COMP122 Week 5

Objects and Classes

UNIVERSITY OF LIVERPOOL

Dr. Patrick Totzke
totzke@liverpool.ac.uk

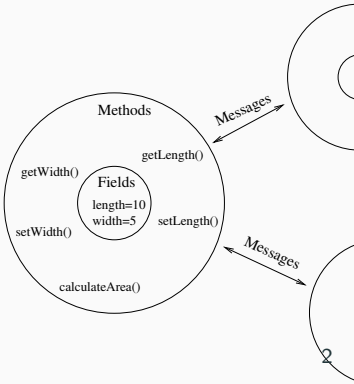
https://liverpool.instructure.com/courses/59716

Objects

An **object** is a model or representation of some part of the world. It contains

- attributes (data) and
- methods (behaviour), which define interaction with other objects.

Each object refers to a particular ATM, person, book, etc.



Example 1

A rectangle has attributes “width” and “height” with values 1 and 2. It can calculate its area.

Example 2

This screen has attributes “width” and “height” with values 2800 and 1752, and a method “nextSlide()”.

Food for thought

What attributes might we need for a class to represent

- a circle?
- a library book?
- a bank account?
- a student?

What methods might we need?

Classes

A **class** is a template or blueprint of all objects of a certain kind.

- Can be instantiated
- Specifies which attributes and methods each instance has, but not their values.
- Defines a data type

Example

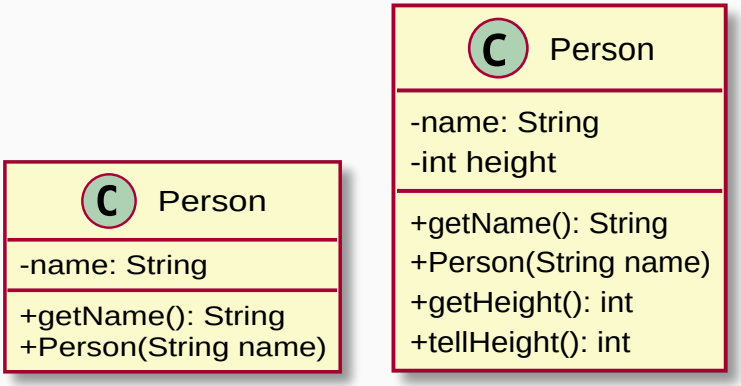
All rectangles have attributes “width” and “height” but different instances may have different values for these attributes.
All rectangles know how to compute their area based on their individual attribute values.

Class definitions in Java

```
1  public class ExampleClass {
2    private int attribute1;
3    private double attribute2;
4
5    // Constructor methods
6    public ExampleClass (int a1){
7      attribute1 = a1;
8    }
9    // Other methods
10   public int calculateValue (int inputVal){
11     return attribute1 * inputVal ;
12   }
13   // main method
14   public static void main (String[] argv){}
15 }
```

Demo

Let's write down the `Person` class from the first lecture:



Instantiating an object

You can create a new instance of a class, i.e., *instantiate* an object using the `new` keyword followed by a call to a constructor method.

Each class definition also defines a type, so you can declare variables of that type etc.

```
ExampleClass obj = new ExampleClass();
Person bojo = new Person("Alexander Boris de Pfeffel Johnson");
```

Here is a simple application that uses the `Person` class

```
public class Greeter {
    public static void main(String [] args) {
        Person bojo = new Person("Alexander Boris de Pfeffel Johnson");

        // this fails because name is a private attribute of Person
        System.out.print(bojo.name);
    }
}
```

Private and Protected Attributes/Methods

- Attributes and methods are categorised as `private` or `protected` members of a class so they cannot be directly accessed by other objects.
- To interact with an object, we must use its interface, i.e. its `public` methods or attributes

Example

A reference to `bojo.name` or `bojo.height` will result in an error because they are `private` to the class.

Loose Coupling

It is good practice to declare attributes to be `private` or `protected`. If the values need to be obtained/modified, a programmer can provide accessors that will return the values and/or mutators to change the value.

Accessors and Mutators/Get and Set Methods

- Public methods are the recommended way to pass information into and out of an object.
- Methods that set or modify an object's instance variables are called *mutators* (or *set methods*); methods that get or retrieve the value of an instance variable are called *accessors* (or *get methods*).
- It is up to the designer of the class to decide which private variables need accessor and mutator methods.
  Example: A Person could have a mutator `setAttire(String nname)` that writes to a (`private String`) attribute `attire`.
  Example: A Bank Account could have a getter `getBalance()` but is unlikely to have a setter `setBalance(int n)`.

