Advanced Higher Computing Science

Software design and development: polymorphism in object-oriented programming workshop materials

This edition: December 2024 (version 1.0)

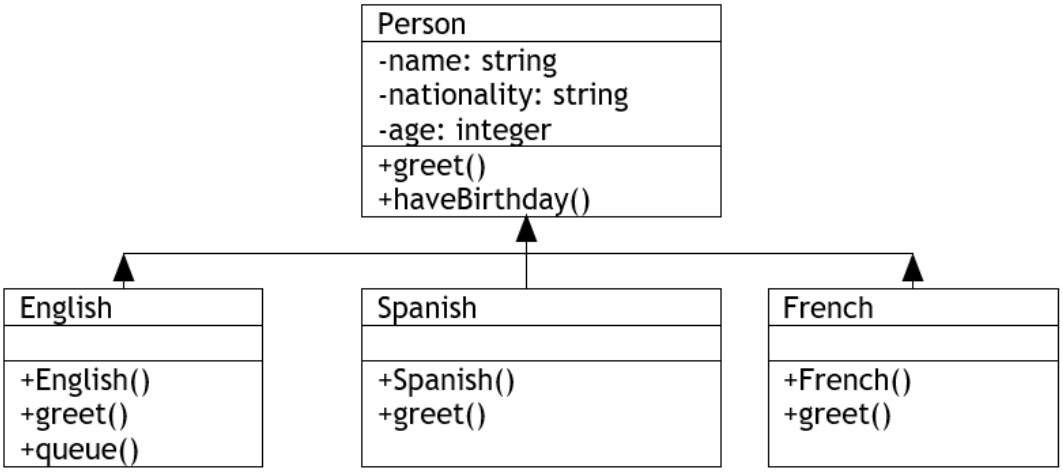
Introduction

This document is for teachers and lecturers and/or Advanced Higher Computing Science candidates.

This document contains polymorphism in object-oriented programming questions and accompanying marking instructions devised for workshops at Understanding Standards events held in 2019 (questions 3 and 4) and 2023 (questions 1 and 2).

Questions

1. A `Person` class has the method `greet()`. The class has three subclasses `English`, `Spanish` and `French` as shown in the UML class diagram. The `haveBirthday()` method is used to add one to the `age` stored, while the `queue()` method adds two years to the stored `age`.



The code used to implement the `greet()` method in each class is shown below:

```
Class Person
PROCEDURE greet()
  SEND "I am " & THIS.age TO DISPLAY
END PROCEDURE
```

```
Class English
OVERRIDE PROCEDURE greet()
  SEND "Hello!" TO DISPLAY
END PROCEDURE
```

```
Class Spanish
OVERRIDE PROCEDURE greet()
  SEND "¡Hola!" TO DISPLAY
END PROCEDURE
```

```
Class French
OVERRIDE PROCEDURE greet()
  SEND "Bonjour!" TO DISPLAY
END PROCEDURE
```

An object of each subclass is instantiated and these are assigned to an array of `Person` objects called `student`, as follows:

```
DECLARE english1 INITIALLY English("Adam", "British", 16)
DECLARE spanish1 INITIALLY Spanish("Jose", "Spanish", 17)
DECLARE french1 INITIALLY French("Yvette", "French", 15)
SET student[0] TO english1
SET student[1] TO spanish1
SET student[2] TO french1
```

Write down the output produced when the following code is executed. You should explain any errors that may be generated.

