



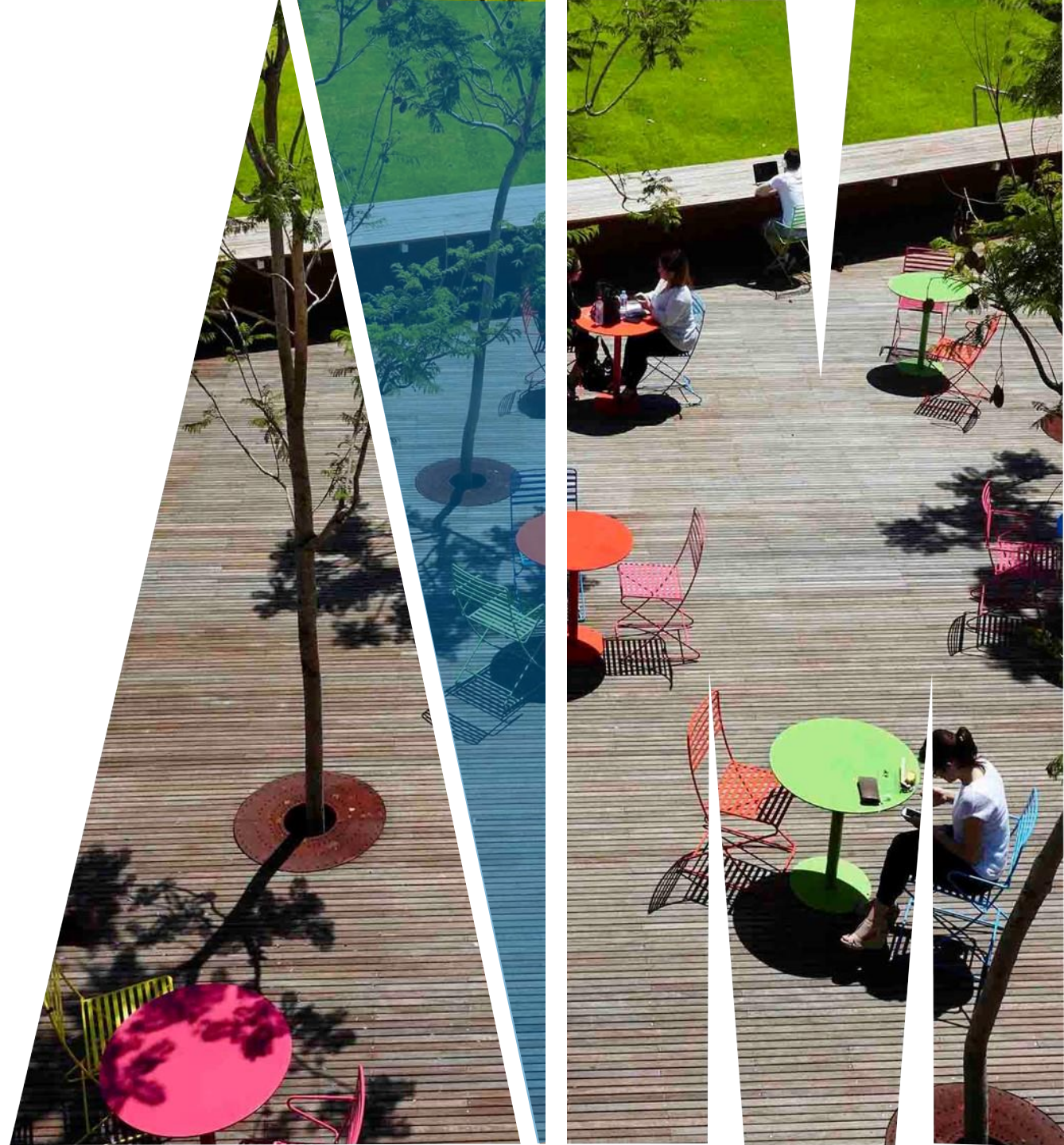
MONASH
University

FIT2099 Object-Oriented Design and Implementation

Classes and objects



MONASH
University



Outline

Encapsulation and Information hiding

Class members

Attributes

Methods

Access modifiers (public and private)

Getters/setters

WHAT IS ENCAPSULATION?

