Noroff
School of technology
and digital media

Thursday - Week 1

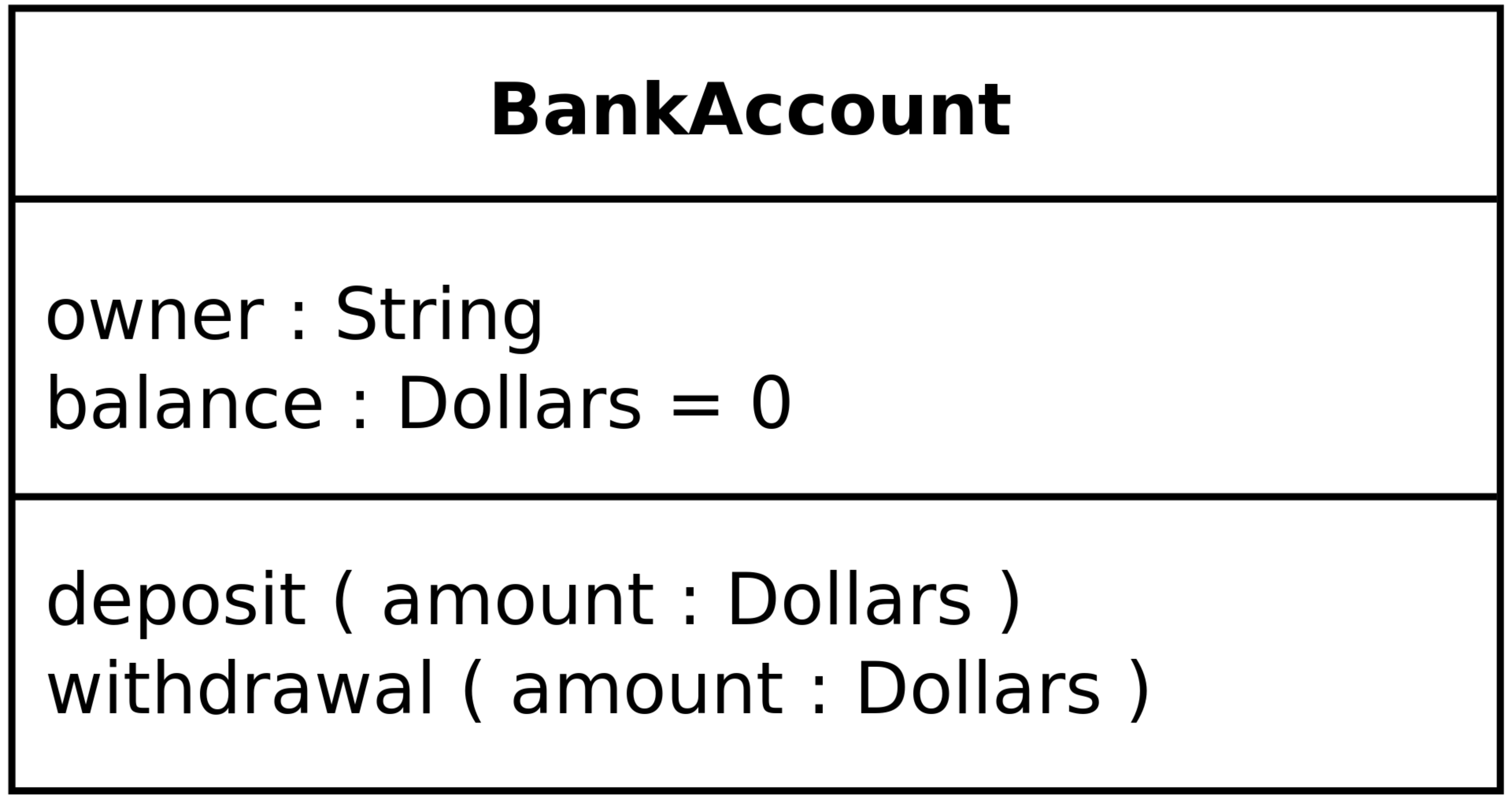
Object Orientation (UML)

Unified Modeling Language

- Gives us a standardised way of describing software
- A variety of different document (diagram) type