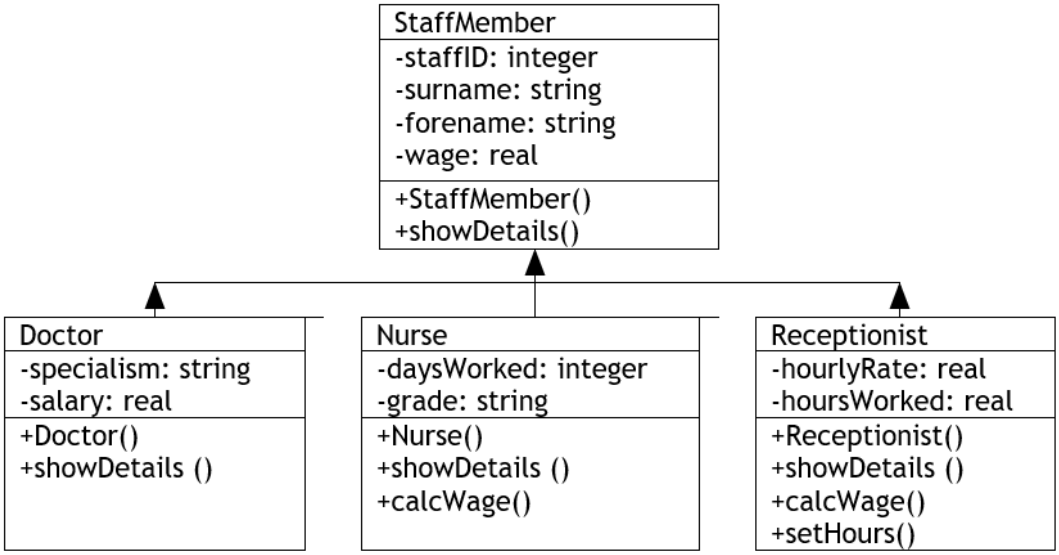
(a) `spanish1.greet()`

(b) `student[1].greet()`

(c) `student[2].greet()`

(d)
```
student[2].haveBirthday()
student[2].greet()
```

2. An object-oriented program is used to process details of staff working in the WellScot Health Centre. The UML class diagram below shows some of the information that is stored.



(a) Use examples from the class diagram to explain the term 'inheritance'.

(b) Dr Connor is a doctor at the WellScot Health Centre. Explain why her attributes include `surname` but do not include `hourlyRate`.

(c) Cleaning staff who work at the WellScot Health Centre should have been included in the class diagram. A cleaner has the additional attributes `hourlyRate` and `hoursWorked` with the constructor `Cleaner()` and methods `calcWage()` and `setHours()`. Add these details to the class diagram.

(d) A new cleaner has started at WellScot Health Centre. The following code is executed:

```
DECLARE cleaner1 INITIALLY Cleaner(1234, "Malek", "Ali",
0.00, 11.53, 0.00)
```

Use this example to explain the terms 'class' and 'object'.

(e) The code for the `showDetails()` method of the `StaffMember` class is shown below.

```
PROCEDURE showDetails()
   SEND "I work at the WellScot Health Centre" TO DISPLAY
END PROCEDURE
```

When the code `doctor1.showDetails()` is executed, the program produces the following output:

```
I am a doctor at the WellScot Health Centre
```

Using a programming language of your choice, write the code for the `showDetails()` method of the `Doctor` class.

(f) The WellScot Health Centre uses an array of `StaffMember` objects called `staff` to keep track of all the staff who currently work there.

Index	Object Indexed
0	doctor1
1	doctor2
2	nurse1
3	receptionist1
4	receptionist2
5	cleaner1
6	doctor3
7	cleaner2
8	nurse2

Write down the output produced when the following code is executed. If an error is generated, you should explain why the error has occurred.

(i) `staff[0].showDetails()`

(ii) `staff[2].calcWage()`

(iii) `staff[3].setHours()`

(iv) `staff[7].showDetails()`

3. An object-oriented program makes use of a `Vehicle` class to store details of a company's vehicles.

The `Van` class is a subclass of the `Vehicle` class with additional instance variables:

- `capacity` that represents the maximum load space measured in litres
- `tailLift` that represents whether the van has a tail lift or not

The class definition for the `Vehicle` class has been provided below. You should note that each of the three instance variables are private variables.

```
CLASS Vehicle IS { STRING regNumber, STRING make, STRING
colour }

METHODS

      PROCEDURE Vehicle(STRING reg, STRING mke, STRING col)
            DECLARE THIS.regNumber INITIALLY reg
            DECLARE THIS.make INITIALLY mke
            DECLARE THIS.colour INITIALLY col
      END PROCEDURE

      PROCEDURE setColour(STRING col)
            SET THIS.colour TO col
      END PROCEDURE

      PROCEDURE purpose()
            SEND "I carry passengers" TO DISPLAY
      END FUNCTION

      FUNCTION getMake() RETURNS STRING
            RETURN THIS.make
      END FUNCTION

END CLASS
```

(a) (i) Use the OO terms ‘instantiation’ and ‘inheritance’ to explain the purpose of the statement:

```
DECLARE vehicle1 AS Van INITIALLY ("ABC 123D", "Ford",
"white", 50, false)
```

2

(ii) The statement `vehicle1.purpose()` should produce the message "I carry cargo".

