OOP: concepts 4

A Level

P P

In the object-oriented programming paradigm, code is organised into classes. Within a class you will find attributes and methods with access modifiers that specify which of these properties can be accessed from outside the class.

Study the following class definition:

Pseudocode

```
CLASS Elf
    PRIVATE strength: Integer
    PRIVATE speed: Integer
    PUBLIC power: String

    PUBLIC PROCEDURE Elf(given_strength, given_speed)
        strength = given_strength
        speed = given_speed
        power = "Archery"
    ENDPROCEDURE

    PUBLIC FUNCTION get_strength()
        RETURN strength
    ENDFUNCTION
ENDCLASS

aegnor = NEW Elf(20, 50)
```

The table below gives a list of terms that are relevant to OOP classes. Use the class definition above to pick an appropriate example for each of the terms. Drag the example into the cell next to the term.

Term	Label
An attribute	
A method	
A class	
A reference variable	
An access modifier	

Term	Label
A data type	
A parameter	

Items:

- strength
- Integer
- PRIVATE
- Elf
- aegnor
- get_strength
- given_strength

Constructor method definition

A Level

P P

What is a constructor method?

○ A method that is used to instantiate an object.

○ A public method that is used to set the values of private attributes.

○ A method that is used to create a child class from the definition of a parent class

○ A method that is used to get the values of public attributes outside the class in which they were defined

OOP: complete code 2

A Level

P P

Jemma is developing a game for her younger brother that takes place in space. So far, she has created part of the definition of the `Planet` class.

Select the statement that can create an object of the `Planet` class called `mars` so that it is habitable, has two satellites, is 227.9 million kilometres away from the sun, and has a dominant colour of red.

Pseudocode

```
CLASS Planet
    PRIVATE distance_sun  // Float given in million km
    PRIVATE satellites  // Integer
    PRIVATE habitable  // Boolean
    PRIVATE main_colour  // String

    PUBLIC PROCEDURE Planet(given_distance_sun, given_satellites,
given_habitable, given_main_colour)
        distance_sun = given_distance_sun
        satellites = given_satellites
        habitable = given_habitable
        main_colour = given_main_colour
    ENDPROCEDURE

    PROCEDURE set_distance_sun(number)
        distance_sun = number
    ENDPROCEDURE
ENDCLASS
```

○ `mars = NEW Planet(227.9, 2, True, 'red')`

○ `Planet = NEW mars(True, 2, 227.9, red)`

○ `mars = NEW Planet(mars.set_distance_sun(227.9), 2, True, 'red')`

○ `mars = NEW Planet(227.9 million km, 2, True, red)`

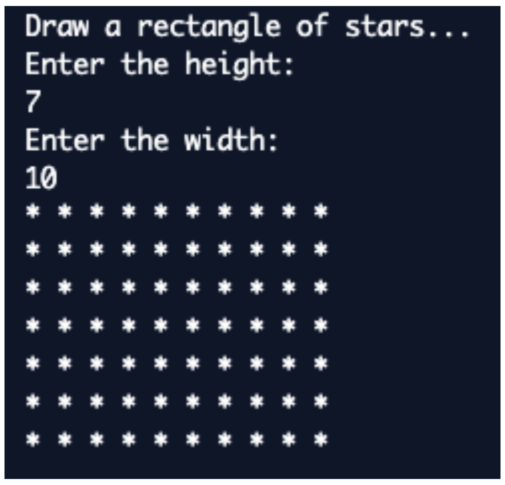
OOP: complete code 5

A Level

P P

Steve is learning object-oriented programming. He wants to write some code that will draw a simple rectangle, so he has created a class called `Rectangle`, with two attributes: `height` and `width` and a method `draw`.

