# Worksheet 2 Object-oriented design principles Answers

**Task 1**

Program the superclass Animal and the subclasses Fish and Duck and test your code.

Pseudocode

Animal = Class

Protected

state: String

size: Integer

Public

Constructor(s, n)

state 🡨 s

size 🡨 n

End Constructor

Procedure feed()

size 🡨 size + 1

Output state, " fed"

End Procedure

Function getState()

Return state

End Function

Function getSize()

Return size

End Function

End Class

Fish = SubClass(Animal)

Private

maxSize : Integer

Public

Constructor(s)

Animal.Constructor(s,1)

maxSize 🡨 2

End Constructor

Procedure setMaxSize(m)

maxSize 🡨 m

End Procedure

Procedure feed Override

size += 2

Output state, " fed"

If size >= maxSize Then

state 🡨 "BIG FISH"

End If

End Procedure

End Class

Duck = Subclass(Animal)

Public

Constructor(s)

Animal.Constructor(s,3)

End Constructor

Procedure feed Override

Animal.feed()

If size = 5 Then

state 🡨 "BIG DUCK"

End If

End Procedure

End Class

thisFish 🡨 Fish("little fish")

thisFish.setMaxSize(3)

thisDuck 🡨 Duck("little duck")

For count 🡨 1 To 3

thisDuck.feed()

Output(thisDuck.getState())

thisFish.feed()

Output(thisFish.getState())

End For

**Python**

class Animal:

def \_\_init\_\_(self, s, n):

self.\_state = s

self.\_size = n

def getState(self):

return self.\_state

def getSize(self):

return self.\_size

def feed(self):

self.\_size += 1

print(self.\_state, "fed")

class Duck(Animal):

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

Animal.\_\_init\_\_(self, s, 3)

def feed(self):

Animal.feed(self)

if self.\_size == 5:

self.\_state = 'BIG DUCK'

class Fish(Animal):

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

Animal.\_\_init\_\_(self, s, 1)

self.\_\_maxSize = 2

def setMaxSize(self, n):

self.\_maxSize = n

def feed(self):

self.\_size += 2

print(self.\_state, "fed")

if self.\_size >= self.\_\_maxSize:

self.\_state = 'BIG FISH'

thisFish = Fish('little fish')

thisFish.setMaxSize(3)

thisDuck = Duck('little duck')

for count in range(3):

thisDuck.feed()

print(thisDuck.getState())

thisFish.feed()

print(thisFish.getState())

*See Python and VB.net programs WS2 Task1.py/vb*

**VB.net**

Module Module1

Class Animal

Protected state As String

Protected size As Integer

Public Sub New(ByVal s As String, ByVal n As Integer)

state = s

size = n

End Sub

Public Overridable Sub feed()

size = size + 1

Console.WriteLine(state & " fed")

End Sub

Function getState() As String

Return state

End Function

Function getSize() As Integer

Return size

End Function

End Class

Class Duck

Inherits Animal

Public Sub New(ByVal s As String)

MyBase.New(s, 3)

End Sub

Public Overrides Sub feed()

MyBase.feed()

If size = 5 Then

state = "BIG DUCK"

End If

End Sub

End Class

Class Fish

Inherits Animal

Private maxSize As Integer

Public Sub New(ByVal s As String)

MyBase.New(s, 1)

maxSize = 2

End Sub

Public Sub setMaxSize(ByVal n As Integer)

maxSize = n

End Sub

Public Overrides Sub feed()

size = size + 2

Console.WriteLine(state & " fed")

If size >= maxSize Then

state = "BIG FISH"

End If

End Sub

End Class

Sub Main()

Dim thisFish As New Fish("little fish")

Dim thisDuck As New Duck("little duck")

thisFish.setMaxSize(3)

For count = 1 To 3

thisDuck.feed()

Console.WriteLine(thisDuck.getState())

thisFish.feed()

Console.WriteLine(thisFish.getState())

Next

Console.ReadLine()

End Sub

End Module

**Task 2**

Draw a class diagram for a superclass Vehicle and subclasses Car and Lorry.

All classes have attributes VehicleID, Mileage (number of whole miles), TankContents (number of litres).

The Car class also has attributes NumberOfSeats.

The Lorry class also has the attributes MaxLoadWeight (in kg) and CurrentLoad (in kg).

Methods for the superclass are Drive, Refuel and CalculateUsage.

The Car class has the additional method AddPassengers.

The Lorry class has additional methods of AddLoad and Unload.

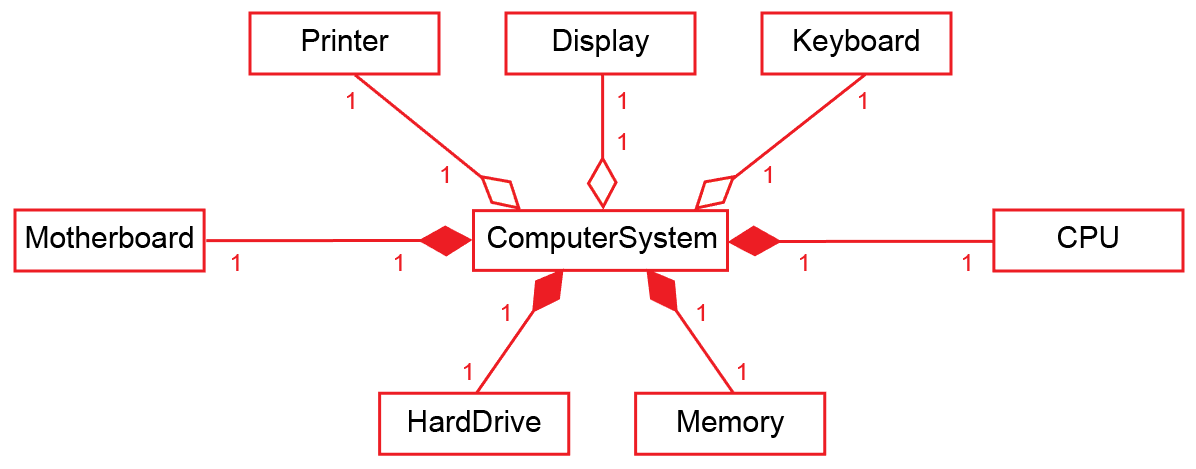
The subclasses implement the CalculateUsage method differently and require access to the Mileage attribute.