Explain how polymorphism would apply in this situation. **2**

(iii) Using a programming language of your choice, write the class definition for the `Van` class. In addition to the constructor and `purpose`() methods, this class should provide getter methods that enable external code to access the values stored in its two instance variables. **4**

(iv) Use appropriate object-oriented terminology to explain why the following statement is invalid.

```
SET vehicle1.regNumber TO "XYZ 987W"
```

2

(b) Draw a UML class diagram to represent the structure of the `Vehicle` and `Van` classes. **3**

(c) An array of `Van` objects called `vanDetails` is used to store details of the 25 vans in the company fleet. The following statement in the main program is used to activate the function `count()`.

```
SET numberFordVans TO count(vanDetails)
```

This function is used to calculate and return the number of Ford vans in the company fleet.

Using a programming language of your choice, write the code for this `count()` function. **3**

Note: teachers and lecturers can extend this question to include a question that requires candidates to use the find maximum algorithm.

4. A `Player` object is defined by the `Player` class shown below.

```
CLASS Player { STRING name, INTEGER score, STRING location }

METHODS

    CONSTRUCTOR Player(STRING nme, INTEGER scr, STRING loc)
          DECLARE THIS.name INITIALLY nme
          DECLARE THIS.score INITIALLY scr
          DECLARE THIS.location INITIALLY loc
    END CONSTRUCTOR

    FUNCTION getName() RETURNS STRING
          RETURN THIS.name
    END FUNCTION

    FUNCTION getScore() RETURNS INTEGER
          RETURN THIS.score
    END FUNCTION

END CLASS
```

(a) Describe the purpose of the constructor method shown in the class definition code above. **2**

An array of `Player` objects called `topTen` contains the names and scores of the 10 highest-scoring players in an online computer game. These details are stored in descending order of score.

(b) The first three members of the array `topTen` are shown below.

topTen

[0]			[1]			[2]		
Jo	964	Ayr	Pete	900	York	Sofia	840	Rome

State the value returned by:

(i) `topTen[1].getName()` **1**

(ii) `topTen[2].getScore ()` **1**

(c) At the end of each game, a new `Player` object called `newPlayer` is created.

The method `compare()` receives the `newPlayer` object containing the player's details and the `topTen` array of `Player` objects.

The method compares the new player's score with those in the `topTen` array. It returns the position where the new score should be inserted in the `topTen` array, or the value -1 if the new score is not high enough to be included.

The method `compare()` has been started below:

```
FUNCTION compare(Player newPlayer, ARRAY OF Player
topTen) RETURNS INTEGER
   ## lines of code missing


END FUNCTION
```

Write the missing code for this `compare()` method. **4**

(d) A method `calculateAverage()` is used to calculate and return the average of the scores stored in the `topTen` array. This function is activated in the main program using the statement:

```
SET averageScore TO calculateAverage(topTen)
```

Write the code for this `calculateAverage()` method. **3**

Note: teachers and lecturers can extend this question to include a question that requires candidates to use the linear search algorithm.

Marking instructions

1. (a) ¡Hola!

(b) ¡Hola!

(c) Bonjour!

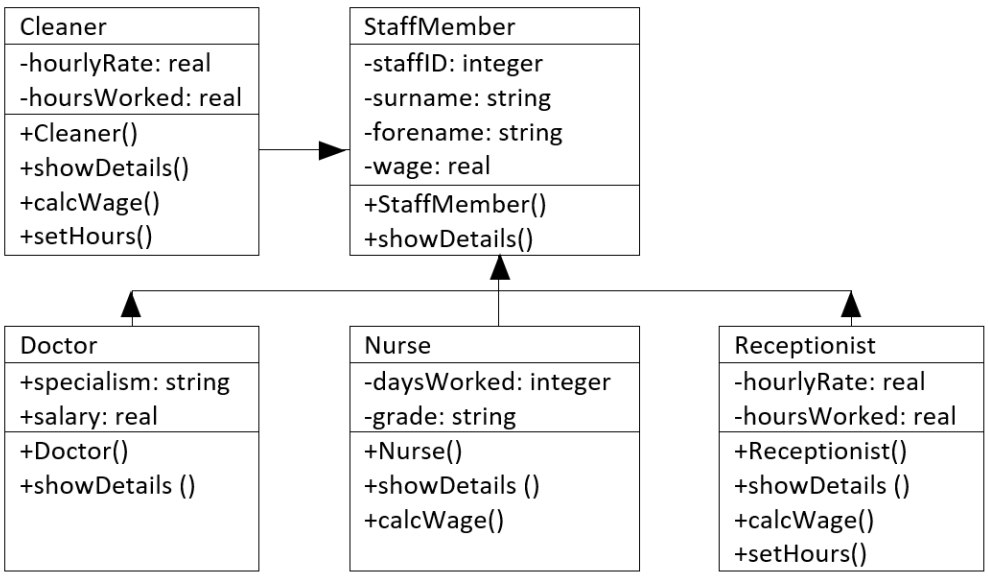
(d) Bonjour!

