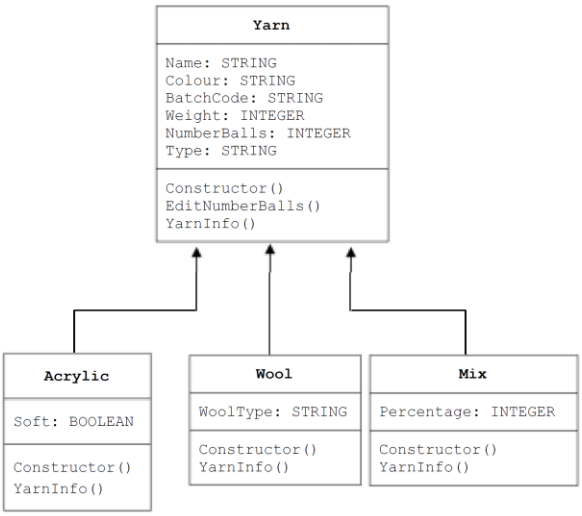
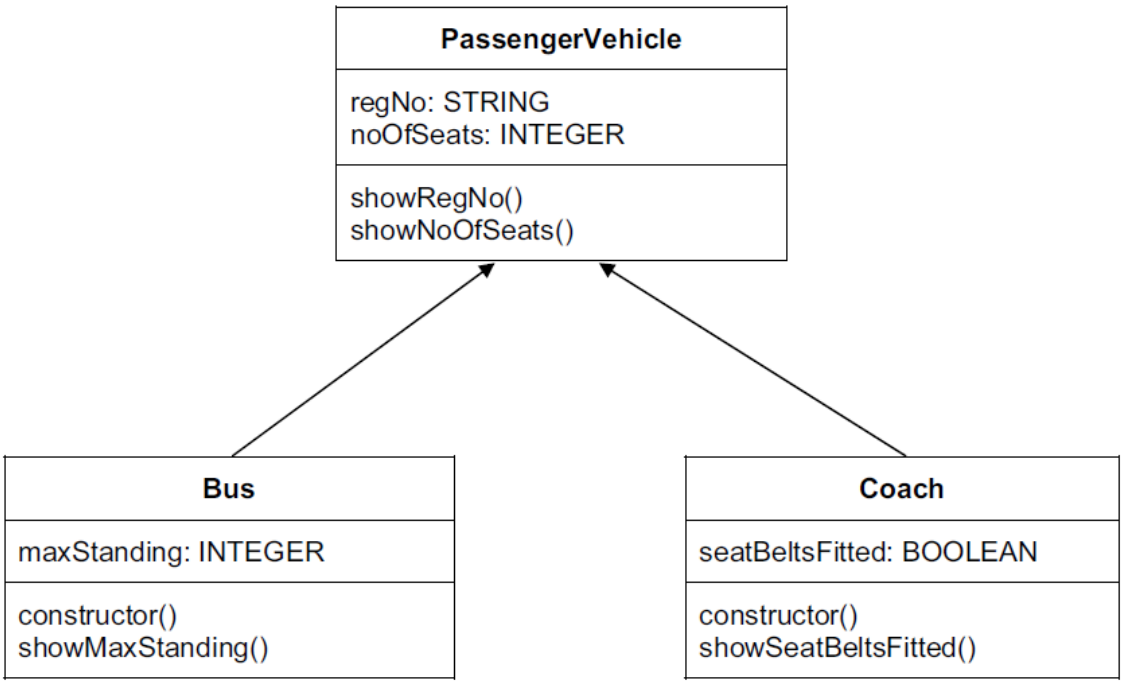
Question No. 1

7(a)	**1 mark** per point
	`Acrylic` has attribute `Soft` of type `BOOLEAN`
	`Wool` has attribute `WoolType` with suitable data type
	`Acrylic` **and** `Wool` have method `YarnInfo()`
	`Acrylic`, `Wool` and `Mix` **at least one** inherit (one arrow correct) from `Yarn` …
	… `Acrylic`, `Wool` and `Mix` **all** inherit (all arrows correct) from `Yarn`
	 
	
		5

7(b)	Properties **max 2**: • the data items / attributes • the data types // characteristics • defined in a class Methods **max 2**: • the procedures/ functions / programmed instructions in a class / super class / base class • … implementing the behaviours • … that act on the properties / attributes Inheritance **max 2**: • Methods and properties / attributes contained in one class/ super class / base class • Are made available to / reused by another class/ derived class	**6**

Question No. 2

(a)



Mark as follows:

noOfSeats declaration and associated show method in PassengerVehicle	[1]
inheritance arrows	[1]
constructor method in Coach	[1]
seatBeltsFitted declaration and associated show method in Coach	[1]

(b) e.g. in Python:

```
class PassengerVehicle():
    def __init__(self, regNo, noOfSeats):
        # Sets all the initial values
        self.__ regNo = regNo
        self.__ noOfSeats = noOfSeats

    def showRegNo(self):
        print("Registration No: ",self.__regNo)

    def showNoOfSeats (self):
        print("No of seats: ",self.__noOfSeats)
```