Class Diagram

- Three parts:
  - Name
  - Attributes (variables)
  - Methods



BankAccount

owner : String
balance : Dollars = 0

deposit (amount : Dollars)
withdrawal (amount : Dollars)

Access Modifiers in UML

+ Public

# Protected

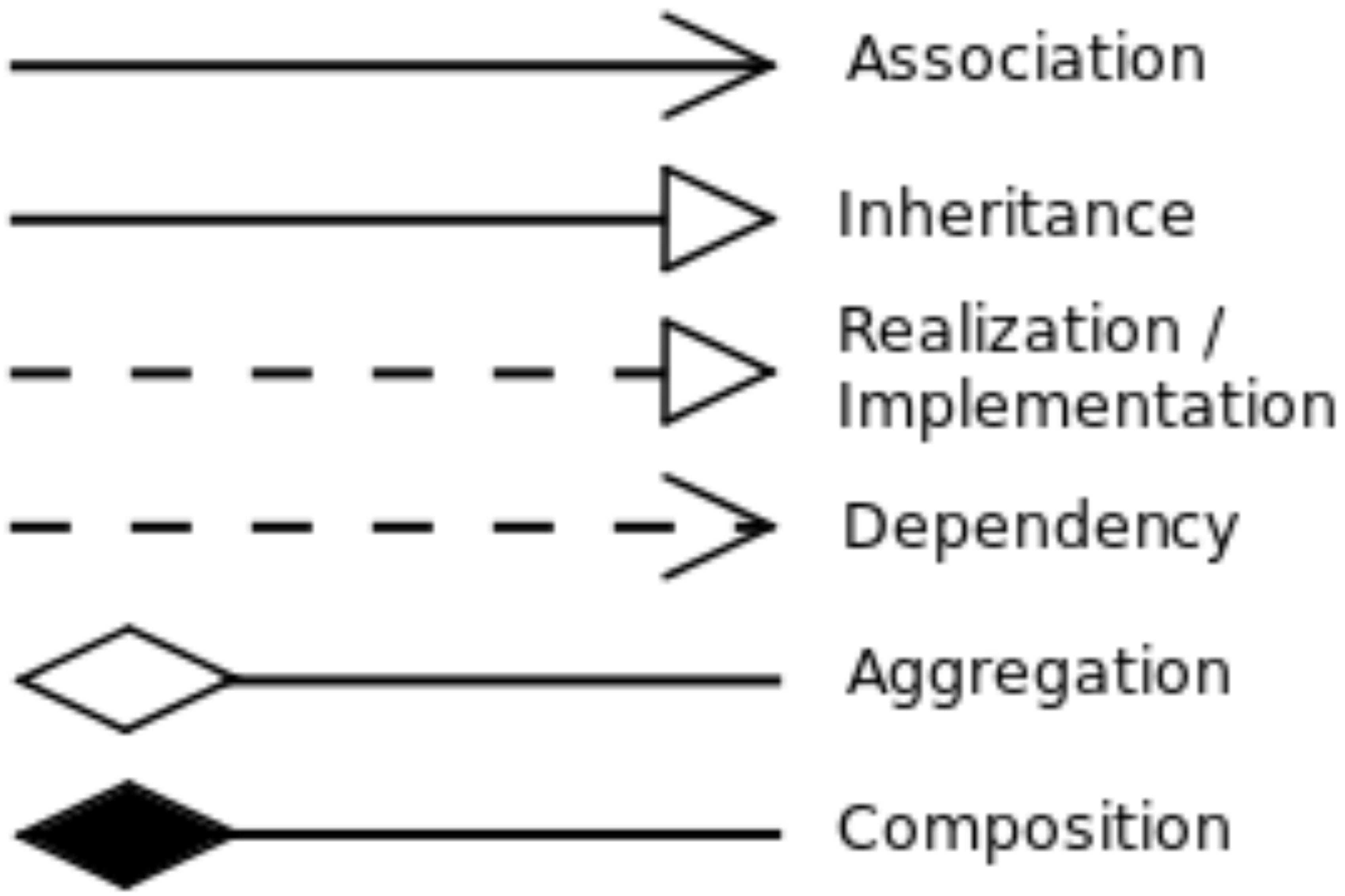
- Private

+ main()

- somePrivateMethod(int num)

Relationships

- We will get to what these mean in a moment…
- The “inheritance” is going to be the mostly commonly seen
- Then “association”



Inheritance

- Inheritance is when a new object inherits some of the code from another object so you have less code to write overall.