The `draw` method will display a rectangle of stars of a given height and width. For example, a rectangle of height 7 and width 10, will look like this:



```
Draw a rectangle of stars...
Enter the height:
7
Enter the width:
10
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

Example output from Steve's program

Pseudocode

```
CLASS Rectangle
    PRIVATE width
    PRIVATE height

    PUBLIC PROCEDURE Rectangle(given_width, given_height)
        width = given_width
        height = given_height
    ENDPROCEDURE

    PUBLIC PROCEDURE draw()
        FOR column = 0 TO height
            FOR row = 0 TO width
               PRINT("* ")
            NEXT row
        NEXT column
    ENDPROCEDURE
ENDCLASS

PRINT("Draw a rectangle of stars…")
height = INPUT("Enter the height: ")
width = INPUT("Enter the width: ")

my_shape = NEW Rectangle(_______, _______) // Missing code
my_shape.draw()
```

In the pseudocode shown above, a new Rectangle object is instantiated and its `draw` method is called. However, the line of code that instantiates the object has missing arguments. Can you identify what they should be?

Enter your answer as `argument1, argument2` using a comma to separate the arguments.

OOP: complete code 6

A Level

P P

The following class has been defined using pseudocode.

Pseudocode

```
CLASS Radio
    PRIVATE volume: integer
    PRIVATE station: string
    PRIVATE on: Boolean

    PUBLIC PROCEDURE Radio(given_station)
        station = given_station
        volume = 3
        on = False
    ENDPROCEDURE

    PUBLIC FUNCTION get_volume()
        RETURN volume
    ENDFUNCTION

    PUBLIC FUNCTION get_station()
        RETURN station
    ENDFUNCTION

    PUBLIC FUNCTION is_on()
        RETURN on
    ENDFUNCTION

    PUBLIC PROCEDURE set_volume(new_volume)
        volume = new_volume
    ENDPROCEDURE

    PUBLIC PROCEDURE set_station(new_station)
        station = new_station
    ENDPROCEDURE

    PUBLIC PROCEDURE switch()
        IF on == True THEN
            on = False
        ELSE
            on = True
        ENDIF
    ENDPROCEDURE
ENDCLASS
```

Which of the following options (expressed in pseudocode) will correctly create an instance of a radio object?

○ `Radio = NEW radio1("Capital FM", 3, True)`

○ `radio1 = NEW Radio("Capital FM")`

○ `radio1 = NEW Radio("Capital FM", 3, True)`

○ `Radio = NEW radio1("Capital FM")`

OOP: complete code 1

A Level

C C

Mike is creating a game inspired by *The Lord of the Rings*. Each character of the game can belong to one of the following tribes: Elves, Dwarves, Hobbits, Men, Wizards, Orcs, and Trolls. The characters that belong to the tribe of Elves have a commonly known name and a secret elven name.

A part of the definitions of the `Elf` and `Character` classes is presented below. `SUPER` is used to call the constructor of the `Elf` parent class from the `Character` child class.

In the main program, an instance of the `Character` class called `my_character` is created, and then **an output statement** is used to demonstrate the value of an attribute.

Pseudocode

```
CLASS Elf
    PRIVATE strength: Integer
    PRIVATE speed: Integer
    PUBLIC power: String

    PUBLIC PROCEDURE Elf(given_strength, given_speed)
        strength = given_strength
        speed = given_speed
        power = "Archery"
    ENDPROCEDURE

    PUBLIC FUNCTION get_strength()
        RETURN strength
    ENDFUNCTION
ENDCLASS

CLASS Character EXTENDS Elf
    PRIVATE elf_name: String
    PUBLIC name: String

    PUBLIC PROCEDURE Character(given_strength, given_speed, given_elf_name,
given_name)
        SUPER(given_strength, given_speed)
        elf_name = given_elf_name
        name = given_name
     ENDPROCEDURE

