

Exercise walk-through

3.2

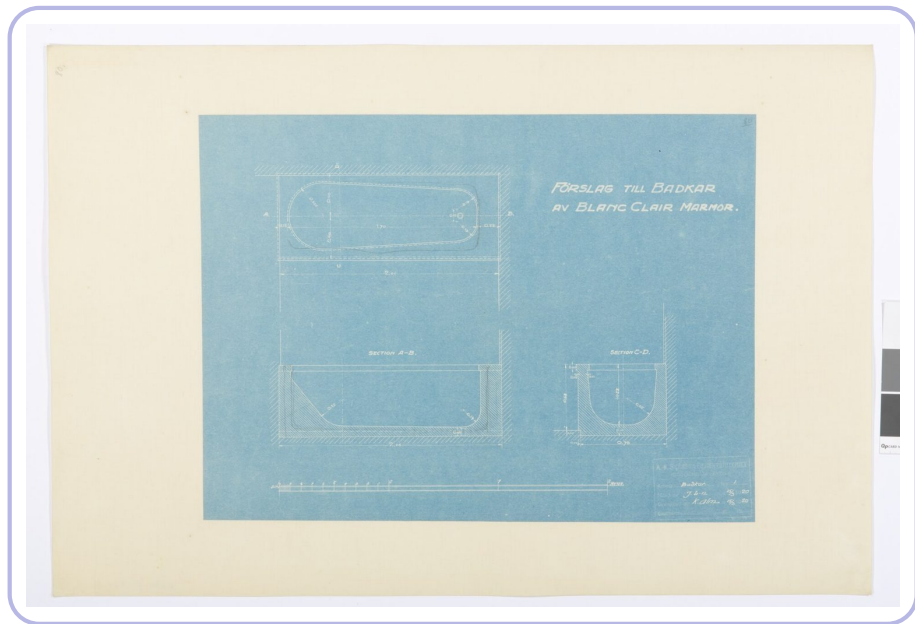
Impact of Program Design

AP Computer Science A

Abstraction & program design

Abstraction hides detail behind a simple interface, so we manage complexity one piece at a time.

Good **program design** breaks a problem into classes, each with one clear responsibility. A well-designed class is easier to test, reuse, and change without breaking the rest.



Class blueprint

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

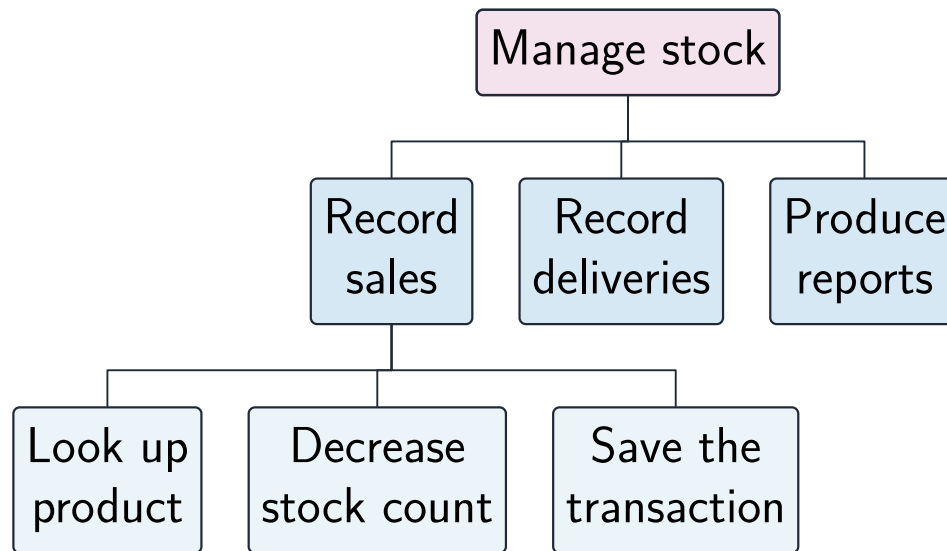
several methods with the **same name** and different parameter lists

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.

Decomposing a program into modules and sub-modules



Decomposing a program into modules and sub-modules

Exercises 2.1-2.3

2.1 State one quality that good program design improves. [1]

2.2 State why you should test with a wide range of inputs. [1]

2.3 State what it means for a program to be “maintainable”. [1]

Exercises 2.1-2.3

2.1 State one quality that good program design improves. [1]

2.1

- reliability or maintainability (any one)
-

2.2 State why you should test with a wide range of inputs. [1]

2.2

- to find defects that only appear on some inputs
-

2.3 State what it means for a program to be “maintainable”. [1]

2.3

- it is easy to understand and change

Exercises 3.1-3.2

3.1 Good program design improves a program's

[1]

☐

A speed only

B reliability and maintainability

C file size

D colour

3.2 Testing with a wide range of inputs helps to

[1]

☐

A slow the program down

B find defects

C add bias

D compress data

Exercises 3.1-3.2

3.1 Good program design improves a program's

[1]

☐

A speed only

B reliability and maintainability

C file size

D colour

3.1 B

- reliability and maintainability.

3.2 Testing with a wide range of inputs helps to

[1]

☐

A slow the program down

B find defects

C add bias

D compress data

3.2 B

- find defects.

Exercise 3.3 (a)

3.3 A poorly designed program is hard to change and often breaks.

(a) State one quality it lacks.

[1]

Exercise 3.3 (a)

3.3 A poorly designed program is hard to change and often breaks.

(a) State one quality it lacks.

[1]

(a)

- reliability (or maintainability)

Exercise 3.3 (b)

3.3 A poorly designed program is hard to change and often breaks.

(b) State one way to catch its bugs.

[1]

Exercise 3.3 (b)

3.3 A poorly designed program is hard to change and often breaks.

(b) State one way to catch its bugs.

[1]

(b)

- test it with many varied inputs

Exercise 3.3 (c)

3.3 A poorly designed program is hard to change and often breaks.

(c) State one risk that poor design poses to users.

[1]

Exercise 3.3 (c)

3.3 A poorly designed program is hard to change and often breaks.

(c) State one risk that poor design poses to users.

[1]

(c)

- unintended consequences