Mark as follows:

data declarations	[1]
use of __ in identifiers to give “private” attribute	[1]
use of ‘self’ parameter	[1]
showRegNo function	[1]
showNoOfSeats function	[1]

(c) e.g. in Python:

```
class Bus(PassengerVehicle):
    def __init__(self, regNo,
 noOfSeats, maxStanding):
        super().__init__(regNo, noOfSeats)
        self.__maxStanding = maxStanding

    def showMaxStanding (self):
        print("No of standing passengers: ", self.__maxStanding)
```

Mark as follows:

inheritance	[1]
__init__ function header	[1]
use of __init__ from superclass	[1]
initialisations in __init__ function	[1]
showMaxStanding function	[1]

(d) **(i)** e.g. in Python:

```
pv1 = Bus("NBR 123", 51,10)
```

[1]

(ii) e.g. in Python:

```
pv1.showRegNo()
```

[1]

```
pv1.showNoOfSeats()
```

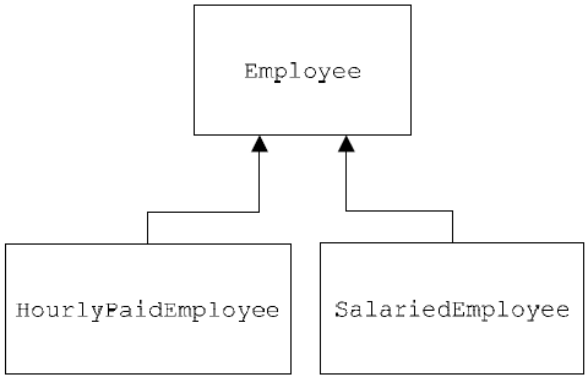
[1]

```
pv1.showMaxStanding()
```

[1]

Question No. 3

(a)



[3]

(b)

Example Python

```
Class Employee():
    def __init__(self):
        self.__EmployeeName = ""
        self.__EmployeeID = ""
        self.__AmountPaidThisMonth = 0
    def SetEmployeeName(self, Name):
        self.__EmployeeName = Name
    def SetEmployeeID(self, ID):
        self.__EmployeeID = ID
    def SetAmountPaidThisMonth(self, Paid):
        self.__AmountPaidThisMonth = Paid
```

[max 5]

(c) **(i)**

HoursWorked	1	
HourlyPayRate	1	
SetHoursWorked	1	
CalculatePay : Override	1 + 1	
SetPayRate	1	**[max 4]**

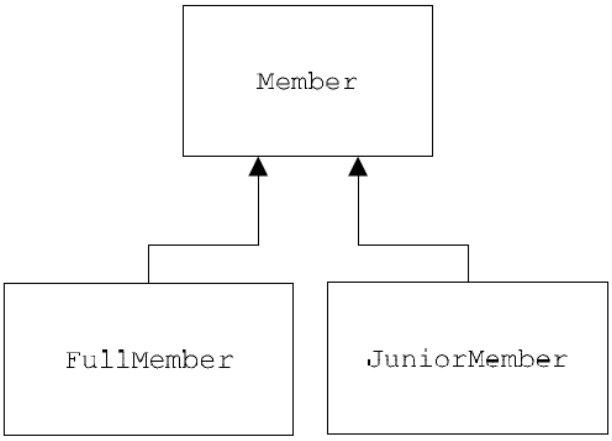
(ii)

AnnualSalary	1	
SetSalary	1	
CalculatePay : Override	1	**[max 2]**

(d) Polymorphism **[1]**

Question No. 4

(a)



[3]

(b)

Example Python

```
class Member() :
    def __init__(self):          PUBLIC
        self.__MemberName = ""
        self.__MemberID = ""
        self.__SubscriptionPaid = False
    def SetMemberName(self, Name):
        self.MemberName = Name
    def SetMemberID(self, ID):
        self.MemberID = ID
    def SetSubscriptionPaid(self, Paid):
        self.SubscriptioPaid = Paid
```

Mark as follows:

Class header	(1 mark)	
`Public` and `Private` used correctly	(1 mark)	
`MemberName` + `MemberID`	(1 mark)	
`SubscriptionPaid`	(1 mark)	
Methods × 3	(1 mark)	**[5]**

(c) (i)

Example Python

```
class JuniorMember (Member):
    def __init__self:
        super().__init__()
        self.DateOfBirth = ""
    def SetDateOfBirth(self, Date):
        self.DateOfBirth = Date
    def SetMemberName(self, Name):
        super().SetMemberName(Name)
    def SetMemberID(self, ID):
        super().SetMemberID(ID)
    def SetSubscriptionPaid(self, Paid):
        super().SetSubscriptioPaid(Paid)
```

[3]

