

Exercise walk-through

3.7

Class Variables and Methods

AP Computer Science A

Encapsulation, and the responsibility that comes with design

encapsulation

keep **instance variables**
them only through methods

private and expose

accessor (getter)

returns a field's value without letting the caller change it

mutator (setter)

changes a field, and is the one place a validity check can live

overloading

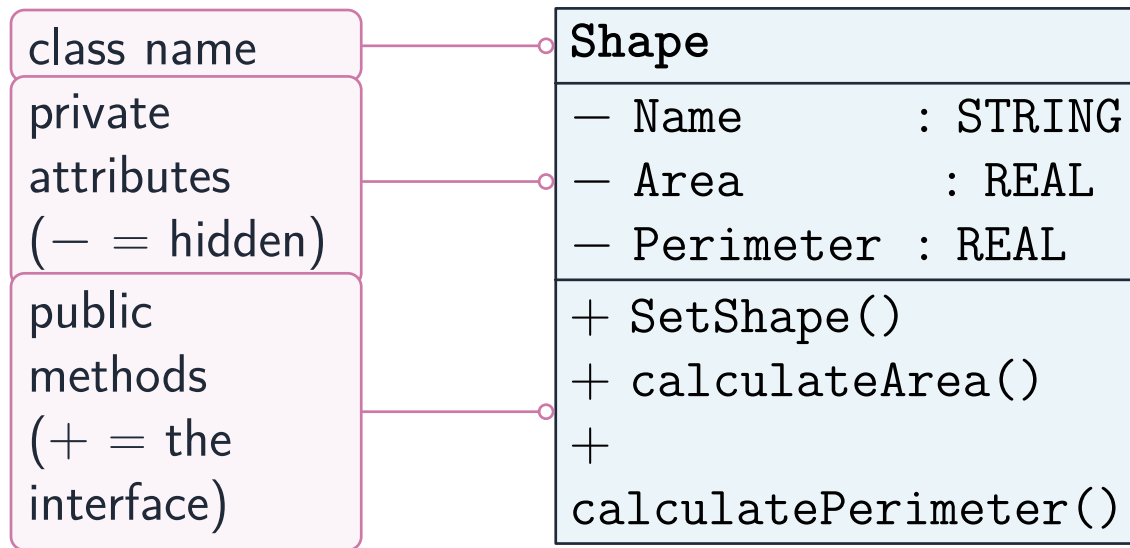
several methods with the **same name** and different parameter lists

system reliability

the program doing its job without failure – sometimes a **legal** requirement, not just a quality goal

Design also carries responsibility beyond the code: **intellectual property** in what you reuse, and **open source** licences that state the conditions of that reuse.

A class diagram



A class diagram: private attributes and public methods

Exam tips

- Design with methods and classes: encapsulate data as `private` fields and expose behaviour through public methods.
- Know the difference between an **object** and its **class**, and that objects are passed **by value** – the parameter gets a copy of the reference, so a method can change the object's state, but reassigning the parameter does not affect the caller (Java has no pass-by-reference).
- Traverse arrays and ArrayLists safely – size is `length` vs `.size()`, and removing during a loop shifts indices.
- **Trace** a **recursive** method to determine its result: find the base case first, then follow each recursive call to its return value (writing recursive code is outside the exam's scope).
- Recognise **inheritance** vocabulary – superclass, subclass, method overriding, and that every class is a subclass of `Object` (designing and implementing inheritance is outside the exam's scope).

Exercises 2.1-2.3

2.1 State what a static variable is shared by.

[1]

2.2 State whether a static variable belongs to the class or to each object.

[1]

2.3 State one use of a static variable.

[1]

Exercises 2.1-2.3

2.1 State what a static variable is shared by. [1]

2.1

- every object of the class
-

2.2 State whether a static variable belongs to the class or to each object. [1]

2.2

- the class
-

2.3 State one use of a static variable. [1]

2.3

- a shared count, or a class constant (any one)

Exercises 3.1-3.2

3.1 A static variable is

[1]

☐

A unique to each object

B shared by all objects of the class

C a local variable

D a parameter

3.2 A static method

[1]

☐

A acts on one object's state

B does not act on an individual object

C requires **new**

D is a constructor

Exercises 3.1-3.2

3.1 A static variable is

[1]

☐

A unique to each object

B shared by all objects of the class

C a local variable

D a parameter

3.1 B

- shared by all objects of the class.

3.2 A static method

[1]

☐

A acts on one object's state

B does not act on an individual object

C requires `new`

D is a constructor

3.2 B

- does not act on an individual object.

Exercise 3.3 (a)

3.3 A class counts how many objects have been created, using a shared count.

(a) Name the kind of variable used for the count.

[1]

Exercise 3.3 (a)

3.3 A class counts how many objects have been created, using a shared count.

(a) Name the kind of variable used for the count.

[1]

(a)

- a static variable

Exercise 3.3 (b)

3.3 A class counts how many objects have been created, using a shared count.

(b) State what it belongs to.

[1]

Exercise 3.3 (b)

3.3 A class counts how many objects have been created, using a shared count.

(b) State what it belongs to.

[1]

(b)

- the class

Exercise 3.3 (c)

3.3 A class counts how many objects have been created, using a shared count.

(c) State whether each object has its own copy of it.

[1]

Exercise 3.3 (c)

3.3 A class counts how many objects have been created, using a shared count.

(c) State whether each object has its own copy of it.

[1]

(c)

- no – there is one shared copy