2.1.1 Thinking abstractly

OCR A-Level Computer Science (H446) - H446/02 - Exam questions and mark schemes

5 past paper questions and 3 practice questions, 28 marks in total. Answer each question in the space provided, then check your work against the mark schemes at the end. compsciwizard.com

Question 1 H446/02 [2 marks]

Define the term 'abstraction' in the context of computational thinking.

Question 2 H446/02 [4 marks]

Explain how abstraction is used when designing a class diagram for an online shopping system.

Question 3 H446/02 [4 marks]

Describe two benefits of using abstraction when developing software.

Question 4 H446/02 [3 marks]

A map application shows roads and landmarks. Explain how abstraction has been used in the design of this application.

Question 5 H446/02 [4 marks]

Explain the difference between representational abstraction and abstraction by generalisation.

Question 6 Practice question (H446/02 style) [4 marks]

A school wants a program that produces its lesson timetable. Devise an abstract model for this problem by identifying two features that must be included and two real-world details that can be ignored. Justify your choices.

Question 7 Practice question (H446/02 style) [3 marks]

Explain, using an example of each, the difference between representational abstraction and abstraction by generalisation.

Question 8 Practice question (H446/02 style) [4 marks]

Describe two ways in which abstraction is used when writing a program in a high-level language.

Mark schemes

One mark per valid point unless the scheme says otherwise. Practice questions are written by Comp Sci Wizard in the style of OCR mark schemes.

Question 1 [2 marks] H446/02

Removing unnecessary detail / Focusing on important aspects / Hiding complexity / Simplifying a problem to its essential features (2 marks)

Question 2 [4 marks] H446/02

Focus on relevant attributes (name, price) not physical properties / Group similar items into classes / Hide implementation details / Use inheritance to share common properties / Show relationships not internal workings (4 marks)

Question 3 [4 marks] H446/02

Reduces complexity - easier to understand / Allows focus on relevant details / Enables code reuse / Makes maintenance easier / Allows working at higher level / Hides implementation from users (4 marks - 2 per benefit)

Question 4 [3 marks] H446/02

Roads simplified to lines / Buildings shown as simple shapes / Ignores physical texture/material / Only shows relevant features for navigation / Scale adjusted to remove unnecessary detail / Color coding for different road types (3 marks)

Question 5 [4 marks] H446/02

Representational abstraction: Removing unnecessary details while keeping essential information / Simplifying representation of real-world objects
Abstraction by generalisation: Grouping similar entities by common characteristics / Creating categories or classes / Identifying shared properties (4 marks - 2 per type)

Question 6 [4 marks] Practice question (H446/02 style)

Include (1 mark each, max 2), e.g.:
• Rooms and their capacities
• Teachers and when they are available
• Classes / subjects and the number of periods each needs

Ignore (1 mark each, max 2), e.g.:
• The colour or decoration of rooms
• The names of individual students (only class sizes matter)
• The weather / what teachers wear

Justification: included features affect whether a timetable is valid; ignored details do not change the solution

(4 marks)

Question 7 [3 marks] Practice question (H446/02 style)

• Representational abstraction removes unnecessary detail so only what is needed remains
• e.g. a tube map shows stations and connections but not real distances or geography
• Abstraction by generalisation groups things into categories / a hierarchy according to shared characteristics
• e.g. treating cars, buses and lorries as 'vehicles' in a parking system

(3 marks – 1 per point, max 3)

Question 8 [4 marks] Practice question (H446/02 style)

Any two, 2 marks each:

- Variables abstract memory addresses – the programmer uses a name rather than a binary address
- Functions / procedures hide their internal code so they can be called by name (procedural abstraction)
- Data structures such as lists and objects hide how the data is actually stored in memory
- A single high-level statement stands for many machine-code instructions

(4 marks)