A software development technique that consists of **isolating a system function** or a set of data and operations on those data within a module and providing precise specifications for the module

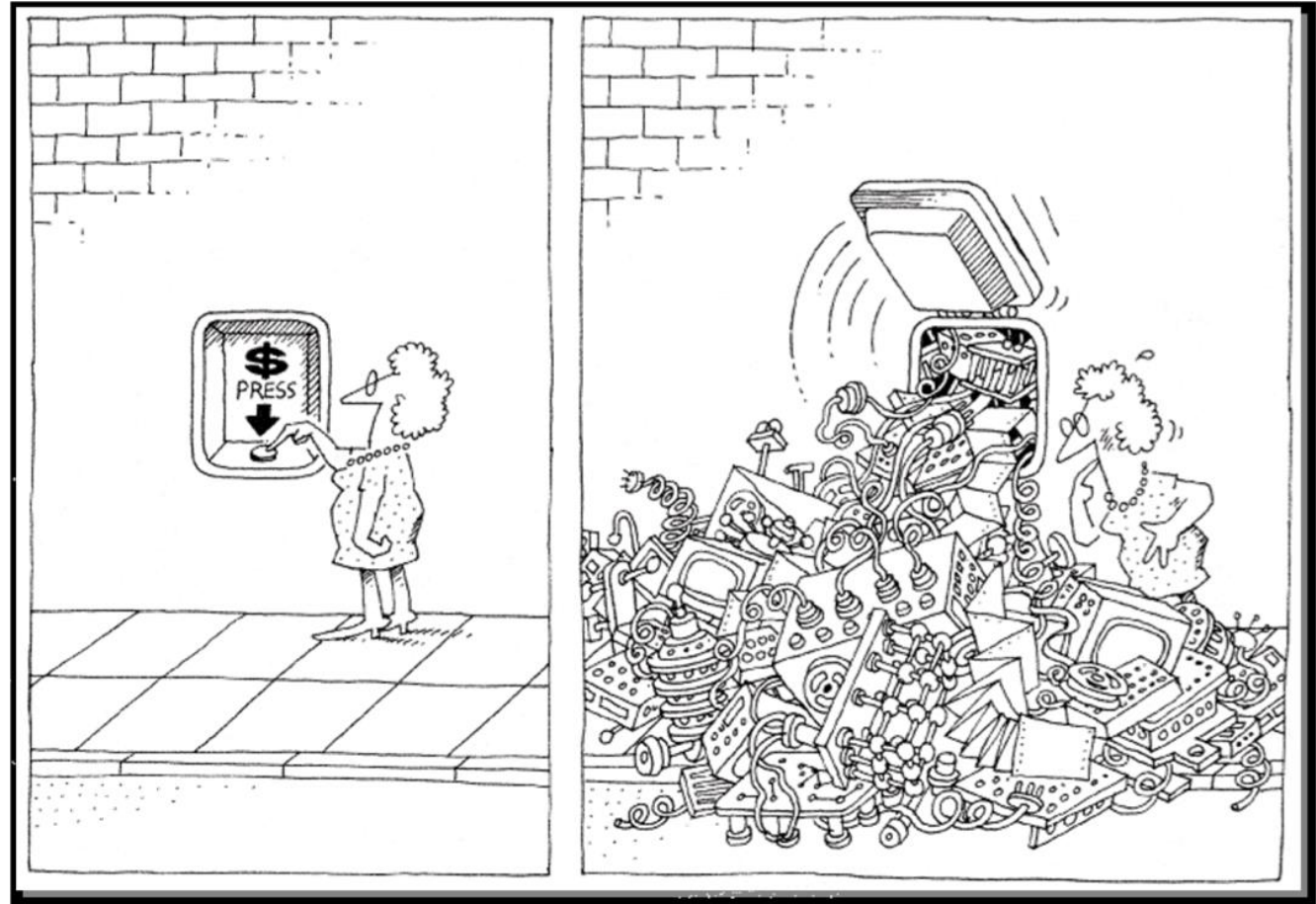
--IEEE Software Engineering Vocabulary



WHAT IS ENCAPSULATION?

