5 — **One** mark per correct answer (**Max 4**)

OOP Term	Purpose
Getter	A method that accesses the value of a property
Setter	A method that changes the value of a property
Object	**An instantiation / instance of a class**
Method	**A programmed function / procedure / subroutine / subprogram defined as part of a class**

3(a)(i) — 1 mark each
- Class header (and end when appropriate)
- Four attributes with appropriate data types
- Constructor header (and end where appropriate) within class taking (min) 4 parameters …
- … assigning each parameter to its attribute

3(a)(ii) — 1 mark each
- `Description()` **method** header (and end where appropriate) with **no** parameter
- Concatenating the attributes with the given message …
- … and returning the created message

Example program code:

Java
```
public String Description(){
      String Message  = "The animal's name is " + Name + ", it makes a " + Sound + ", its size is " +
+ " and its intelligence level is " + Intelligence;
     return Message;
}
```

VB.NET
```
Function Description()
    Dim Message As String = "The animal's name is " & Name & ", it makes a " & Sound & ", its size is
CStr(Size) & " and its intelligence level is " & CStr(Intelligence)
    Return Message
End Function
```

Python
```
def Description(self):
        Message = "The animal's name is " + self.Name + ", it makes a " + self.Sound + ", its size is
str(self.Size) + " and its intelligence level is " + str(self.Intelligence)
        return Message
```

3(b)(i)	1 mark each • Class header (and end where appropriate) **inherits** from `Animal` • Constructor header (and end where appropriate) taking 6 parameters within class **and** calling parent constructor with the four parameters … • … `WingSpan` and `NumberWords` attributes defined with data types **and** parameters assigned within constructor • `ChangeNumberWords()` **method** header (and end where appropriate) takes one parameter **and** adds parameter to attribute `NumberWords`

Example program code:

Java

```
class Parrot extends Animal{
      public Integer WingSpan;
      public Integer NumberWords;

      public Parrot(String pName, String pSound, Integer pSize, Integer pIntelligence, Integer pWingSp
Integer pNumberWords){
            super(pName, pSound, pSize, pIntelligence);
            WingSpan = pWingSpan;
            NumberWords = pNumberWords;
      }

      public void ChangeNumberWords(Integer Change){
            NumberWords = NumberWords + Change;
      }}
```

VB.NET

```
Class Parrot
    Inherits Animal
    Dim WingSpan As Integer
    Dim NumberWords As Integer
    Sub New(pName, pSound, pSize, pIntelligence, pWingSpan, pNumberWords)
        MyBase.New(pName, pSound, pSize, pIntelligence)
        WingSpan = pWingSpan
        NumberWords = pNumberWords
    End Sub
```

3(b)(ii)	1 mark each • `Description()` method header (and end where appropriate) taking **no** parameters **and** overriding/overloads/extending/using parent method • Concatenating and returning the correct string

Example program code:

Java

```
public String Description(){
      String Message  = "The animal's name is " + Name + ", it makes a " + Sound + ", its size is " +
+ " and its intelligence level is " + Intelligence + ". It has a wingspan of " + WingSpan + "cm and c
" + NumberWords + " words.";
     return Message;
}
```

VB.NET

```
Overloads Function Description()
   Dim Message As String = "The animal's name is " & Name & ", it makes a " & Sound & ", its size is
CStr(Size) & " and its intelligence level is " & CStr(Intelligence) & ". It has a wingspan of " &
CStr(WingSpan) & "cm and can say " & CStr(NumberWords) & " words."
    Return Message
End Function
```

Python

```
def Description(self):
        Message = "The animal's name is " + self.Name + ", it makes a " + self.Sound + ", its size is
str(self.Size) + " and its intelligence level is " + str(self.Intelligence) + ". It has a wingspan of
str(self.WingSpan) + "cm and can say " + str(self.NumberWords) + " words."
        return Message
```

3(c)(i)	1 mark each • Class header (and end where appropriate) **inherits** from `Animal` • Constructor header (and end where appropriate) taking 5 parameters within class **and** calling parent constructor with parameters • Attribute `Territory` defined as int and parameter assigned within constructor • `SetTerritory()` **method** header (and end) takes 1 parameter and adds parameter to attribute `TerritorySize`

Example program code:

Java
```
class Wolf extends Animal{
      public Integer TerritorySize;

      public Wolf(String pName, String pSound, Integer pSize, Integer pIntelligence, Integer
pTerritorySize){
            super(pName, pSound, pSize, pIntelligence);
            TerritorySize = pTerritorySize;
      }

      public void SetTerritorySize(Integer Change){
            TerritorySize = TerritorySize + Change;
      }}
```

VB.NET
```
Class Wolf
    Inherits Animal
    Dim TerritorySize As Integer
    Sub New(pName, pSound, pSize, pIntelligence, pTerritory)
        MyBase.New(pName, pSound, pSize, pIntelligence)
        TerritorySize = pTerritory
    End Sub

    Sub SetTerritorySize(Change)
        TerritorySize = TerritorySize + Change
    End Sub
End Class
```