- The new object is a sub class (child) of the super class (parent)

Aggregation vs Composition

- Aggregation
  - The one class “has-a” instance of the other class
  - This other can exist without this relationship
- Composition
  - The one class “has-a” instance of the other class
  - The other class CANNOT exist without this relationship

Aggregation

- A car has wheels.

- You can have a wheel without it being attached to a car
- If you remove a wheel, the car does not function

Composition

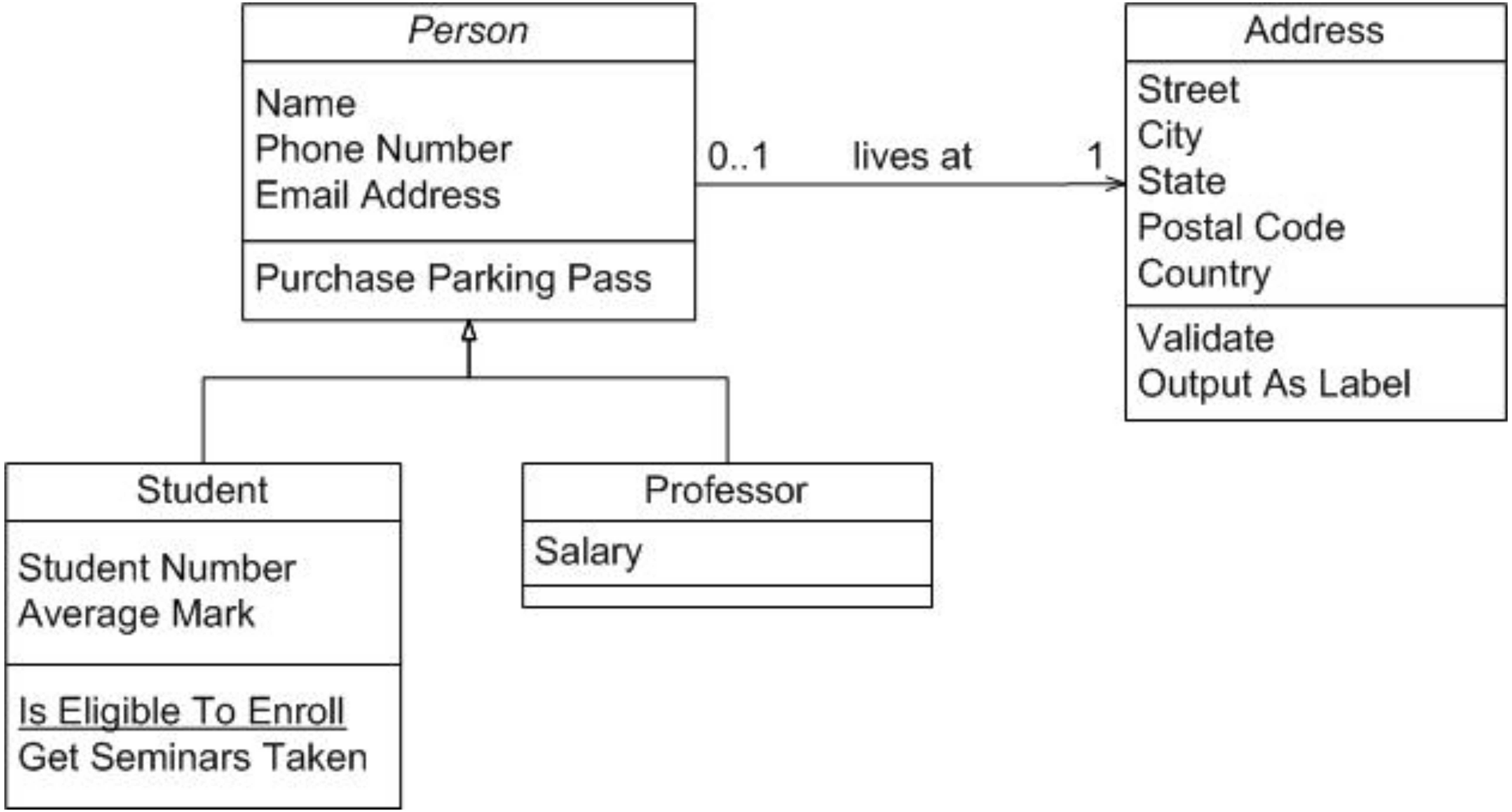
- A person has arms.