Access to the **names, meanings, and values** of the responsibilities of a class is entirely separated from access to their **realization**.

--IEEE Software Engineering
Vocabulary



WHAT IS ENCAPSULATION?

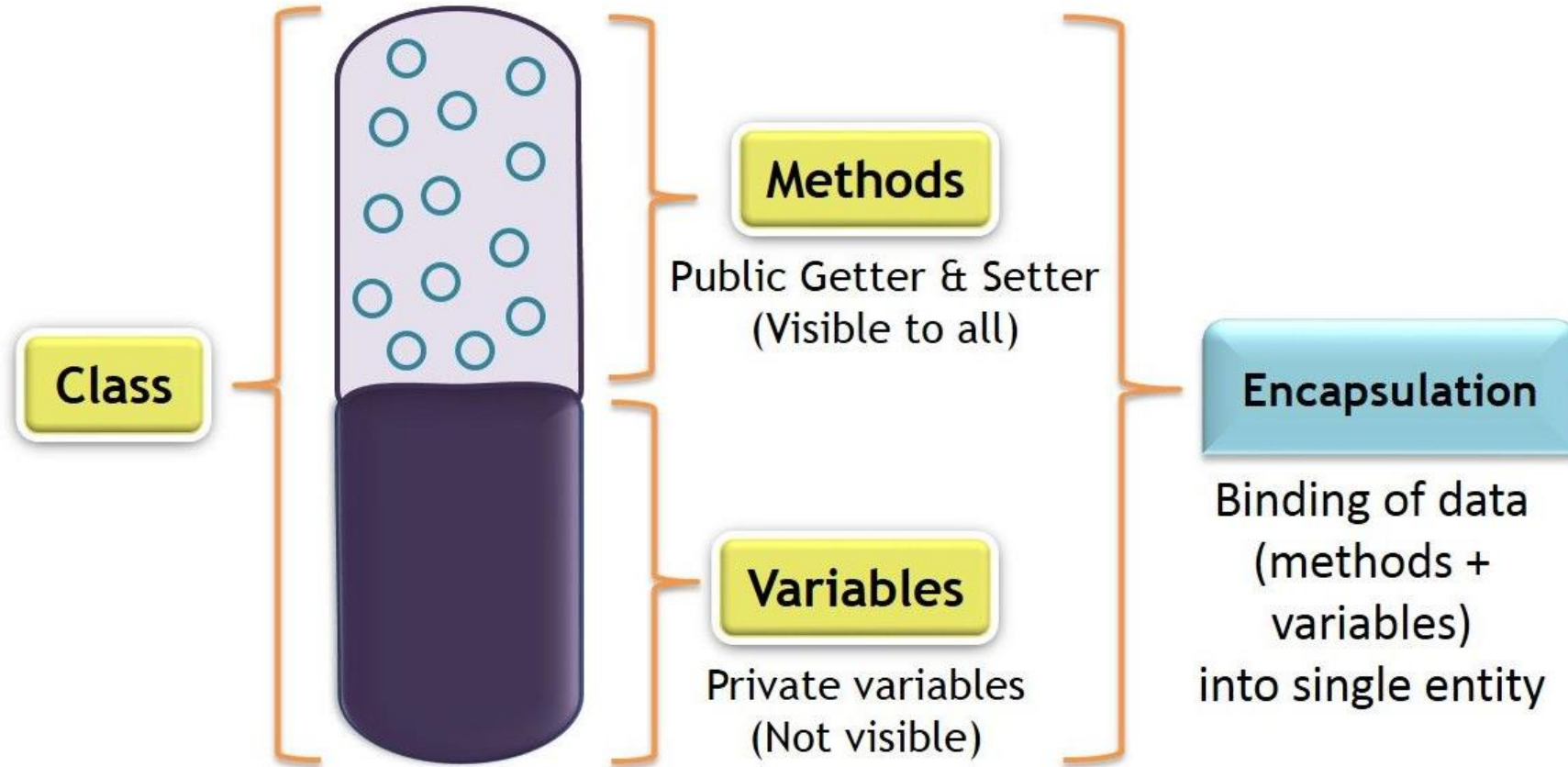
The idea that a module has an outside that is distinct from its inside, that it has an **external interface** and an **internal implementation**