Private/Protected/Public

	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
`public`	+	+	+	+	+
`protected`	+	+	+	+	
no modifier	+	+	+		
`private`	+				

Your Questions?

Q30

Q:

> Is there any way that students could be given access to assignment 2 on the 11th of March rather than the 13th, so that we can use the weekend of week six to work on it. Given that there is a tesr for COMP 116 in week 8 and so the weekend of week 7 will need to used to revise for this.

Q31

Q: I was also wondering if the code is all checked purely by the automated marker or whether it is also checked by a human at some point?

Q32

Q: How do I call Caesar.rotate from the Brutus class?

A: Great idea! Since all methods in Caesar are declared public *static*, you can call them on the class like this.

```
1  char Xrotated = Caesar.rotate(5,'x');
```

Q: I submitted multiple times.
Can I pick which one you'll grade?

A: No. I'll only look at your latest submission (and its timestamp).

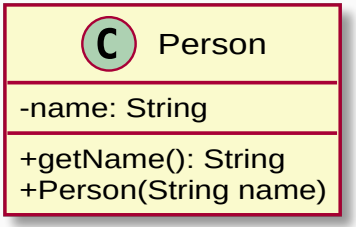
Q33

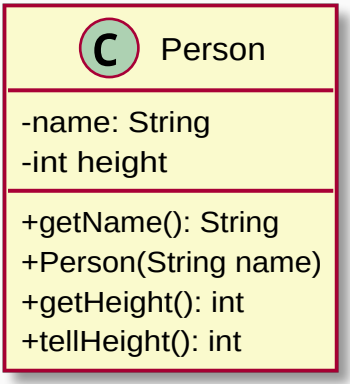
Q: Why do we have a separate Greeter class to host the main method?

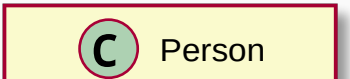
A: This is indeed not necessary.

Demo

Let's write down the `Person` class from the first lecture:







Q: Why is it necessary to have a constructor?

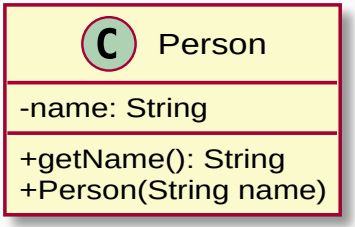
A: It is not. If you don't define one in your class then it will just have a default constructor (one without parameters) that will be used implicitly.

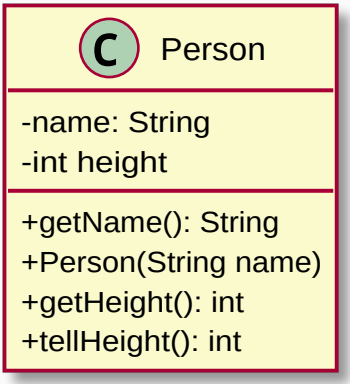
Q: What is `this`?

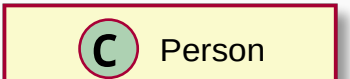
A: This keyword can be used to refer to the current object: to remove ambiguity when referring to attributes, or to call one constructor from another one.

Demo

Let's write down the `Person` class from the first lecture:





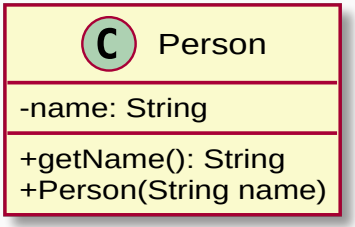


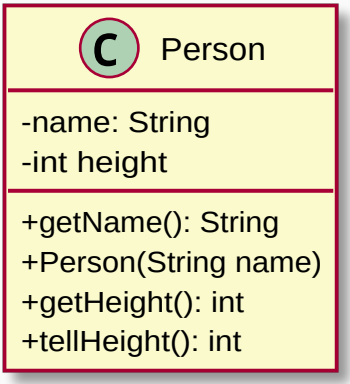
Q: What's the meaning of `static` again?

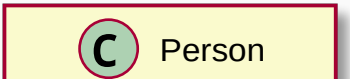
A: static methods/attributes are shared among all instances (i.e., belong to the class).

Demo

Let's write down the `Person` class from the first lecture:







Python?

Summary of Week 5: Start of Block 2

We looked at...

- Objects & Classes
- Constructor Methods
- Persons and Greeters

Next Week:

- Inheritance
- Polymorphism