2. (a) Inheritance is the feature of OO programming that allows one class to inherit the properties and methods from another class. The class doing the inheriting is a subclass and the class whose properties and methods are inherited is the superclass.

For example, the subclass `Nurse` inherits all of the properties and methods of the superclass `StaffMember`. This means that all objects that belong to the Nurse class will have seven properties (`staffID`, `surname`, `forename`, `wage`, `daysWorked` and `grade`) and three methods (`Nurse()`, `showDetails()` and `showDetails()`).

(b) The details of Dr Connor will be stored in an object that belongs to the `Doctor` class. In addition to the properties of its own class (`specialism` and `salary`), the property `surname` will be inherited from the superclass `StaffMember`. However, the property `hourlyRate` is associated with the subclass `Receptionist` and cannot be access by `Doctor` type objects.

(c)



(d) A class provides a detailed description that is the blueprint or template used to create an object.

An object stores the data values that define one specific example or instance of that class.

In this example, the code is used to create an object called `cleaner1`, which belongs to the `Cleaner` class.

(e)
```
OVERRIDE PROCEDURE showDetails()
      SEND "I am a doctor at the WellScot Health Centre" TO
      DISPLAY
END PROCEDURE
```

(f) (i) I am a doctor at the WellScot Health Centre

(ii) This code will generate an error.

The array element `staff[2]` indexes the object `nurse1`, which belongs to the subclass `Nurse`. Although the method `calcWage()` is associated with the `Nurse` subclass, the array of `StaffMember` objects `staff` is acting as a `StaffMember` type wrapper that hides the subclass characteristics of the object being indexed. Since the superclass is not associated with the `calcWage()` method, this attempt to apply the method to a `StaffMember` type object will generate an error.

(iii) This code will generate an error.

The array element `staff[3]` indexes the object `receptionist1`, which belongs to the subclass `Receptionist`. Although the method `setHours()` is associated with the `Receptionist` subclass, the array of `StaffMember` objects `staff` is acting as a `StaffMember` type wrapper that hides the subclass characteristics of the object being indexed. Since the superclass is not associated with the `setHours()` method, this attempt to apply the method to a `StaffMember` type object will generate an error.

(iv) I work at the WellScot Health Centre

3. (a) (i) A new object called `vehicle1` has been instantiated. This object belongs to the `Van` class. Since `Van` is a subclass of the `Vehicle` class, `vehicle1` inherits each instance variable and method belonging to the `Vehicle` superclass. The values provided in the `DECLARE` statement are assigned, in the sequence listed, to the three instance variables inherited from the `Vehicle` class and the additional two instance variables belonging to the `Van` class.

Award 1 mark for an explanation of instantiation that makes reference to the code provided.

Award 1 mark for an explanation of encapsulation that makes reference to the code provided.

(ii) The `vehicle1` object inherits the `purpose()` method from the `Vehicle` superclass. Since the output from this inherited method differs from the output that is required, polymorphism must be used to redefine the `purpose()` method for the `Van` subclass. In this way, the `purpose()` method for the `Van` subclass overrides the inherited method thereby allowing all `Van` objects to respond differently.

Award 1 mark for an explanation of inherited method.

Award 1 mark for an explanation of the need to use polymorphism to override the inherited method to alter its behaviour.