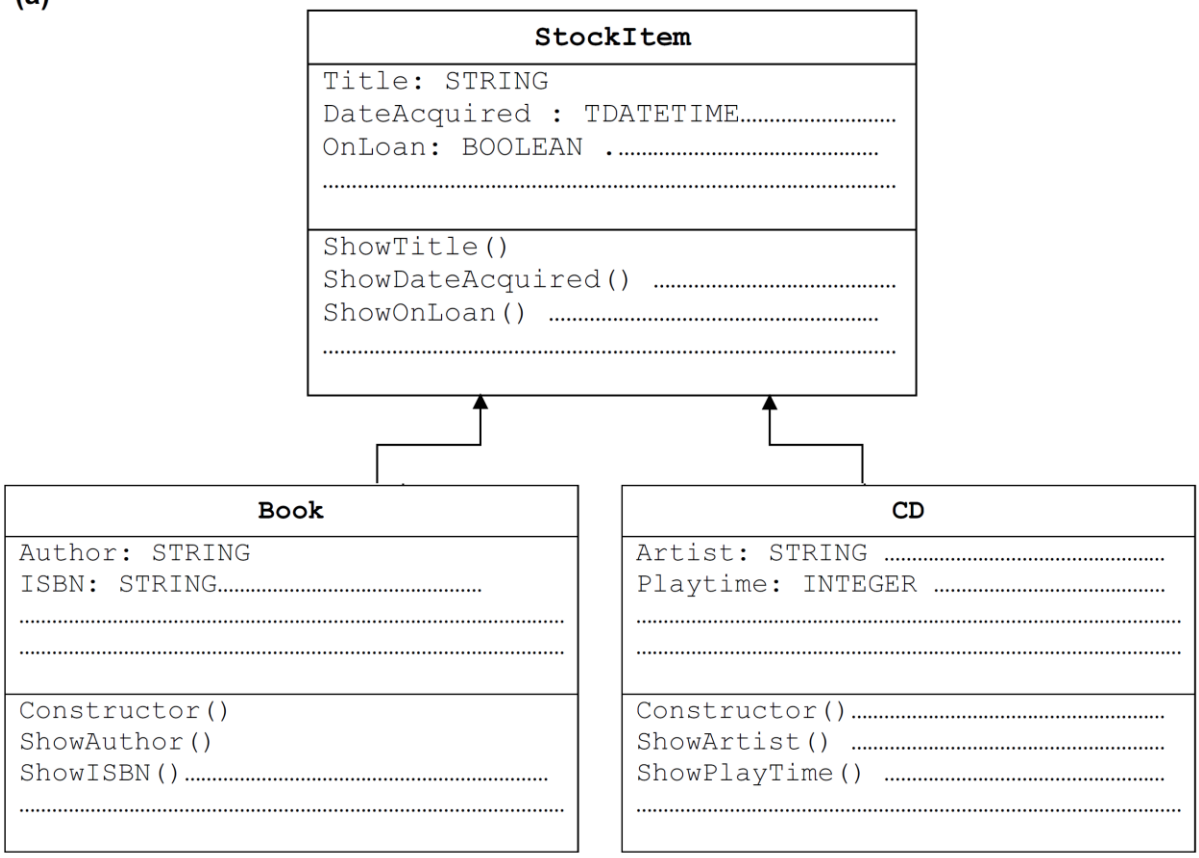
(ii)

Example Python

```
NewMember := JuniorMember()                          (1 mark)
NewMember.SetMemberName("Ahmed")
NewMember.SetMemberID("12347")                       (1 mark)
NewMember.SetSubscriptionPaid(TRUE)
NewMember.SetDateOfBirth("12/11/2001")               (1 mark)    [3]
```

Question No. 5

(a)



[max. 7]

(b) (i) *Mark as follows:*
Class header
Methods
Properties

Python

```
class StockItem :
    def __int__(self) :
        self.__Title = ""
        self.__DateAquired = ""
        self.__OnLoan = False

    def ShowTitle() :
        pass
    def ShowDateAcquired() :
        pass
    def ShowOnLoan() :
        pass
```