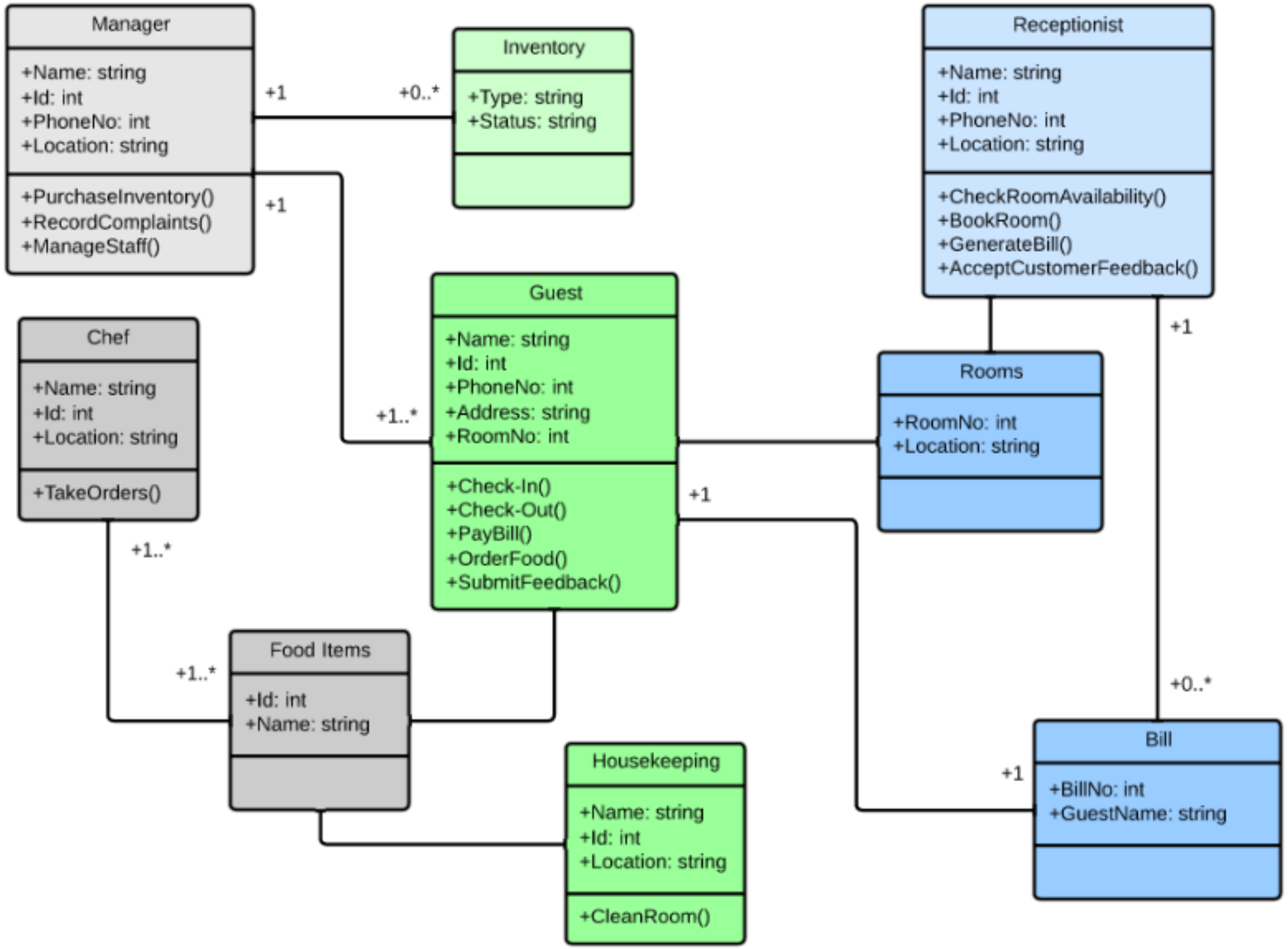
- An arm that is not attached to a person is arguably useless
- The person doesn’t work so well if you remove an arm either