cf. data abstraction, **information hiding**

--IEEE Software Engineering Vocabulary



ONE WAY TO ACHIEVE ENCAPSULATION



THE CLASS



A class is an extensible program-code-**template for creating objects**.

The **members** of a class may be:

i) initial values for state (**attributes** also called *fields* or *data members*).

An object's attributes store its state

These look like variables that are part of the class but not part of a method

ii) implementations of behaviour (member functions or **methods** also called *operations*)

syntax similar to function definitions in non-OO languages

can tell the object to do something (a **command**)

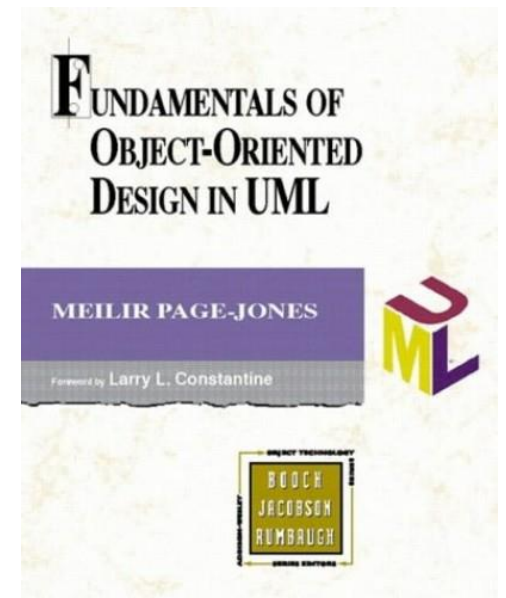
can ask the object a question about itself (a **query**)

PAGE-JONES ON ENCAPSULATION

Encapsulation is the **grouping of ideas into one unit**, which can thereafter be referred to by a **single name**.

Object-oriented encapsulation is the **packaging of operations and attributes** representing state into an object type so that state is accessible or modifiable only via the **interface** provided by the encapsulation.

-- Meilir Page-Jones,
Fundamentals of Object-Oriented Design in UML



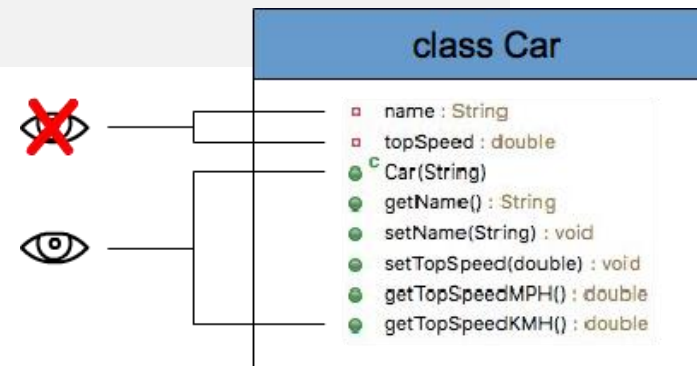
WHAT IS INFORMATION HIDING?