(ii) *Mark as follows:*
Class header and showing superclass
Methods
Properties

Python

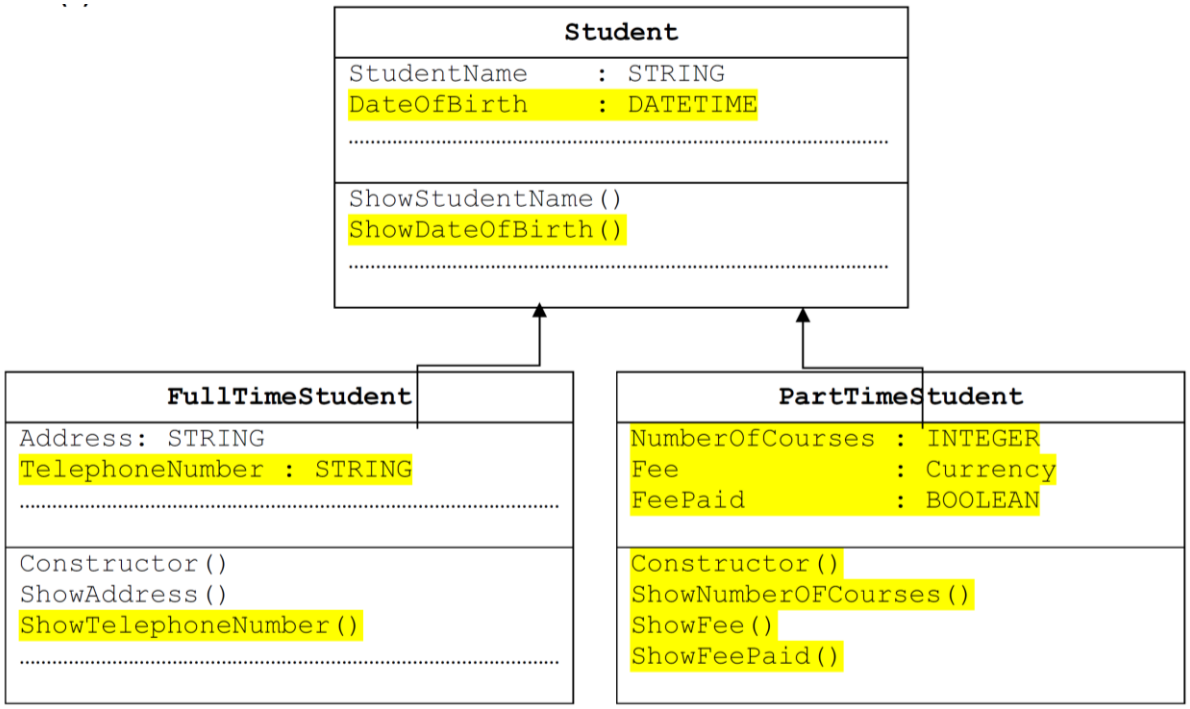
```
class Book(StockItem) :
    def __init__(self) :
        self.__Author = ""
        self.__ISBN = ""
    def ShowAuthor() :
        pass
    def ShowISBN() :
        pass
```

(iii)

Python

```
NewBook = Book()                                    1
NewBook.Title = "Computers"
NewBook.Author = "A.Nyone"
NewBook.ISBN = "099111"                             1
NewBook.DateAcquired = "12/11/2001"
NewBook.OnLoan = False                              1
```

Question No. 6



Mark as follows:

Base class:
- dateOfBirth declaration and associated method in Student
- constructor

Subclasses:
- telephoneNumber declaration and associated method in FullTimeStudent
- NumberOFCourses declaration and associated method in PartTimeStudent
- fee declaration and associated method in PartTimeStudent
- feepaid declaration and associated method in PartTimeStudent
- constructor method in PartTimeStudent
- inheritance arrows

Ignore data types, ignore other methods/attributes
Ignore brackets after methods [Max 7]

(b) (i) Mark as follows (parts to be ignored in grey):

If no programming language stated, map to 1 of the 3 below (or check in Q1ai)
Class header & ending (watch out these may be squashed into the next clip)
Ignore methods
2 attributes with correct data types
No mark if subclass properties shown here
Attributes required:
StudentName
DateOfBirth (accept variations e.g. DoB)

Python

```
class Student :
    def __int__(self) :
        self.__StudentName = ""
        self.__DateOfBirth = ""  # date(1,1,2015)
    def ShowStudentName() :
        pass
    def ShowDateOfBirth() :
        pass
```

Ignore __ before attributes [2]

(ii) Mark as follows:

- Class header and showing superclass
- Properties (Do not award this mark if properties from base class included here)
  Data types must be correct
- Methods (Do not award this mark if methods from base class included here)
  must show heading and ending of procedure/function declaration
  Ignore PUBLIC, PRIVATE

Python

```
class FullTimeStudent(Student) :
    def __init__(self) :
        self.__Address = ""
        self.__TelephoneNumber = ""
    def ShowAddress() :
        pass
    def ShowTelephoneNumber() :
        pass
```

[3]

(iii) 1 mark per statement to max 3
Missing string delimiters: penalise once
Accept use of constructor

Python

```
NewStudent = FullTimeStudent()
NewStudent.StudentName = "A.Nyone"
NewStudent.DateOfBirth = "12/11/1990"
NewStudent.TelephoneNumber = "099111"
```

Alternative

```
NewStudent = FullTimeStudent('A.Nyone', '12/11/1990', '099111')
```

(c) (i) 1 mark per bullet to max 2: [2]

- The attributes can only be accessed in the class
- Properties are needed to get/set the data // It provides/uses encapsulation
- Increase security/integrity of attributes

(ii) 1 mark per bullet [2]

- The public methods can be called anywhere in the main program // Public methods can be inherited by sub-classes

- The private methods can only be called within the class definition // cannot be called outside the class definition // Private methods cannot be inherited by sub-classes

Question No. 7

(i) 1 mark per bullet to max 3 **[3]**

- Method header
- initialising Code to ""
- initialising State to "Open-NoCode"

e.g.