    PUBLIC FUNCTION get_elf_name()
        RETURN elf_name
    ENDFUNCTION
ENDCLASS

// Main program
PROCEDURE new_character()
    my_character = NEW Character(200, 1000, "Greenleaf", "Legolas")
    PRINT(...................) // Missing code for the output statement
ENDPROCEDURE
```

Look at the list of output statements and select the **two** statements that will cause an error if they are used in the main program to display the value of an attribute.

□ `PRINT(my_character.elf_name)`

□ `PRINT(my_character.name)`

□ `PRINT(my_character.speed)`

□ `PRINT(my_character.power)`

□ `PRINT(my_character.get_elf_name())`

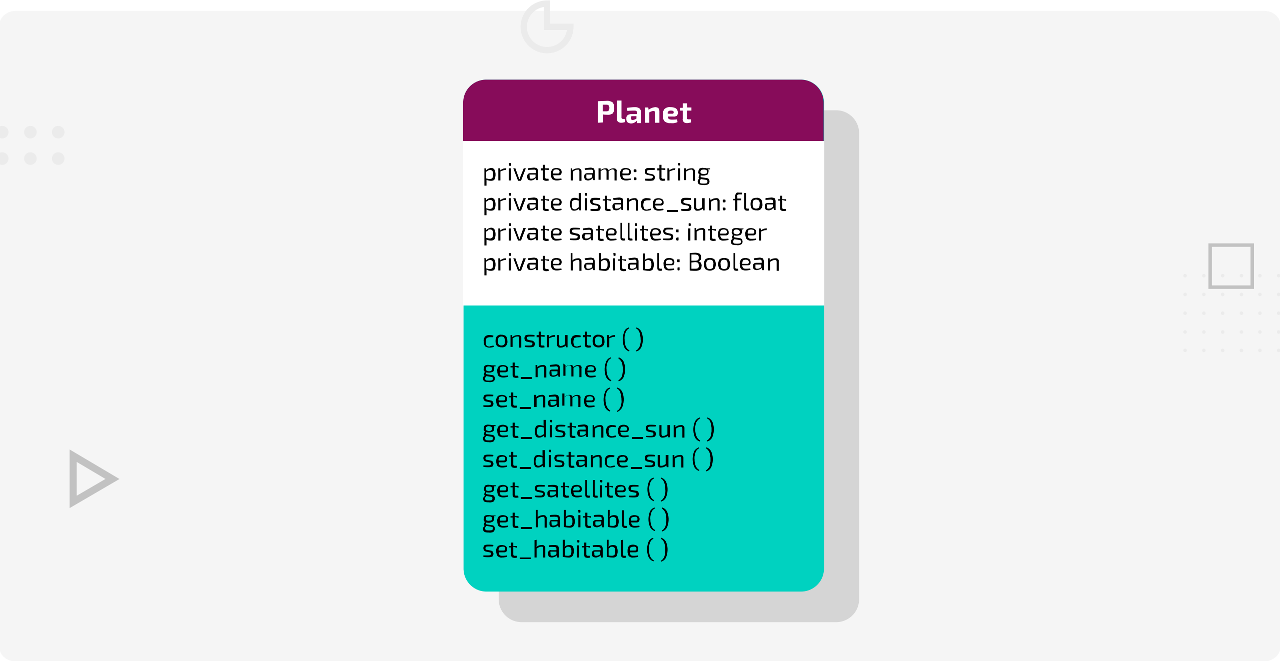
□ `PRINT(my_character.get_strength())`

OOP: complete code 3

A Level

P P

Jemma has extended a game that she is developing for her younger brother that takes place in space. A description of the Planet class that she has defined in her program is presented below.



Planet

private name: string
private distance_sun: float
private satellites: integer
private habitable: Boolean

constructor ( )
get_name ( )
set_name ( )
get_distance_sun ( )
set_distance_sun ( )
get_satellites ( )
get_habitable ( )
set_habitable ( )

Planet class

The Galaxy class contains a planets array that stores the objects of type Planet that are used in the game. It also contains a show_habitable_planets method that searches the planets array and shows the number of planets that are habitable.

Select the statement that completes the missing code in the if statement of the show_habitable_planets method.

Pseudocode

