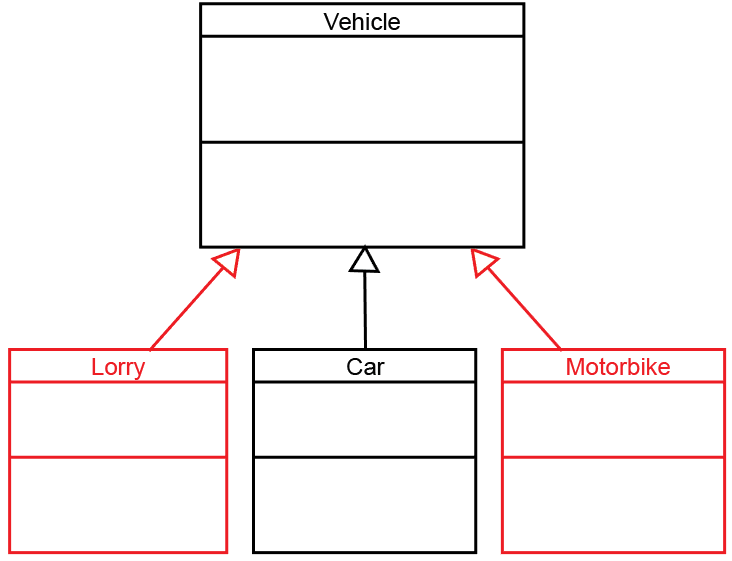
# Homework 2 Object-oriented design principles Answers

1. An incomplete class diagram for a system which holds and processes details of vehicles registered in the UK is shown below.



(a) Show on the diagram where the classes Lorry and Motorbike should be added. [2]

(see diagram). Open-headed arrows should be used.

(b) State the reason you positioned them as you did. [1]

Lorry “is a” vehicle, motorbike “is a” vehicle, therefore both are subclasses of the superclass vehicle which will inherit all their attributes and methods

(c) The registration numbers need to be stored.

State which class should store the registration numbers and give a reason for   
your answer. [2]

Vehicle should store the registration number since all vehicles will have this attribute. It will be inherited by Lorry, Motorbike and Car.

(d) Explain the term polymorphism, and give an example of how it could be used   
in the above example. [3]

The ability of the language to process objects differently depending on their class. Methods can be redefined so that for example a method setAnnualLicenceFee in Vehicle may be redefined in each class to set the Licence Fee differently.

2. **Composition** and **aggregation** are both “has a” relationships.

(a) Explain the difference between them. [2]

Composition – if the containing object is destroyed, so are the objects it contains.

Aggregation – the objects remain when the container object is destroyed.

(b) For each of the pairs of classes below, state whether the relationship is **composition** or **aggregation** and draw diagrams to show the relationship.

(i) Team and player [3]

Aggregation



(ii) hotel and room [3]

Composition



3. A simple drawing program can draw two different shapes, **Rectangle** and **Circle**.

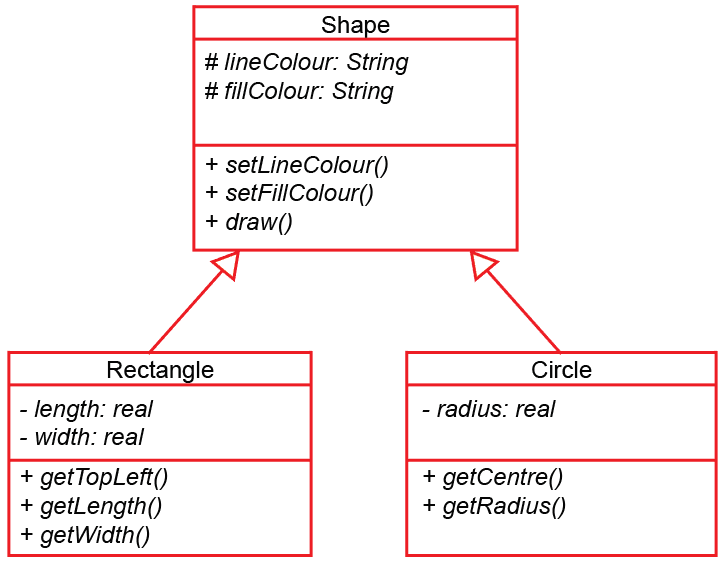
All **shape** objects have attributes **lineColour** and **fillColour**, and methods **setLineColour()**, **setFillColour()** and **draw()**.