PYTHON:

```
def __init__(self):
    self.__code = ""
    self.__state = "Open-NoCode"
```

(ii) 1 mark per bullet to max 2 **[2]**

- method header
- Setting code to ""

e.g.

PYTHON:

```
def reset(self):
    self.__code = ""
```

(iii) 1 mark per bullet to max 2 **[2]**

- method header with parameter
- setting state to parameter value
- Outputting state

e.g.

PYTHON:

```
def SetState(self,NewState):
    self.__state = NewState
    print(self.__state)
```

(iv) 1 mark per bullet to max 2 **[2]**

- setting code to parameter
- Outputting New cost set and code

e.g.

PYTHON:

```
def SetNewCode(self, NewCode):
    self.__code = NewCode
    print("New code set: ", self.__code)
```

(v) 1 mark per bullet to max 4 **[4]**

- function header taking string parameter, returns Boolean
- check length of string is 4
- check each character is a digit
- return of correct Boolean value for both cases

e.g

PYTHON:

```
def __valid(self, s):
    digits = ['0','1','2','3','4','5','6','7','8','9']
    isValid = False
    if (len(s) == 4):
        if (s[0] in digits) & (s[1] in digits) & (s[2] in digits) &
        (s[3] in digits):
            isValid = True
    return(isValid)
```

(vii) **1 mark per bullet to max 4** **[4]**

- method header
- Initialising ThisSafe to instance of SafetyDepositBox
- Loop forever
- Call method StateChange on ThisSafe

e.g.

PYTHON:

```
def main():
    ThisSafe = SafetyDepositBox()
    while True:
        ThisSafe.StateChange()
```

(c) **(i)** 1 mark per bullet to max 2: **[2]**

- The attributes can only be accessed in the class
- Properties are needed to get/set the data // It provides/uses encapsulation
- Increase security/integrity of attributes

(ii) 1 mark per bullet **[2]**

- The public methods can be called anywhere in the main program // Public methods can be inherited by sub-classes
- The private methods can only be called within the class definition // cannot be called outside the class definition // Private methods cannot be inherited by sub-classes

Question No. 8

(b) (i) 1 mark per bullet to max 3: [3]

- method header and close
- initialising amount to 0
- initialising state to "Idle"

e.g.

PYTHON:

```
def __init__(self):
    self.__amount = 0
    self.__state = "Idle"
```

(ii) 1 mark per bullet to max 2: [2]

- method header, close with parameter
- setting state to parameter value
- outputting state

e.g.

PYTHON:

```
    def SetState(self, NewState):
        self.__State = NewState
        print(self.__State)
```

(iii) 1 mark per bullet to max 2: [2]

- output Amount
- set amount to zero

e.g.

PYTHON:

```
    def ReturnCoins(self):
        print(self.__Amount)
        self.__Amount = 0
```

(iv) 1 mark per bullet to max 3: [3]

- function header, take string as parameter, return Boolean
- check the parameter is a valid coin
- return of value for both cases

e.g.

PYTHON:

```
    def __validCoin(self, s):
        coins = ['10','20','50','100']
        if s in coins:
            isValid = True
        else:
            isValid = False
        return(isValid)
```

(v) 1 mark per bullet to max 2 [2]

- Cast parameter as integer
- Add value to amount

e.g.

PYTHON:

```
def coinInserted(self, s):
    coinValue = int(s)
    self.__amount = self.__amount + coinValue
```

(vii) 1 mark per bullet to max 4 [4]

- declaration of main method header
- Initialising ParkingMeter as instance of TicketMachine
- Looping while true/until false
  - Calling stateChange method on ParkingMeter

e.g.

PYTHON:

```
def main():
    ParkingMeter = TicketMachine()
    while True:
        ParkingMeter.stateChange()
```

(c) (i) 1 mark per bullet to max 2: [2]

- The attributes can only be accessed in the class
- Properties are needed to get/set the data // It provides/uses encapsulation
- Increase security/integrity of attributes

(ii) 1 mark per bullet [2]

- The public methods can be called anywhere in the main program // Public methods can be inherited by sub-classes
- The private methods can only be called within the class definition // cannot be called outside the class definition // Private methods cannot be inherited by sub-classes

Question No. 9

6(a)	1 mark per bullet to max 6 - Declaring a class with the name animal - Declaring variables for across, down and score (all Integers) - …as private/protected - Correct constructor header and ending - Randomly generating an across between 0–39 inc. in constructor - Randomly generating a down between 0–39 inc. in constructor - Initialising Score to zero in constructor - Correct get for `Across` - Correct set for `Across`	**6**