3(c)(ii)	1 mark each • `Description()` method header (and end where appropriate) taking **no** parameters **and** overriding/overloads/extending/using parent method • Concatenating **and** return correct message

Example program code:

Java
```
public String Description(){
      String Message  = "The animal's name is " + Name + ", it makes a " + Sound + ", its size is " +
+ " and its intelligence level is " + Intelligence + ". Its territory is " + TerritorySize + " square
miles.";
      return Message;
}
```

VB.NET
```
Overloads Function Description()
    Dim Message As String = "The animal's name is " & Name & ", it makes a " & Sound & ", its size is
CStr(Size) & " and its intelligence level is " & CStr(Intelligence) + ". Its territory is " &
CStr(TerritorySize) & " square miles."
    Return Message
End Function
```

Python
```
def Description(self):
        Message = "The animal's name is " + self.Name + ", it makes a " + self.Sound + ", its size is
str(self.Size) + " and its intelligence level is " + str(self.Intelligence) + " it's territory is " +
str(self.TerritorySize) +" square miles."
        return Message
```

3(d)(i)	1 mark each • 1 correct instance created and stored in a suitable variable/structure • 2nd and 3rd correct instances created and stored in a suitable variable/structure

Example program code:

Java
```
Parrot Animal1 = new Parrot("Chewie", "Squawk", 1, 10, 30, 29);
Wolf Animal2 = new Wolf("Nighteyes", "Howl", 8, 7, 100);
Animal Animal3 = new Animal("Copper", "Neigh", 10, 6);
```

VB.NET
```
Dim Animal1 As Parrot
Animal1 = New Parrot("Chewie", "Squawk", 1, 10, 30, 29)
Dim Animal2 As Wolf
Animal2 = New Wolf("Nighteyes", "Howl", 8, 7, 100)
Dim Animal3 As Animal
Animal3 = New Animal("Copper", "Neigh", 10, 6)
```

Python
```
Animal1 = Parrot("Chewie","Squawk",1,10,30,29)
Animal2 = Wolf("Nighteyes","Howl",8,7,100)
Animal3 = Animal("Copper", "Neigh", 10, 6)
```

3(d)(ii)	1 mark each • Calling `SetTerritorySize(-20)` for instance of Nighteyes • Calling `ChangeNumberWords(2)` for instance of Chewie • Calling `Description()` for all 3 animals (after any updates) and outputting return values

Example program code:

Java
```
Animal2.SetTerritorySize(-20);
Animal1.ChangeNumberWords(2);
System.out.println(Animal1.Description());
System.out.println(Animal2.Description());
System.out.println(Animal3.Description());
```

VB.NET
```
Animal2.SetTerritorySize(-20)
Animal1.ChangeNumberWords(2)
Console.WriteLine(Animal1.Description())
Console.WriteLine(Animal2.Description())
Console.WriteLine(Animal3.Description())
```

Python
```
Animal2.SetTerritorySize(-20)
Animal1.ChangeNumberWords(2)
print(Animal1.Description())
print(Animal2.Description())
print(Animal3.Description())
```

3(d)(iii)	1 mark each: • All three messages accurate with all relevant values • … showing correctly updated territory for Nighteyes and words for Chewie

e.g.

```
The animal's name is Chewie, it makes a Squawk, its size is 1 and its intelligence level is
It has a wingspan of 30cm and can say 31 words.
The animal's name is Nighteyes, it makes a Howl, its size is 8 and its intelligence level is
it's territory is 80 square miles.
The animal's name is Copper, it makes a Neigh, its size is 10 and its intelligence level is
```


3(a)(i)	1 mark each • Class `Node` header (and end) • `TheData` declared as Integer, `NextNode` declared as `Node` • Constructor header (and end) taking (min) 1 parameter … • … storing parameter to `TheData` within constructor and storing null value to `NextNode` within constructor

Example program code:

Python
```
class Node:
    def __init__(self, NodeData):
        self.__TheData = NodeData #Integer
        self.__NextNode = None #Node
```
Java
```
class Node{
     public Integer TheData;
     public Node NextNode;

     public Node(Integer NodeData){
          TheData = NodeData;
          NextNode = null;
    }}
```

VB.NET
```
Class Node
   Public TheData As Integer
   Public NextNode As Node
   Sub New(NodeData)
      TheData = NodeData
      NextNode = Nothing
   End Sub
End Class
```

3(a)(ii)	1 mark each • 1 get method header (and close) taking no parameters … • … returning correct value (without overwriting) • 2nd correct get method

Example program code:

Python
```
def GetData(self):
    return self._TheData
def GetNextNode(self):
    return self._NextNode
```