Rectangle objects have attributes **length** and **width**, and methods **getTopLeft()**,   
**getLength()**, **getWidth()**.

Circle objects have attribute **radius** and methods  **getCentre()**, **getRadius().**

The **draw()** method needs to be polymorphic.

Draw a class diagram showing public (+), private (-) and protected (#) attributes and   
methods for each class. [9]



[Total 25 marks]

**Practical exercises**

A class is to be defined that acts as a stack data structure. A class diagram is shown below.

(a) Write the code to declare NodeClass [4]

(b) Write the code to declare StackClass [12]



(c) Test your program code and screen capture your output. [4]

**Python**

class NodeClass :

def \_\_init\_\_(self):

self.\_\_Data = ''

def SetData(self, d):

self.\_\_Data = d

def GetData(self):

return(self.\_\_Data)

class StackClass :

def \_\_init\_\_(self):

self.\_\_Stack = [NodeClass() for i in range(51)]

self.\_\_TopOfStackPointer = -1

self.\_\_MaxSize = 50

def pushOnStack(self, d):

if self.\_\_TopOfStackPointer == self.\_\_MaxSize :

print("stack full")

else:

self.\_\_TopOfStackPointer += 1

self.\_\_Stack[self.\_\_TopOfStackPointer].SetData(d)

def popFromStack(self) :

if self.\_\_TopOfStackPointer == -1 :

return("stack empty")

else:

data = self.\_\_Stack[self.\_\_TopOfStackPointer].GetData()

self.\_\_TopOfStackPointer -= 1

return (data)

ThisStack = StackClass()

print(ThisStack.popFromStack())

ThisStack.pushOnStack('A')

ThisStack.pushOnStack('B')

ThisStack.pushOnStack('C')

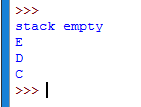
ThisStack.pushOnStack('D')

ThisStack.pushOnStack('E')

print(ThisStack.popFromStack())

print(ThisStack.popFromStack())

print(ThisStack.popFromStack())



**VB.net**

Module Module1

Class NodeClass

Private Data As String

Public Sub New()

Data = ""

End Sub

Public Sub SetData(ByVal d As String)

Data = d

End Sub

Public Function GetData() As String

Return (Data)

End Function

End Class

Class StackClass

Private Stack(50) As NodeClass

Private TopOfStackPointer As Integer

Private MaxSize As Integer

Public Sub New()

TopOfStackPointer = -1

MaxSize = 50

For count = 0 To MaxSize

Stack(count) = New NodeClass

Next

End Sub

Public Sub pushOnStack(d)

If TopOfStackPointer = MaxSize Then

Console.Writeline("stack full")

Else

TopOfStackPointer += 1

Stack(TopOfStackPointer).setData(d)

End If

End Sub

Public Function popFromStack()

Dim data As String

If TopOfStackPointer = -1 Then

Return ("stack empty")

Else

data = Stack(TopOfStackPointer).getData()

TopOfStackPointer -= 1

Return (data)

End If

End Function

End Class

Sub Main()

Dim ThisStack As New StackClass()

Console.WriteLine(ThisStack.popFromStack())

ThisStack.pushOnStack("A")

ThisStack.pushOnStack("B")

ThisStack.pushOnStack("C")

ThisStack.pushOnStack("D")

ThisStack.pushOnStack("E")

Console.WriteLine(ThisStack.popFromStack())

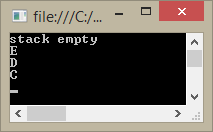
Console.WriteLine(ThisStack.popFromStack())

Console.WriteLine(ThisStack.popFromStack())

Console.ReadLine()

End Sub

End Module



[Total 20 Marks]