6(a) or

```
Class Animal
   Private Across As Integer
   Property _Across As Integer
      Get
         Return _Across
      End Get
      Set(Value As Integer)
         Across = Value
      End Set
   End Property
   Private Down As Integer
   Private _Score As Integer
   Sub New()
      Randomize()
      Across = randomnumber.Next(0, 40)
      Down = randomnumber.Next(0, 40)
      _Score = 0
   End Sub
End Class
```

```
Example: Python
class Animal :
   def __init__ (self) :
      x = random.randint(0,39)
      y = random.randint(0,39)
      self.Across = x
      self.Down = y
      self.Score = 0

   def SetAcross(A) :
      self.Across = A

   def GetAcross() :
      return self.Across
```

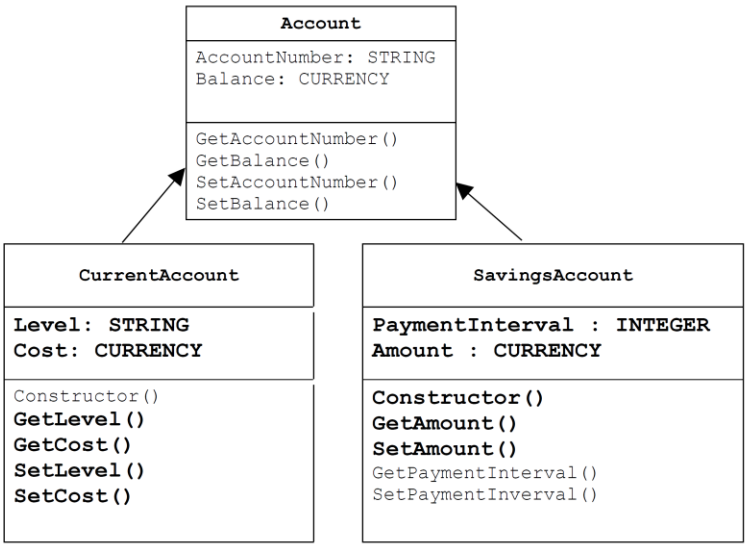
6(b)	1 mark per bullet to max 5
	- constructor method heading and ending
	- Initialise all 40 by 40 elements of Grid as '' or equivalent
	- Loop 5 times…
	- …Creates a new instance of animal inside loop...
	- ...and adds it to array `AnimalList`
	- Call generate food **and** initialise `StepCounter` to 0
	
	Example Python
	`def __init__ (self) :`
	`  self.grid = [[' ' for i in range(40)] for j in range(40)]`
	`  self.AnimalList = []`
	`  self.StepCounter = 0`
	`  for i in range(5) :`
	`  newAnimal = Animal ()`
	`  self.AnimalList.append(newAnimal)`
	`  self.GenerateFood()`
	
	Example VB
	`Sub New()`
	`  For x = 0 To 39`
	`  For y = 0 To 39`
	`  grid(x, y) = ""`
	`  Next`
	`  Next`
	`  For z = 0 To 4`
	`  AnimalList(z) = New Animal`
	`  Next`
	`  Call GenerateFood()`
	`End Sub`
		5

6(c)(i)	1 mark per bullet: - Function header and ending taking one value as parameter - Check if coordinate = 0 (on lower bound) - …generate random number (0 or 1) - Check if coordinate = 39 (on upper bound) - …generate random number (–1 or 0) - Generate random number (e.g. –1, 0, 1) - Return the generated value	**max 4**

6(c)(i)	Example VB
	`Function GenerateDirection(ByRef coord As Integer)`
	`  Dim lowerbound As Integer = -1`
	`  Dim upperbound As Integer = 1`
	`  If coord = 0 Then`
	`  lowerbound = 0`
	`  ElseIf coord = 39 Then`
	`  upperbound = 0`
	`  End If`
	`  GenerateDirection = randomnumber.Next(lowerbound, upperbound)`
	`End Function`
	
	Example Python
	`def GenerateDirection(Coord) :`
	`  lowerBound = -1`
	`  upperBound = 1`
	`  if Coord == 0 :`
	`  lowerBound = 0`
	`  elif Coord == 39 :`
	`  upperBound = 0`
	`  return random.randint(lowerBound, upperBound)`	

6(c)(ii)	1 mark per bullet to max 4
	- Procedure move header, no parameters
	- Calling GenerateDirection **twice** sending across and down as separate parameters
	- Add return value to Across
	- Add return value to Down
	- Check if the grid, at the (new) coordinates == "F"
	- ..if true, Call EatFood
	
	Example python
	
	`def Move(self) :`
	`  self.Across += GenerateChangeInCoordinate(self.Across)`
	`  self.Down += GenerateChangeInCoordinate(self.Down)`
	`  if grid[self.Across][self.Down] == 'F' :`
	`  self.EatFood()`
	`  return`
		4