Association

- Sometime two classes just need to communicate

- I have a sweater.
- I can take it off, I’m still me and my sweater is still just as useful


Person
Name
Phone Number
Email Address
Purchase Parking Pass
0..1
lives at
1
Address
Street
City
State
Postal Code
Country
Validate
Output As Label
Student
Student Number
Average Mark
Is Eligible To Enroll
Get Seminars Taken
Professor
Salary



Manager

+Name: string
+Id: int
+PhoneNo: int
+Location: string

+PurchaseInventory()
+RecordComplaints()
+ManageStaff()

Inventory

+Type: string
+Status: string

Receptionist

+Name: string
+Id: int
+PhoneNo: int
+Location: string

+CheckRoomAvailability()
+BookRoom()
+GenerateBill()
+AcceptCustomerFeedback()

Chef

+Name: string
+Id: int
+Location: string

+TakeOrders()

Guest

+Name: string
+Id: int
+PhoneNo: int
+Address: string
+RoomNo: int

+Check-In()
+Check-Out()
+PayBill()
+OrderFood()
+SubmitFeedback()

Rooms

+RoomNo: int
+Location: string

Food Items

+Id: int
+Name: string

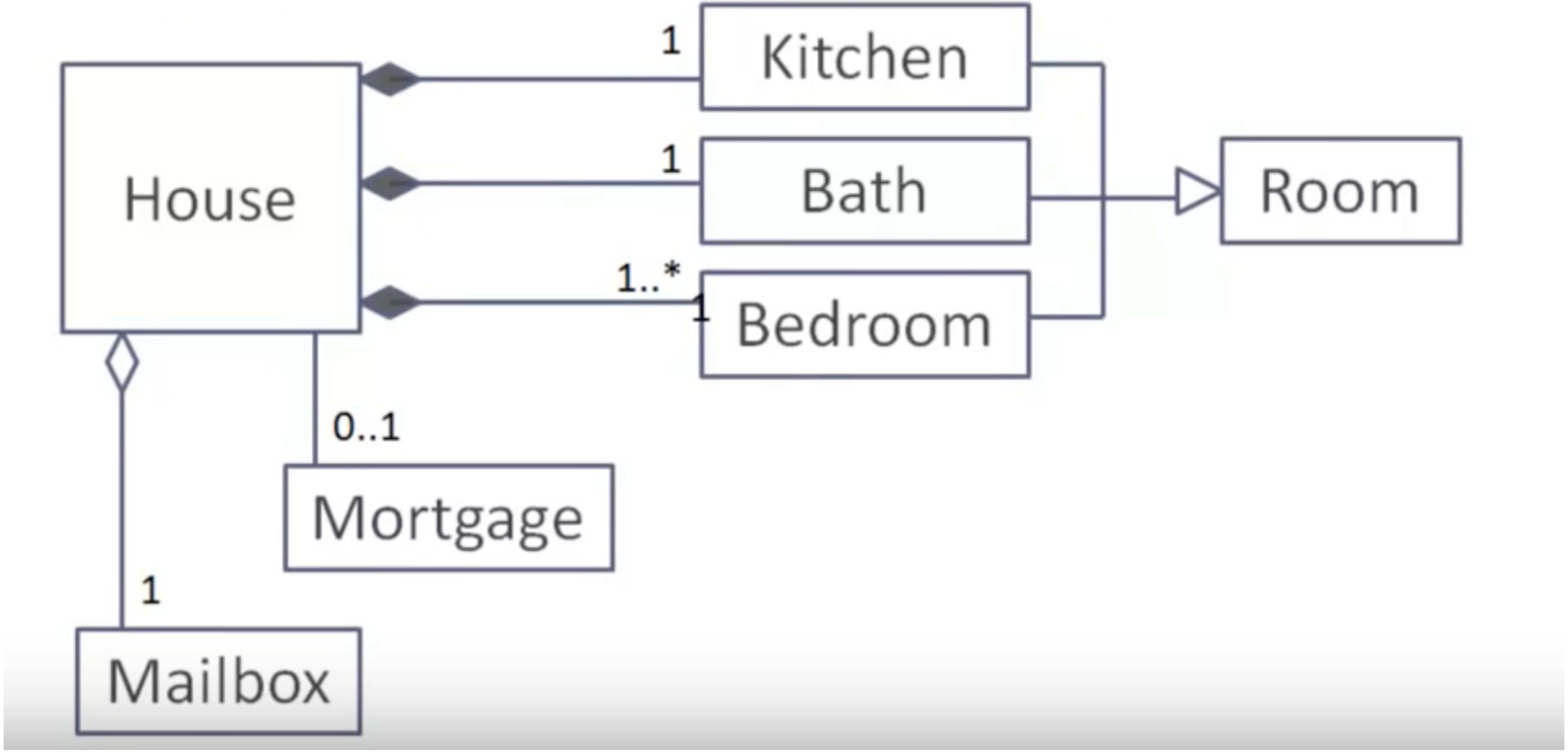
Housekeeping

+Name: string
+Id: int
+Location: string

+CleanRoom()

Bill

+BillNo: int
+GuestName: string


House
Kitchen
Bath
Bedroom
Room
Mortgage
Mailbox
1
1
1..*
1
0..1
1

Noroff
School of technology
and digital media

Thursday - Week 1

Inheritance

Inheritance

- Inheritance means that that one class inherits all of the properties of another
- This lets us model systems in a logical manner
- It also ensures that out code is more flexible

Inheritance in Java

- We use the *extends* keyword:

```
public childClass extends parentClass{
//class code
}
```

- In java the child class is called the ***subclass***
- And the parent class is called the ***superclass***

Constructors

- Because the child is still the ‘same’ as it’s parent when we create the child we may need to call the parent’s constructor
- We do this using `super()`
- We can choose when this happens,
  - Logically it’s normal done ASAP inside the child constructor

Abstract classes

- We can mark a class as ***abstract***

  ```
  public abstract class Mammal { . . .
  ```

- This means it cannot be instantiated (created) directly
- It can only be inherited
- Or extended by another abstract class

Polymorphism!

- When we create a subclass it exists as both it’s own type and that of it’s parent