Every module is characterized by its **knowledge of a design decision which it hides from all others**. Its *interface* or *definition* was chosen to reveal as little as possible about its inner workings: data structures, its internal linkings, accessing procedures and modifying procedures are part of a single module.

-- David Parnas (**1972**)

Information/implementation hiding is the use of encapsulation to **restrict from external visibility** certain information or implementation decisions that are internal to the encapsulation structure.

-- Meilir Page-Jones



A CLASS



The image features a large iceberg floating in a blue ocean under a blue sky with white clouds. The iceberg is split horizontally by the water's surface. The top part, which is above water, is relatively small and has a jagged, snow-covered peak. The bottom part, which is submerged, is much larger and has a complex, textured shape with many sharp edges and indentations. Two red curly braces are used to label the parts of the iceberg. One brace is on the right side, spanning the width of the visible peak and pointing to the text 'Public interface'. The other brace is also on the right side, spanning the width of the much larger submerged portion and pointing to the text 'Private information'.

Public interface

Private information

WHY ENCAPSULATING?

Lets us define a **boundary** around chunks of code

then we can hide what's inside that boundary from code outside the boundary

Hiding the stuff that is inside makes those chunks **look simpler** from the outside

simple things are easier to use

Also allows us to prevent unnecessary code dependencies

(**ReD: Reducing Dependencies**)

can't depend on something that you can't see

EXAMINING A CLASS

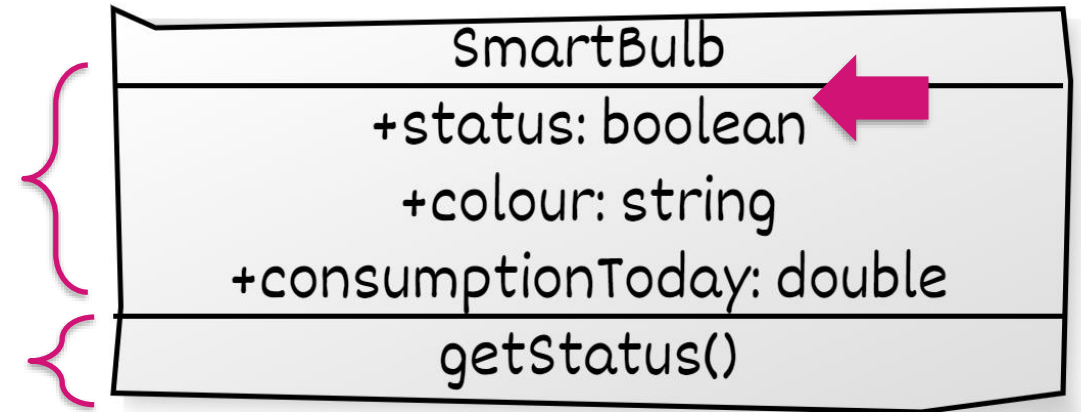
A class is represented by a rectangle having three sections –

- the **top** section containing the **name of the class**
- the **middle** section containing class **attributes**
- the **bottom** section representing **methods** of the class



Attributes

Methods



WHAT IS UML?

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling **language** in the field of software engineering that is intended to provide **a standard way to visualize the design of a system.**



BUILDING A CLASS DIAGRAM

Go to: app.diagrams.net

The screenshot displays the web application 'Week 1 - L02 diagrams.drawio' by Roberto Martinez-Maldonado. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Arrange', 'Extras', and 'Help'. A toolbar below the menu contains various drawing tools. On the left, a 'UML' panel shows a collection of UML symbols, with a red arrow pointing to the 'Class' icon. The central workspace features a class diagram for 'SmartBulb' with the following attributes and methods:

```
classDiagram
    class SmartBulb {
        +status: boolean
        +colour: string
        +wattsToday: double
        +getStatus(): boolean
    }
```

The right sidebar contains settings for 'Diagram' and 'Style'. Under 'View', options include 'Grid' (checked), 'Page View' (checked), 'Background' (unchecked), and 'Shadow' (unchecked). Under 'Options', 'Connection Arrows' (checked), 'Connection Points' (checked), 'Guides' (checked), and 'Autosave' (checked) are listed. The 'Paper Size' is set to 'A4 (210 mm x 297 mm)' in 'Portrait' orientation. At the bottom, there are buttons for 'Edit Data' and 'Clear Default Style', and a status bar showing 'Page-1'.