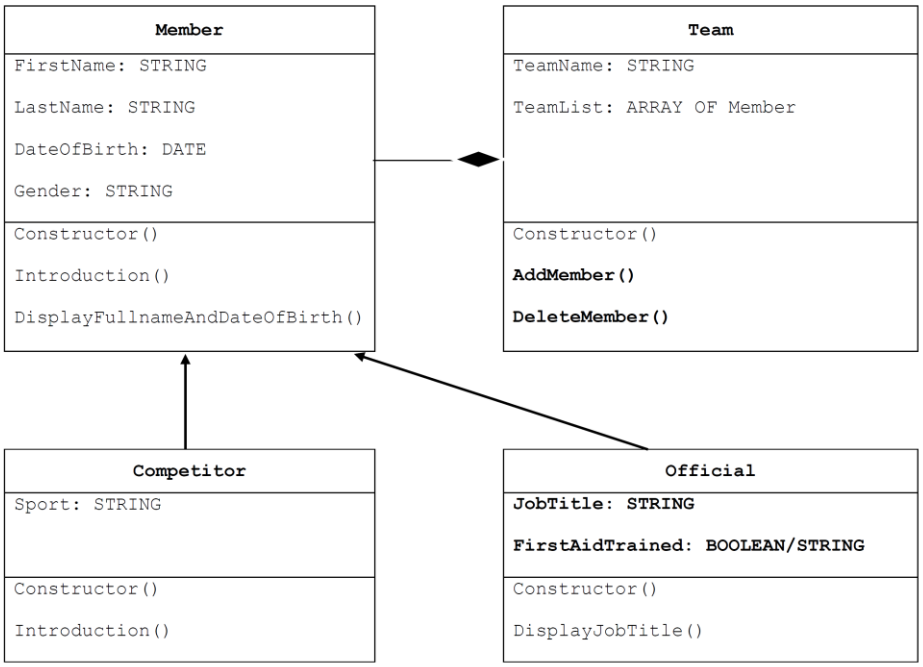
Question No. 10

6(a)	1 mark per bullet ☐ Inheritance correctly shown from `CurrentAccount` and `SavingsAccount` to `Account` ☐ Level and cost methods, get and set functions in `CurrentAccount` ☐ Get and set Amount and constructor in `SavingsAccount`  	3

6(b)	1 mark per bullet to max 5
	☐ Class heading and ending
	☐ Identifying inheritance
	☐ Declaring `AccountNumber`, `Balance`
	☐ Use of private/protected for `AccountNumber` and `Balance`
	☐ One Correct `Get` Method
	☐ One Correct `Set` Method
	☐ Second correct `Get` and `Set` Methods
	
	Example Python
	`class Account:`
	`  def __init__(self, accountNumber, balance):`
	`  self.__accountNumber = accountNumber`
	`  self.__balance = balance`
	
	`  def getAccountNumber(self):`
	`  return self.__accountNumber:`
	`  def setAccountNumber(self, AccountNumber):`
	`  self.___AccountNumber = AcountNumber`
	
	`  def getBalance(self):`
	`  return self._balance:`
	`  def setBalance(self, Balance):`
	`  self.___Balance = Balance`
		5

6(c)	1 mark per bullet to max 5
	☐ Class declaration and end
	☐ Declaration of inheritance
	☐ `Amount` and `PaymentInterval` as Private/protected with appropriate data types
	
	Constructor:
	☐ Override / Overriding in constructor
	☐ Constructor heading and end
	☐ …taking values as parameters
	☐ Constructor setting all values using base class
	☐ Initialisations of new attributes in the constructor
	☐ … all set to the parameters
	
	Example Python
	`class SavingsAccount(Account):`
	
	`  def_init_(self, AccountNumber, Balance, PayInt, AmountP):`
	`  super().__init__(AccountNumber, Balance)`
	`  self.__PaymentInterval = PayInt`
	`  self.__Amount = AmountP`
		5

Question No. 11

4(a)	1 mark per bullet: □ Team methods □ Official attributes □ Two inheritance arrows **or** containment  	3

4(b)	1 mark per bullet to max 5
	
	□ class declaration
	□ `FirstName`, `LastName`, `DateOfBirth` and Gender all defined as private
	
	□ constructor declaration
	□ …all four attributes assigned values from parameters
	
	□ (Public) method for `Introduction`
	□ …outputs message with `FirstName` and `LastName` attributes
	// returns `FirstName` and `LastName`
	
	□ Public method for `DisplayFullNameAndDateofbirth`
	□ … outputs message with `FirstName`, `LastName` and `DateOfBirth`
	// returns `FirstName`, `LastName`, `DateOfBirth`
	
	Python example code:
	
	`class Member:`
	
	`  def __init__(self, Fname, Lname, DOB, GenderVal):`
	`  self.__FirstName = Fname`
	`  self.__LastName = Lname`
	`  self.__DateOfBirth = DOB`
	`  self.__Gender = GenderVal`
	
	`  def Introduction(self):`
	`  return "Hello, I am ", self.__FirstName, " ",`
	`  self.LastName`
	
	`  def DisplayFullnameAndDateofbirth(self):`
	`  print self.__FirstName, self.__LastName,`
	`  self.__DateOfBirth`
		5