```
CLASS Galaxy
    PRIVATE Planet planets[15] // Array of type Planet

    PUBLIC FUNCTION show_habitable_planets()
        total_habitable = 0 // Number of habitable planets
        FOR count = 0 TO LEN(planets) - 1
            IF ........................ THEN // Missing code
                total_habitable = total_habitable + 1
            ENDIF
        NEXT count
        RETURN total_habitable
    ENDFUNCTION
ENDCLASS
```

○ `Galaxy[count].get_habitable() == True`

○ `Planet.habitable == True`

○ `planets.habitable == True`

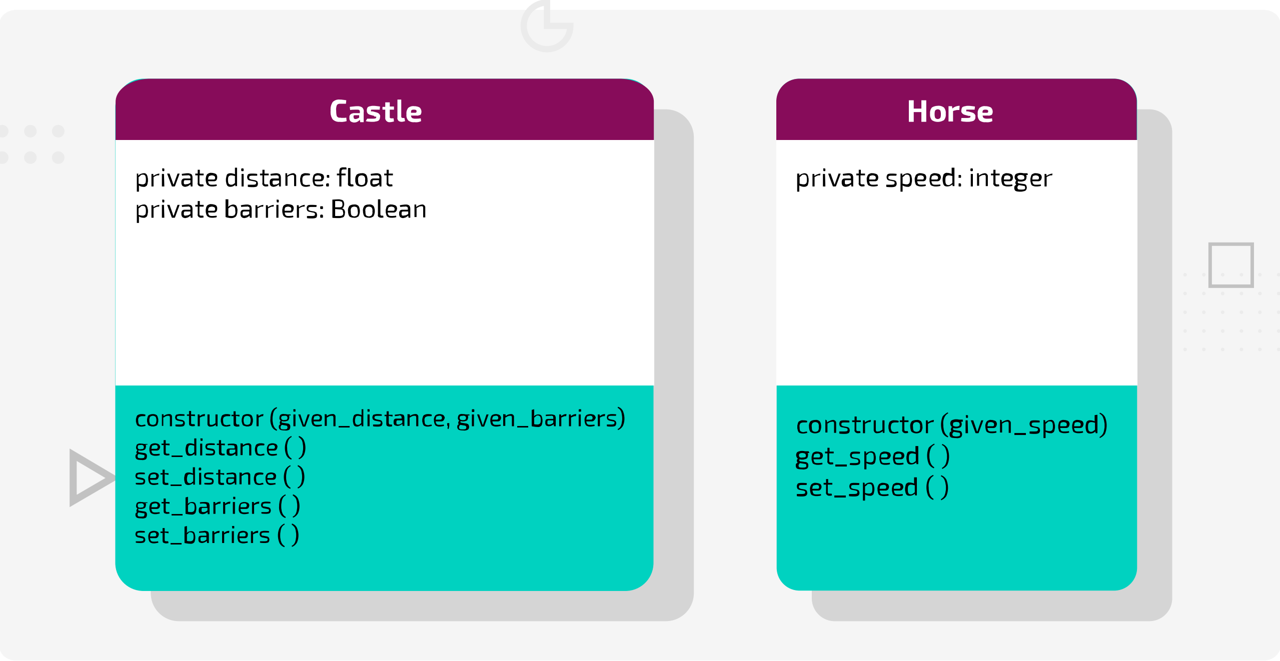
○ `planets[count].get_habitable() == True`

OOP: complete code 4

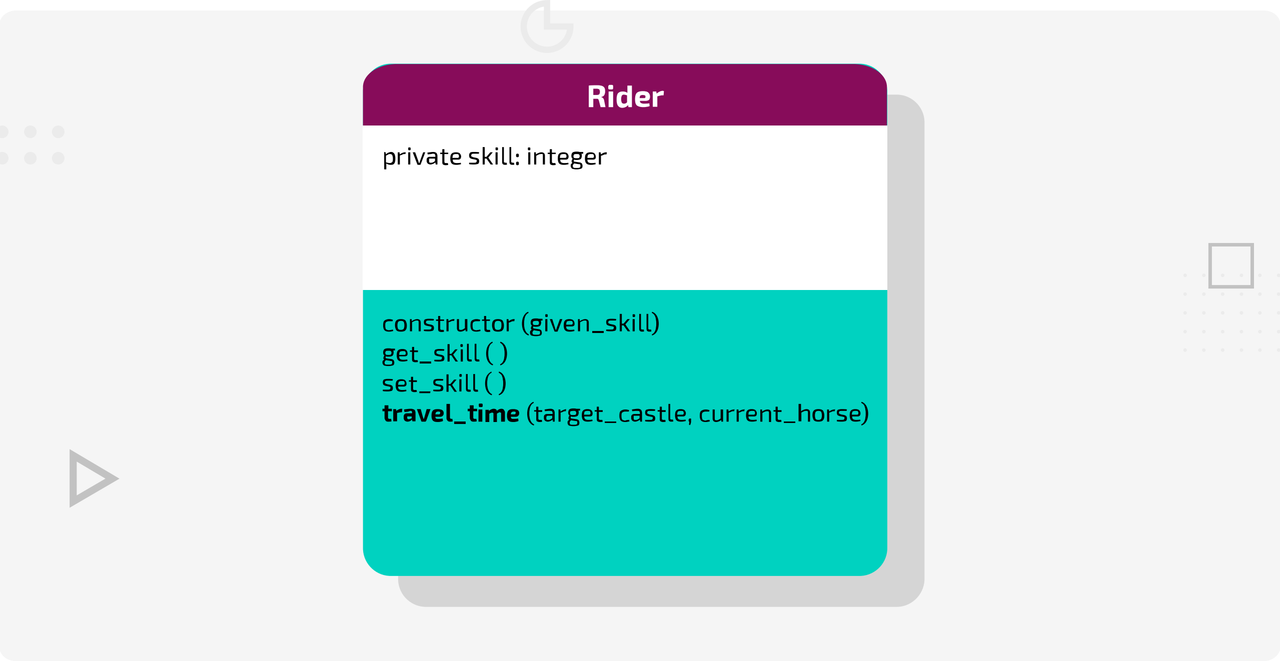
A Level

C C

Maxime is programming a horse riding simulation where a rider travels to various historic castles in the UK with different breeds of horses. Each breed of horse can travel at a different speed, for example, a Thoroughbred can travel at 88 km/h, a QuarterHorse at 70 km/h, and an Appaloosa at 65 km/h. She has created three classes for her program.



Castle and Horse classes


Rider
private skill: integer
constructor (given_skill)
get_skill ( )
set_skill ( )
travel_time (target_castle, current_horse)

Rider class

The travel_time method of the Rider class has the following functionality:

- If the skill of the person travelling is greater than 8 and there are no barriers on the way to the castle, then the **time** required for travel is calculated as the distance to the castle divided by the speed of the horse
- In any other scenario, it takes double the amount of time to reach the castle

Enter the statement that completes the code under the IF statement.

Pseudocode

