

THEME B: COMPUTATIONAL THINKING · B3.1

Fundamentals of OOP for a single class — Mark scheme

34 marks · covers B3.1.1–B3.1.5

How to mark. Each creditworthy point is on its own line and ends with a semicolon (;). A line beginning with ... is an *expansion* of the line above it, so the pair is worth the marks shown. A slash (/) separates acceptable alternative wordings. Words in (brackets) are not needed for the mark. Mark positively: credit what the student has got right rather than penalising what is missing, and accept any valid alternative not listed here. Do not award more than the maximum for a part. Where a later part uses a wrong answer from an earlier part correctly, award follow-through marks (FT). HL marks parts that assess higher level only content.

Q	ANSWER	MARKS
1(a) B3.1.1	Award [2] max Award [1] for each. A class is a template or blueprint that defines the attributes and methods that objects of that type will have; An object is a specific instance created from that class, with its own values for those attributes;	2
1(b) B3.1.1	Award [4] max This is an evaluate question (AO3), so both sides are required. Award [1] per point up to [4] max, but award a maximum of [3] if only one side is given. Arguments for OOP Related data and the operations on it are kept together in one class, so the code reflects the problem being modelled; Encapsulation means each class controls its own state, so one developer's code cannot corrupt another's data directly; Developers can work on separate classes at the same time with a clearly defined interface between them; Inheritance and polymorphism allow shared behaviour to be written once and reused, reducing duplication; A class can be tested and replaced independently, which helps maintenance in a large codebase; Arguments against OOP It adds structure and vocabulary that is unnecessary overhead for a small or simple program; Poorly designed class hierarchies can be harder to follow than straightforward procedural code; Object creation and method dispatch carry a small performance and memory cost; It takes longer to design up front, and a bad design is expensive to change later; Judgement (credit any supported conclusion, for example) For a large program with several developers the benefits of clear boundaries and independent testing outweigh the extra design effort, which is why OOP is the usual choice at that scale;	4
2(a) B3.1.2	Award [4] max	4

Q	ANSWER	MARKS
	Award [1] for a three-part box with the class name at the top; [1] for the three attributes with types; [1] for sensible methods including assign and release; [1] for visibility markers used correctly (private attributes, public methods). <pre> +-----+ Locker +-----+ - lockerNumber : int - size : String - occupied : boolean +-----+ + Locker(number, size) + assign() : boolean + release() : void + isOccupied() : boolean +-----+ </pre> Accept any reasonable attribute names and a reasonable set of methods, provided the diagram has the three-compartment structure. Accept getters and setters in place of <code>isOccupied</code>. Do not award the visibility mark if attributes are shown as public.	
2(b) B3.1.2	Award [2] max Award [1] for each. Private: a minus sign (-); Public: a plus sign (+); Accept the mention of # for protected as additional but not required.	2
3(a) B3.1.4	Award [5] max Award [1] for a correct class declaration; [1] for the three attributes declared private; [1] for a constructor with the two parameters; [1] for assigning both parameters to the attributes; [1] for setting occupied to false in the constructor. Accept either language. JAVA <pre> public class Locker { private int lockerNumber; private String size; private boolean occupied; public Locker(int number, String lockerSize) { lockerNumber = number; size = lockerSize; occupied = false; } } </pre> PYTHON <pre> class Locker: def __init__(self, number, size): self.__locker_number = number self.__size = size self.__occupied = False </pre>	5

Q	ANSWER	MARKS
	Python has no true private keyword; accept a leading underscore or double underscore as the equivalent, or a plain attribute name provided the candidate states the intent. Do not award the final mark if <code>occupied</code> is left uninitialised.	
3(b) B3.1.4	Award [2] max Award [1] for calling the constructor correctly and [1] for storing the result in a variable. JAVA <pre>Locker l12 = new Locker(12, "large");</pre> PYTHON <pre>l12 = Locker(12, "large")</pre>	2
4(a) B3.1.5	Award [3] max Award [1] per point up to [3] max. Both terms should be addressed. Encapsulation means bundling the data and the methods that operate on it together inside one class; Information hiding means restricting direct access to that data from outside the class, typically by making attributes private; Access is then provided only through public methods, which the class controls; ...(so) the class can validate any change and keep its own state consistent; The internal representation can be changed later without breaking code that uses the class, as long as the public methods stay the same;	3
4(b) B3.1.5	Award [2] max Award [1] for the point and [1] for the expansion. Must be applied to this attribute. If it were public, any part of the program could set a locker to unoccupied while a member's belongings were still inside; ...(so) making it private forces changes to go through <code>assign()</code> and <code>release()</code>, which can check the current state first; A public attribute could be set to an inconsistent value with no check; ...(so) the object's state would no longer reflect reality, and two members could be given the same locker;	2
5(a) B3.1.3	Award [3] max Award [1] per point up to [3] max. The contrast must be clear. An instance variable belongs to one object, and every object has its own separate copy, whereas a static variable belongs to the class and there is only one copy shared by all objects; An instance variable is created when the object is created, whereas a static variable exists as soon as the class is loaded, before any object is made; A change to a static variable is visible to every object, whereas a change to an instance variable affects only that object; A static variable is accessed through the class name rather than through an object;	3
5(b) B3.1.3	Award [2] max Award [1] for a sensible use and [1] for the justification. A static counter, for example <code>totalLockers</code>, incremented in the constructor;	2

Q	ANSWER	MARKS
	The count belongs to the class as a whole rather than to any single locker, so one shared copy is correct and every new object automatically updates the same total; Accept other valid uses, such as a static constant holding the number of sizes available.	
6 B3.1.4 B3.1.5	Award [5] max Award [1] for a correct method header returning a Boolean; [1] for testing the current state; [1] for setting occupied to true only when it was free; [1] for returning true in that case; [1] for returning false without changing anything when it was already occupied. Accept either language. JAVA <pre> public boolean assign() { if (occupied == false) { occupied = true; return true; } else { return false; } } </pre> PYTHON <pre> def assign(self): if not self.__occupied: self.__occupied = True return True else: return False </pre> Accept <code>if (!occupied)</code>. Award [3] max if the method sets <code>occupied</code> to true unconditionally, since an already-assigned locker would then be reassigned. Award [4] max if the method prints instead of returning on either path.	5

Total for this paper: 34 marks