- Thus we can use it interchangeably as either
- If we have a dog named Spot
  - Spot is a dog
  - Spot is a pet
  - Spot is an animal

The one caveat of Inheritance

- You may only inherit from one superclass
- Although it (the super) may inherit from another
- Thus you can chain inheritance
- Spot -> Dog -> Pet -> Animal -> Living Creature

Noroff
School of technology
and digital media

Thursday - Week 1

Interfaces

Interfaces

- Interfaces overcome the problem of multiple inheritance
- They provide a "contract" or a promise that certain things must exist
- We use the *implements* keyword:

```
public class Dog implements Trainable {
// code
}
```

What goes inside an Interface?

- Method signatures
- Variable declarations*
- No code logic

* Always public, only use for constants

An interface is a contract

- Every method in the interface **must** be accounted for in the implementing class
- Variables are automatically accessible (public)

Multiple Interfaces

- We can however implement multiple interfaces in a single class
- This lets as build modular solutions

Interfaces are not Classes

- Thus an interface may inherit from many other interfaces
- This lets us build much more inters testing and dynamic systems

Noroff
School of technology
and digital media

Thursday - Week 1

Tasks

Task 10: Payment System

- Consider a payment system
  - Customers may use cash or card
  - A card may be a savings card or a credit card
  - Cash will probably require change

- Design a system that allows for these options:
  Use an interface or inheritance or both?

- Implement your solution in a simple console application

Task 11: Group Task - Part 1

- For groups of 2 to 3 people (**not solo**)
- Each person hands in on Moodle, list all team members
- This is due Friday afternoon (17:00)

- Watch this video: https://www.youtube.com/watch?v=jPcBU0Z2Hj8
- Read about algebraic notation: https://en.wikipedia.org/wiki/Algebraic_notation_(chess)

Task 11: Group Task - Part 2

- Write a program that takes in a co-ordinate on the board
- It must then tell me if it is is possible to complete the 8-queens problem with a queen at the chosen location
- If it is possible it must print a board (in console) that shows one such possible solution