World".
 */
public class HelloWorld {
    // ---------------METHODS-------------
    /*  Main Method */
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```java
/**
 * Author: Patrick Totzke
 * The HelloWorld class implements an application that
 * prints out "Hello World".
 */
public class HelloWorld {
    // ---------------METHODS-------------
```

Schedule for Thursday

1. Java Compiler/ Interpreter
2. Hello.java
3. GitHub.com
4. CS50 labs / IDE
5. Coding on Windows
6. CodeGrade

Summary of Week 1

We looked at. . .

- OOP modeling (motivation)
- Module organization
- how to compile and run a “Hello World” Java program

Next Week

- Java data types and control flow
- typical syntax errors.