Java
```
public Integer GetData(){
       return TheData;
}
public Node GetNextNode(){
       return NextNode;
}
```

VB.NET
```
Function GetData()
   Return TheData
End Function
Function GetNextNode()
   Return NextNode
End Function
```

3(a)(iii)	1 mark each • `SetNextNode()` method header (and close) taking 1 parameter (of type `Node`) … • … storing parameter in `NextNode`

Example program code:

Python
```
def SetNextNode(self, pNextNode):
    self._NextNode = pNextNode
```

Java
```
public void SetNextNode(Node pNextNode){
     NextNode = pNextNode;
}
```

VB.NET
```
Sub SetNextNode(pNextNode)
   NextNode = pNextNode
End Sub
```

3(b)(i)	1 mark each • Class `LinkedList` header (and close) and constructor header with no parameter (and close) … • … declaring `HeadNode` as type `Node` and storing null value in constructor

Example program code:

Python
```
class LinkedList:
    def _init_(self):
        self._HeadNode = None #Node
```
Java
```
class LinkedList{
      public Node HeadNode;
      public LinkedList(){
          HeadNode = null;
      }}
```

VB.NET
```
Class LinkedList
   Private HeadNode As Node
   Sub New()
      HeadNode = Nothing
   End Sub
End Class
```

3(b)(ii)	1 mark each • `InsertNode()` method header (and close) taking one (integer) parameter • Creating new instance of `Node` with the parameter as the argument • Calling `SetNextNode()` for new node with `HeadNode` as parameter • Replacing `HeadNode` with new node

Example program code:

Python
```
def InsertNode(self, NodeData):
    TheNode = Node(NodeData)
    TheNode.SetNextNode(self._HeadNode)
    self._HeadNode = TheNode
```

Java
```
public void InsertNode(Integer NodeData){
     Node TheNode = new Node(NodeData);
     TheNode.SetNextNode(HeadNode);
     HeadNode = TheNode;
}
```

VB.NET
```
Sub InsertNode(NodeData)
   Dim TheNode As Node = New Node(NodeData)
   TheNode.SetNextNode(HeadNode)
   HeadNode = TheNode
End Sub
```

3(b)(iii)	1 mark each • `Traverse()` method header (and close) with no parameter and returns created string • Starts at head node and follows nodes using `GetNextNode()` until no nodes left … • … concatenates the data from each node and formats correctly

3(b)(iv)	1 mark each to max 6 • `RemoveNode()` method header (and close) taking (integer) parameter and returning Boolean in all cases • Checking if head node is null and returning `FALSE` • Checking if head node equals parameter and returning `TRUE` if true … • … and updating `HeadNode` to `HeadNode.GetNextNode()` • Following nodes comparing data from each node to parameter … • ... if found updating next node and returning `TRUE` • … if end of list returning `FALSE`

3(c)(i)	1 mark each • Creating new `LinkedList` object • Calling `InsertNode()` five times with correct data in correct order • Calling `RemoveNode(30)` and calling `Traverse()` and store/output the return value, before `RemoveNode()` and after Full marks can be awarded to students who may have stored and/or outputted the return value from the function call.

3(c)(ii)	1 mark each • Output of linked list with 50 40 30 20 10 • Output of 2nd linked list with 50 40 20 10

e.g.

```
50 40 30 20 10
50 40 20 10
```


10(a)	**One** mark per mark point (**Max 3)** MP1 Attributes/properties … MP2 … with their data types // … are variables bound to the class MP3 Methods … MP4 … that are subroutines / functions / procedures that act upon the attributes MP5 Getters/setters … MP6 … are methods that can fetch / update the contents of attributes MP7 A constructor … MP8 … that is used to create instances / objects of this class.
10(b)	**One** mark per mark point (**Max 3)** MP1 A class is only defined once but many objects can be created from that class // A class is a template / blueprint from which objects are created // An object is an instance of a class MP2 No memory is allocated when a class is defined, but objects are allocated memory space whenever they are created MP3 A class cannot be manipulated as it is not available in the memory, but objects can be manipulated MP4 A class is defined but an object is declared / created / instantiated. MP5 A class can use inheritance. An object cannot.

9(a)	To ensure that the attributes are only accessible using the class's own methods/within the class.
9(b)	**One** mark per mark point (**Max 5**) MP1 Two correct attributes with sensible names and correct data types. MP2 Constructor present. MP3 Two correct setters with exact names and appropriate parameters and data types. MP4 Two correct getters with appropriate names. MP5 Name assigned to pet name getter matches the attribute. **Pet** PetID : STRING PetType : STRING OwnerTelephone : STRING DateRegistered : DATE **PetName : STRING** **OwnerName : STRING** **Constructor()** **SetPetID(APetID : STRING)** **SetDateRegistered(RegDate : DATE)** **GetPetName()** **GetOwnerTelephone()**