```
PUBLIC FUNCTION travel_time(target_castle, current_horse)
    IF skill > 8 AND target_castle.get_barriers() == False
        time_required =  ........................................ // Missing code
    ELSE
        // It takes double the time
    ENDIF

    RETURN time_required
ENDFUNCTION
```

Based on the implementation of the travel_time method, what is the time required to reach the castle if the below set of objects are used in the simulation? Give your answer as a number.

Pseudocode

```
    my_castle = new Castle(130, True)

    my_appaloosa = new Horse(65)

    my_rider = new Rider(9)
```

OOP: complete code 7

GCSE A Level
P P P P

The following class has been defined using pseudocode.

Pseudocode

```
CLASS Radio
    PRIVATE volume: integer
    PRIVATE station: string
    PRIVATE on: Boolean

    PUBLIC PROCEDURE Radio(given_station)
        station = given_station
        volume = 3
        on = False
    ENDPROCEDURE

    PUBLIC FUNCTION get_volume()
        RETURN volume
    ENDFUNCTION

    PUBLIC FUNCTION get_station()
        RETURN station
    ENDFUNCTION

    PUBLIC FUNCTION is_on()
        RETURN on
    ENDFUNCTION

    PUBLIC PROCEDURE set_volume(new_volume)
        volume = new_volume
    ENDPROCEDURE

    PUBLIC PROCEDURE set_station(new_station)
        station = new_station
    ENDPROCEDURE

    PUBLIC PROCEDURE switch()
        IF on == True THEN
            on = False
        ELSE
            on = True
        ENDIF
    ENDPROCEDURE

ENDCLASS
```

An object has been created based on the class definition shown above and assigned to a reference variable called `my_radio`. Which of the following options (expressed in pseudocode) will set the volume of this radio object to 6?

○ `my_radio.set_volume() = 6`

○ `my_radio.volume = 6`

○ `my_radio.set_volume(6)`

Encapsulation

A Level

P P

Select one statement that describes why the principle of **encapsulation** is important for the design of OOP programs.

○ It ensures that child classes can take the attributes and methods of the parent class.

○ It ensures that any interaction with an object, specifically the manipulation of its data, is only allowed via its public interface.

○ It allows the methods of child classes to behave in different ways to those inherited from a parent class.

○ It enables the creation of objects with specific states and behaviour.