Féidearthachtaí as Cuimse
Infinite Possibilities

Relationships between classes

Object Oriented programming

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

So far.. classes have interacted in various ways with each other

- Classes can import each other
  - `import javax.swing.jframe`

    `i.e.` One class “uses” another class
- Classes can inherit from another class
  - “is a type of” “is-a”
- Classes can implement Interfaces

Importing a class

- Needed when you want to USE another class that is not in the same package.

- Means you don't have to put in the fully qualified name in.

  - `e.g. import java.util.scanner`

```
Scanner myInput = new Scanner(..)
```

(And not java.util.Scanner = new Scanner(..);

Extending (inheriting) a class

- Subclass inherits behaviour and attributes

```
public class HourslyEmployee extends Employee
```

etc

Implementing an interface

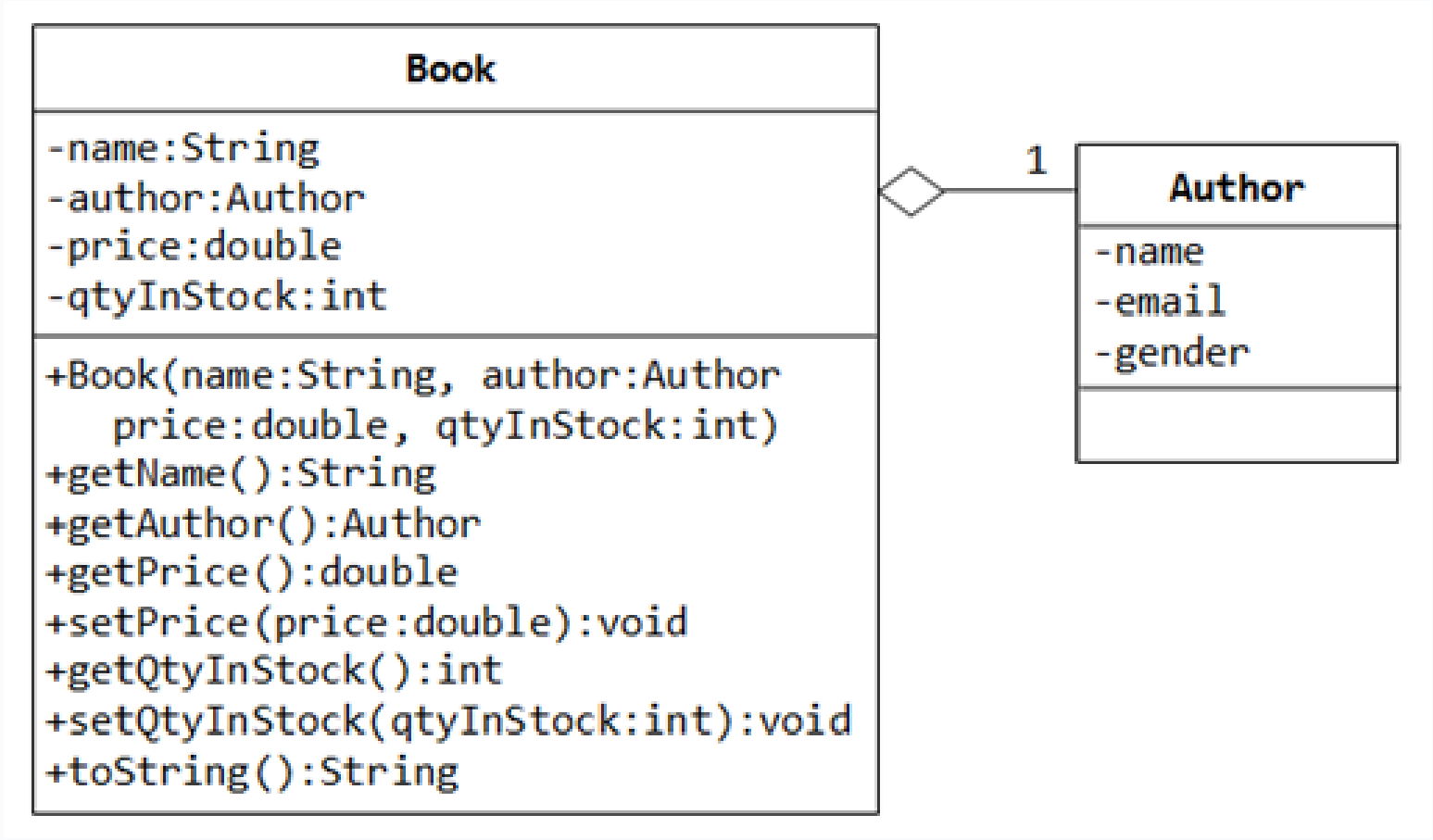
- Get a class to sign up to implementing particular behaviour (i.e. a set of methods in the interface)

- `class Employee implements Payable`

Classes have interacted in various ways with each other

- Classes can import functionality
  - `import javax.swing.jframe`
    "uses"
- Classes can inherit from another class
  - "is a type of" "is-a"
- Classes can implement Interfaces
- ***Classes can have references to other objects as members ("has-a")*** **= *Composition***

Classes can be composed of other classes (composition)



An important part of OO

- You’re tasked with developing an application
  - How should I divide up my classes?
  - What classes should I create?
  - Not a cut and dried process !
  - General process will be explained

Avoid just creating big long rambling class for your application!

How should I divide up my classes?”

This means:

What **parts** of the problem should become separate classes?

What **responsibility** does each class have?

How do they **talk to each other**?

You want:

- **High cohesion** – each class does one main job
- **Low coupling** – classes don’t depend too heavily on lots of other classes

What classes should I create?

Very simple starting rules:

Look for nouns in the problem description

“Book”, “Member”, “Loan”, “Account”, “Order”, “Product”, “Payment”

These often become classes.

Give each class clear responsibilities

Book → title, author, ISBN

Member → name, ID, list of borrowed books

Library → stores collections of books and members, handles borrowing/returning

Separate ‘things’ and ‘controllers’

“Things” (domain objects): Book, Student, Product

“Managers/Controllers”: Library, OrderManager, GameController

Don’t put everything in one class

If a class is doing too much, split it.

Not a cut and dried process!

There is **no single correct answer**.

Different programmers might design **slightly different class sets**.

You often:

- Start with a simple design
- Code a bit
- Realise a class is too big / badly named
- **Refactor** (split, rename, move methods)

Design improves **iteratively**, not in one perfect step.

General process

- Read the problem, underline nouns and verbs.
- Turn main nouns into candidate classes.
- Decide attributes and methods for each class.
- Draw a simple UML diagram (classes + relationships).
- Check: responsibilities clear? any class doing too much?
- Adjust and refine as you go while coding.