ANSWERS

A1.7 Classes and objects

Programming techniques and object-oriented programming · A level · OCR H446 1.2.4, AQA 7517 4.1.2.3, Eduqas A500QS 1.4 · about 20 min

What this lesson is about

Classes, objects, attributes, methods and constructors; encapsulation, access specifiers, getters and setters, and class diagrams.

- Class, object, attribute, method
- Encapsulation and information hiding
- Class diagrams
- In the exam languages

Questions

5 marks in all

1. What is instantiation? [1 mark]

- ☐ **A** Creating an object from a class
- ☐ **B** Defining the methods of a class
- ☐ **C** Hiding attributes from other code
- ☐ **D** Making a subclass

> **Answer:** A. An object is an instance of a class; making one is instantiation.

2. What does this program print? [1 mark]

```
class Counter:
    def __init__(self, start):
        self.__count = start

    def up(self):
        self.__count = self.__count + 1
        return self.__count

a = Counter(0)
b = Counter(10)
a.up()
a.up()
print(a.up(), b.up())
```

> **Answer:**
>
> ```
> 3 11
> ```
>
> Each object has its own private count: a goes to 3, b to 11.

3. In a class diagram, what does the symbol - before an attribute mean? [1 mark]

- ☐ **A** Private
- ☐ **B** Public
- ☐ **C** Protected
- ☐ **D** Static

> **Answer:** A. + is public, - is private and # is protected.

4. Which are reasons for making an attribute private and providing a setter method? [1 mark]

Tick every answer that is true.

- ☐ **A** The setter can refuse invalid values
- ☐ **B** The way the value is stored can change without breaking code that uses the class
- ☐ **C** Other code can change the attribute directly
- ☐ **D** The object cannot be put into an invalid state from outside

> **Answer:** A, B, D. Encapsulation means outside code must go through the methods, which control every change.

5. What is the name of the method that runs automatically when an object is created, to give its attributes their starting values? [1 mark]

> **Answer:** constructor. In Python the constructor is __init__; in OCR's reference language it is new.

The task: the odometer class

Write the class `Odometer` from the class diagram. - The constructor `__init__(self, name)` stores `name` (a string) in the private attribute `self.__name` and sets the private attribute `self.__total` to 0. - `drive(self, cm)` takes a whole number of cm. If `cm` is 0 or more it drives forward `cm`; if it is negative it drives backward by `-cm`. Either way it adds the size of the move, `abs(cm)`, to `self.__total`. - `get_total(self)` returns the total. - `report(self)` returns the string `<name>: <total> cm`. Then make two objects, `trip_a = Odometer("trip A")` and `trip_b = Odometer("trip B")`, and call `trip_a.drive(20)`, `trip_b.drive(-10)` and `trip_a.drive(15)` in that order. Print `trip_a.report()` then `trip_b.report()`, which should show `trip A: 35 cm` and `trip B: 10 cm`.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

class Odometer:
    def __init__(self, name):
        self.__name = name

trip_a = Odometer("trip A")
```

The hint students can ask for: Each object needs its own name and its own running total, so both belong in the constructor. drive(cm) does the driving and adds to that object's total only. Think about what a negative distance should add.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

class Odometer:
    def __init__(self, name):
        self.__name = name
        self.__total = 0

    def drive(self, cm):
        if cm >= 0:
            forward(50, distance=cm)
        else:
            backward(50, distance=-cm)
        self.__total = self.__total + abs(cm)

    def get_total(self):
        return self.__total

    def report(self):
        return f"{self.__name}: {self.get_total()} cm"

trip_a = Odometer("trip A")
trip_b = Odometer("trip B")
trip_a.drive(20)
trip_b.drive(-10)
trip_a.drive(15)
print(trip_a.report())
print(trip_b.report())
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.