(iii)
```
CLASS Van INHERITS Vehicle WITH { REAL capacity,
BOOLEAN tailLift }

METHODS
    PROCEDURE Van(REAL cap, BOOLEAN tail)
          DECLARE THIS.capcity INITIALLY cap
          DECLARE THIS.tailLift INITIALLY tail
    END PROCEDURE

    OVERRIDE PROCEDURE purpose()
          SEND "I carry cargo" TO DISPLAY
    END FUNCTION

    FUNCTION getCapacity() RETURNS REAL
          RETURN THIS.capacity
    END FUNCTION

    FUNCTION getTailLift() RETURNS BOOLEAN
          RETURN THIS.tailLift
  END FUNCTION
END CLASS
```

Award 1 mark for code that indicates inheritance from the `Vehicle` class with two additional instance variables.

Award 1 mark each for constructor and `purpose()` methods.

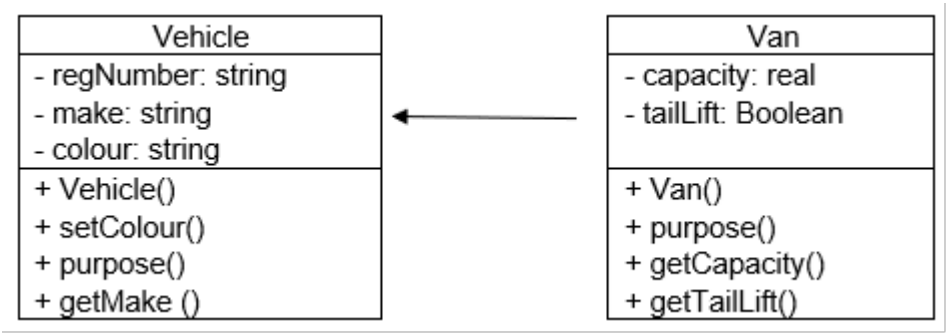
Award 1 mark for both getter methods.

(iv) The instance variable `regNumber` is private, meaning that it is encapsulated. To access the value stored in the variable and edit its contents, a method must be used: the instance variable `regNumber` cannot be edited directly.

Award 1 mark for an explanation that refers to encapsulation of the variable.

Award 1 mark for an explanation that refers to the need to use a method.

(b)



Award 1 mark for correct instance variables and methods of `Vehicle` class.

Award 1 mark for correct instance variables and methods of `Van` class.

Award 1 mark for correct indication of inheritance.

(c)

```
FUNCTION count (ARRAY OF Van vanDetails) RETURNS INTEGER
  SET total TO 0
  FOR index FROM 0 TO 24 DO
   IF vanDetails[index].getMake() ="Ford" THEN
    SET total TO total + 1
   END IF
  END FOR
  RETURN total
END FUNCTION
```

Award 1 mark for correct use of `vanDetails` array.

Award 1 mark for correct use of `getMake()` method.

Award 1 mark for correct processing of array total.

4. (a) When it is invoked, this constructor method is used to instantiate a new object that belongs to the `Player` class. The values provided by the user are assigned to this new object's three instance variables `name`, `score` and `location`.

Award 1 mark for an explanation that refers to instantiation of an object.

Award 1 mark for an explanation that refers to assignment of values to instance variables.

(b) (i) Pete
(ii) 840

Award 1 mark each.

(c)
```
FUNCTION compare (Player newPlayer, ARRAY OF Player topTen)
RETURNS INTEGER
  SET index TO 0
  SET include TO false
  SET position TO -1
   REPEAT UNTIL include = true OR index = 10
    IF newPlayer.getScore() > topTen[index].getScore()THEN
      SET include TO true
      SET position TO index
    END IF
   END REPEAT
   SET index TO index + 1
   RETURN position
END FUNCTION
```

Award 1 mark for correct use of `topTen` **array.**

Award 1 mark for correct use of `getScore()` **method.**

Award 1 mark for traversing `topTen` **array.**

Award 1 mark for correctly recording insertion position.

(d)
```
FUNCTION calculateAverage (ARRAY OF Player topTen) RETURNS
REAL
    SET result TO 0.0
    FOR index FROM 0 TO 9
       SET result TO result + topTen[index].getScore()
    END FOR
    SET result TO result/10
    RETURN result
END FUNCTION
```

Award 1 mark for correct use of `topTen` **array.**

Award 1 mark for correct use of `getScore()` **method.**

Award 1 mark for correct processing of array average.