EXAMINING A CLASS



SmartBulb
<div data-bbox="1396 394 1554 602" data-label="Image"></div> <div data-bbox="1505 602 2201 786" data-label="Text"><p>+ status: boolean + colour: string + consumptionToday: double</p></div>
<div data-bbox="1505 882 2048 939" data-label="Text"><p>+ getStatus(): boolean</p></div>

ACCESS MODIFIERS

Public – A public member is visible from **anywhere in the system**. In class diagram, it is prefixed by the symbol '+’.

Private – A private member is visible only from **within the class**. **It cannot be accessed from outside the class**. A private member is prefixed by the symbol '–’.

A CLASS



The image features a large iceberg floating in a blue ocean under a blue sky with white clouds. The iceberg is split horizontally by the water's surface. The top part, which is above water, is relatively small and has a jagged, snow-covered peak. The bottom part, which is submerged, is much larger and shows a complex, textured structure with many smaller peaks and valleys. Two red curly braces are used to label the parts of the iceberg. One brace is positioned to the right of the above-water part, and the other is to the right of the submerged part.

Public interface

Private information

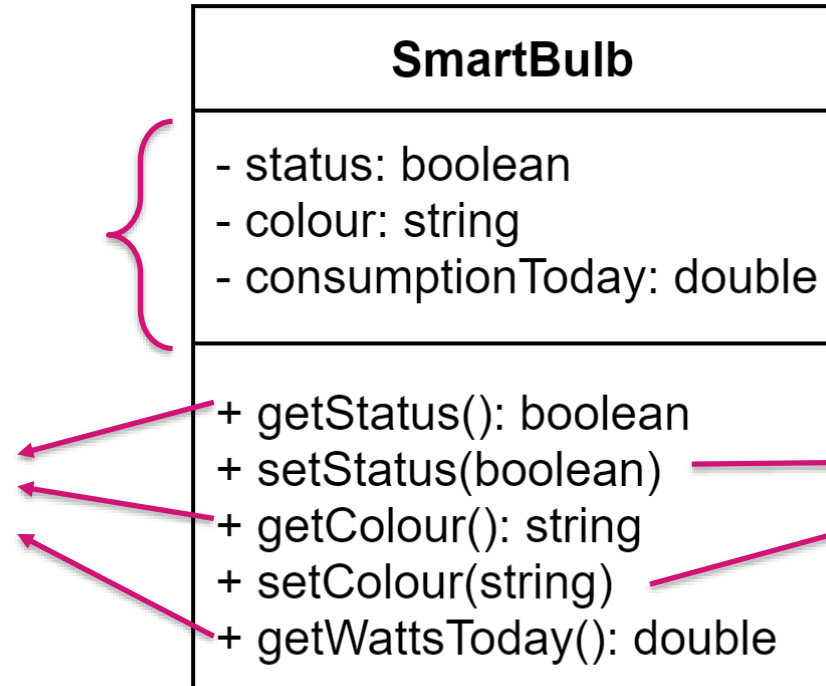
EXAMINING A CLASS



Hidden
information

Getters

Accessing (reading) private information through a public interface. It is an **Accessor**.



Setters

Modifying private information through a public interface. It is a **Mutator**.

Summary

Encapsulation and Information hiding

Class members

Attributes

Methods

Access modifiers (public and private)

Getters/setters



MONASH
University

Thanks



MONASH
University