**Task 3**

Draw a diagram of a ComputerSystem class that is composed of the following classes:

Display, Keyboard, Memory, CPU, HardDrive, Printer, Motherboard.



*(Note: there are other possible interpretations)*

**Task 4**

Program the classes Order, OrderStatus and Product and test your code.

Pseudocode

Order = Class

Private

OrderNumber: String

OrderDate: Date

ProductsOrdered: Array Of Product

NumberOfItemsOrdered: Integer

Status: OrderStatus

End Class

OrderStatus = Class

Private

HasShipped: Boolean

End Class

Product = Class

Private

ProductID: String

ProductPrice: Currency

End Class

product1 🡨 new Product("beans", 0.45)

product2 🡨 new Product("eggs", 1.25)

myOrder 🡨 new Order(1, "1/1/17")

myOrder.OrderItem(product1)

myOrder.OrderItem(product2)

Output(myOrder.getOrderStatus())

Output(myOrder.getOrderItemID(1))

Output(myOrder.getOrderItemPrice(1))

Output(myOrder.getOrderItemID(2))

Output(myOrder.getOrderItemPrice(2))

**Note: you will need to add the required constructor and method definitions**

**Python**

class Order:

def \_\_init\_\_(self, s, d):

self.\_\_orderNumber = s

self.\_\_orderDate = d

self.\_\_numberOfItemsOrdered = 0

self.\_\_productsOrdered = [Product("",0.0)]

self.\_\_status = OrderStatus()

def orderItem(self, p):

self.\_\_numberOfItemsOrdered += 1

i = self.\_\_numberOfItemsOrdered

self.\_\_productsOrdered.append(p)

def getOrderStatus(self):

return self.\_\_status.getHasShipped()

def getOrderItemID(self, i):

return self.\_\_productsOrdered[i].getProductID()

def getOrderItemPrice(self, i):

return self.\_\_productsOrdered[i].getProductPrice()

class OrderStatus:

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

self.\_hasShipped = False

def getHasShipped(self):

return self.\_hasShipped

class Product:

def \_\_init\_\_(self, ID, Price):

self.\_\_productID = ID

self.\_\_productPrice = Price

def getProductID(self):

return self.\_\_productID

def getProductPrice(self):

return self.\_\_productPrice

product1 = Product("beans", 0.45)

product2 = Product("eggs", 1.25)

myOrder = Order(1,"1/1/17")

myOrder.orderItem(product1)

myOrder.orderItem(product2)

print(myOrder.getOrderStatus())

print(myOrder.getOrderItemID(1))

print(myOrder.getOrderItemPrice(1))

print(myOrder.getOrderItemID(2))

print(myOrder.getOrderItemPrice(2))

*See Python and VB.net programs WS2 Task4.py/vb*

**VB.net**

Module Module1

Class Order

Private OrderNumber As Integer

Private OrderDate As Date

Private NumberOfItemsOrdered As Integer

Private ProductsOrdered(10) As Product

Private Status As OrderStatus

Public Sub New(s As Integer, d As Date)

OrderNumber = s

OrderDate = d

NumberOfItemsOrdered = 0

Status = New OrderStatus()

End Sub

Public Sub OrderItem(p As Product)

Dim i As Integer

NumberOfItemsOrdered += 1

i = NumberOfItemsOrdered

ProductsOrdered(i) = p

End Sub

Public Function getOrderStatus()

Return Status.getHasShipped()

End Function

Public Function getOrderItemID(i)

Return ProductsOrdered(i).getProductID()

End Function

Public Function getOrderItemPrice(i)

Return ProductsOrdered(i).getProductPrice()

End Function

End Class

Class OrderStatus

Private HasShipped As Boolean

Public Sub New()

HasShipped = False

End Sub

Public Function getHasShipped()

Return HasShipped

End Function

End Class

Class Product

Private ProductID As String

Private ProductPrice As Decimal

Public Sub New(ID As String, Price As Decimal)

ProductID = ID

ProductPrice = Price

End Sub

Public Function getProductID() As String

Return ProductID

End Function

Public Function getProductPrice() As Decimal

Return ProductPrice

End Function

End Class

Dim myOrder As Order

Dim product1 As Product

Dim product2 As Product

Sub Main()

product1 = New Product("beans", 0.45)

product2 = New Product("eggs", 1.25)

myOrder = New Order(1, "1/1/17")

myOrder.OrderItem(product1)

myOrder.OrderItem(product2)

Console.WriteLine(myOrder.getOrderStatus)

Console.WriteLine(myOrder.getOrderItemID(1))

Console.WriteLine(myOrder.getOrderItemPrice(1))

Console.WriteLine(myOrder.getOrderItemID(2))

Console.WriteLine(myOrder.getOrderItemPrice(2))

Console.ReadLine()

End Sub

End Module