4(c)	1 mark per bullet to max 5
	
	□ Class declaration that inherits from Member
	□ Constructor declaration taking all five parameters
	□ …that inherits from Member
	□ Declaration of Sport as private String
	□ …. and assigning to parameter
	□ Introduction method declaration (with polymorphism)
	□ …returning/outputting message with `FirstName`, `LastName` and `Sport` variables
	
	Python example code:
	
	`class Competitor(Member):`
	`  def __init__(self, Fname, Lname, DOB, GenderVal, MySport):`
	`  Member.__init__(self, Fname, Lname, DOB, GenderVal)`
	`  self.__Sport = MySport`
	
	`  def Introduction(self):`
	`  print "Hello, I am %s %s and my sport is %s" % (self.FirstName,`
	`  self.LastName, self.__Sport)`
		5

4(d)	1 mark per bullet
	
	□ variable BMXJudge assigned value
	□ call Official
	□ with all 6 parameters assigned correctly
	
	Python example code:
	
	`BMXJudge = Official("Omar", "Ellaboudy", "17/03/1993", "Male", true, "Judge")`
	
	Visual Basic example code:
	
	`BMXJudge = New Official("Omar", "Ellaboudy", "17/03/1993", "Male, true, "Judge")`
	
	Pascal example code:
	
	`BMXJudge := Official("Omar", "Ellaboudy", "17/03/1993", "Male", true, "Judge")`
		3

Question No. 12

4(a)	• Correct header and close (where applicable) with one parameter (ignore other parameters)
	• parameter (any identifier) assigned to attribute `FoodID`
	• Correct values assigned to `Name` (`""`) **and** `Calories` (`0`)
	
	PYTHON
	`def __init__(self, NewFoodID):`
	`  self.__FoodID = NewFoodID`
	`  self.__Name = ""`
	`  self.__Calories = 0`
	PASCAL
	`Constructor FoodItem.Create(NewFoodID : String);`
	`Begin`
	`  FoodID := NewFoodID;`
	`  Name : = "";`
	`  Calories := 0;`
	`End;`
	VB
	`Public Sub New(ByVal NewFoodID As String)`
	`  FoodID = NewFoodID`
	`  Name = ""`
	`  Calories = 0`
	`End Sub`
		3

4(b)	• Correct function header and close (where applicable) with no parameter (if they have a return data type it must be correct (Integer), but not necessary)
	• Returns `Calories` without other input/assignment (using return command, or assigning to `GetCalories`)
	
	PYTHON
	`def GetCalories(self):`
	`  return(self.__Calories)`
	PASCAL
	`Function FoodItem.GetCalories() : Integer;`
	`Begin`
	`  GetCalories := Calories;`
	`End`
	VB
	`Public Function GetCalories() As Integer`
	`  Return Calories`
	`End function`
		2

4(c)	• Correct **function** header (and close) with **one** parameter passed (ignore additional parameters) (if they have a return data type it must be correct (Boolean), but not necessary)
	• Checks **parameter** is an integer between 0/1 **and** less than 2000.
	• …**Returns** true if parameter is valid **and** assigns **parameter** to `Calories`
	• …**Returns** false if invalid **and** does not assign the parameter to `Calories`
	
	`FUNCTION SetCalories(NumCalories : INTEGER) RETURNS BOOLEAN`
	`  DECLARE Valid : BOOLEAN`
	`  IF NumCalories > 0 AND NumCalories < 2000`
	`  THEN`
	`  Calories ← NumCalories`
	`  Valid ← TRUE`
	`  ELSE`
	`  Valid ← FALSE`
	`  ENDIF`
	`  RETURN Valid`
	`ENDFUNCTION`
		4

4(d)(i)	Two from: • Limits access to given/set/get methods only // can only be accessed through the methods • …attributes cannot be **accidentally** changed // ensure attribute integrity/security against **accidental** change (not program) • Use of set method allows for validation of attribute • .. ensure attribute not set to inappropriate value // make sure attribute value is valid • Ensures encapsulation	**2**
4(d)(ii)	Two from: • Child class can use/has the attributes/methods of its parent class (Accept transferring attributes/methods. • The class `DailyCalories` inherits (attributes/methods) from the class `CustomerProfile` • `DailyCalories` can **use/extend** the attributes/methods from `CustomerProfile` // by example	**2**
4(d)(iii)	Two from: • Child class method/attribute can **override** parent class method/attribute // related / parent and child class have same method that has **different functions/purpose** • `GetTotalCalories()`/`SetTotalCalories()` method from `CustomerProfile` **overwritten**/has **different function** in `DailyCalories` • `TotalCalories` in `DailyCalories` **overrides** `TotalCalories` in `CustomerProfile`	**2**
4(e)	Two from: • Writing a program as a sequence of (explicit) steps/commands // sequence of events/steps // step-by-step instructions • … to gain a required outcome/result // focus is on how to achieve a result / solve a problem • The statements in the program manipulate the data • An example would be procedural programming	**2**