Chapter 27 Object-oriented Programming (OOP): Answers to coursebook questions and tasks

Syllabus sections covered: 4.3.1

Task 27.01 part 1

Python

```python
class Car:
   def __init__(self, n, e):   # constructor
      self.__VehicleID = n
      self.__Registration = ""
      self.__DateOfRegistration = None
      self.__EngineSize = e
      self.__PurchasePrice = 0.00

   def SetPurchasePrice(self, p):
      self.__PurchasePrice = p

   def SetRegistration(self, r):
      self.__Registration = r

   def SetDateOfRegistration(self, d):
      self.__DateOfRegistration = d

   def GetVehicleID(self):
      return(self.__VehicleID)

   def GetRegistration(self):
      return(self.__Registration)

   def GetDateOfRegistration(self):
      return(self.__DateOfRegistration)

   def GetEngineSize(self):
      return(self.__EngineSize)

   def GetPurchasePrice(self):
      return(self.__PurchasePrice)

ThisCar = Car("ABC1234", 2500)
ThisCar.SetPurchasePrice(12000)
print(ThisCar.GetVehicleID())
```

VB.NET

```vbnet
Module Module1
    Class Car
        Private VehicleID As String
        Private Registration As String = ""
        Private DateOfRegistration As Date = #1/1/1900#
        Private EngineSize As Integer
        Private PurchasePrice As Decimal = 0.0

        Public Sub New(ByVal n As String, ByVal e As String)
            VehicleID = n
            EngineSize = e
        End Sub
        Public Sub SetPurchasePrice(ByVal p As Decimal)
            PurchasePrice = p
        End Sub

        Public Sub SetRegistration(ByVal r As String)
            Registration = r
        End Sub

        Public Sub SetDateOfRegistration(ByVal d As Date)
            DateOfRegistration = d
        End Sub

        Public Function GetVehicleID() As String
            Return (VehicleID)
        End Function

        Public Function GetRegistration() As String
            Return (Registration)
        End Function

        Public Function GetDateOfRegistration() As Date
            Return (DateOfRegistration)
        End Function

        Public Function GetEngineSize() As Integer
            Return (EngineSize)
        End Function

        Public Function GetPurchasePrice() As Decimal
            Return (PurchasePrice)
        End Function

    End Class


    Sub Main()
        Dim ThisCar As New Car("ABC1234", 2500)
        ThisCar.SetPurchasePrice(12000)
        Console.WriteLine(ThisCar.GetVehicleID())
        Console.ReadLine()
        ThisCar = Nothing ' garbage collection

    End Sub

End Module
```

Pascal

```
program Project2;

{$APPTYPE CONSOLE}

uses
  SysUtils;

type
   Car   = class
      private
         VehicleID : string;
         Registration : string;
         DateOfRegistration : TDateTime;
         EngineSize : integer;
         PurchasePrice : currency;
      public
         constructor Create(n : string; e : integer);
         procedure SetPurchasePrice(p : currency);
         procedure SetRegistration(r : string);
         procedure SetDateOfRegistration(d :
TDateTime);
         function GetVehicleID : string;
         function GetRegistration : string;
         function GetDateOfRegistration: TDateTime;
         function GetEngineSize   : integer;
         function GetPurchasePrice  : currency;
   end;

constructor Car.Create(n : string; e : integer);
begin
   VehicleID := n;
   EngineSize := e;
   Registration := '';
   DateOfRegistration := 0;
   PurchasePrice := 0;
end;

procedure Car.SetPurchasePrice(p : currency);
begin
   PurchasePrice := p;
end;

procedure Car.SetRegistration(r : string);
begin
   Registration := r;
end;

procedure Car.SetDateOfRegistration(d : TDateTime);
begin
   DateOfRegistration := d;
end;

function Car.GetVehicleID : string;
begin
   GetVehicleID := VehicleID;
end;

function Car.GetRegistration : string;
begin
   GetRegistration := Registration;
end;

function Car.GetDateOfRegistration : TDateTime;
begin
   GetDateOfRegistration := DateOfRegistration;
end;

function Car.GetEngineSize : integer;
```

```
begin
   GetEngineSize := EngineSize;
end;

function Car.GetPurchasePrice : currency;
begin
   GetPurchasePrice := PurchasePrice;
end;
// ***** end of class declaration and inplementation
*****
var ThisCar : Car;

begin
   ThisCar := Car.Create('ABC1234', 2500);
   ThisCar.SetPurchasePrice(12000);
   WriteLn(ThisCar.GetVehicleID);
   WriteLn(ThisCar.GetPurchasePrice:5:2);
   ReadLn;
end.
```

Task 27.01 part 2

Class diagram

Company
CompanyName : STRING EmailAddress : STRING DateOFLastContact : DATE

Constructor()
SetDateOfLastContact()
GetCompanyName()
GetEmailAddress()
GetDateOfLastContact()

Python

```
from datetime import *
class Company:
   def __init__(self, n, e):   # constructor
      self.__CompanyName = n
      self.__EmailAddress = e
      self.__DateOfLastContact = None

   def SetDateOfLastContact(self, d):
      self.__DateOfLastContact = d

   def GetCompanyName(self):
      return(self.__CompanyName)
   def GetEmailAddress(self):
      return(self.__EmailAddress)

   def GetDateOfLastContact(self):
      return(self.__DateOfLastContact)

ThisCompany = Company("SLimited", "abc@slimited.cie")
ThisCompany.SetDateOfLastContact(date(2016,2,1))
print(ThisCompany.GetDateOfLastContact())
```

VB.NET

```
Module Module1
    Class Company
        Private CompanyName As String
        Private EmailAddress As String
        Private DateOfLastContact As Date

        Public Sub New(n, e)    ' constructor
            CompanyName = n
            EmailAddress = e
            DateOfLastContact = #1/1/1900#
        End Sub

        Public Sub SetDateOfLastContact(d)
            DateOfLastContact = d
        End Sub

        Public Function GetCompanyName()
            Return (CompanyName)
        End Function

        Public Function GetEmailAddress()
            Return (EmailAddress)
        End Function

        Public Function GetDateOfLastContact()
            Return (DateOfLastContact)
        End Function

    End Class

    Sub Main()
        Dim ThisCompany As New Company("SLimited",
"abc@slimited.cie")
        ThisCompany.SetDateOfLastContact(#1/2/2016#)
        Console.WriteLine(ThisCompany.GetDateOfLastContact())
        Console.ReadLine()
    End Sub

End Module
```

Pascal

```
    type
      Company = class
        Private
          CompanyName : String;
          EmailAddress : String;
          DateOfLastContact : TDateTime;
        public
          constructor Create(n , e : string);
          Procedure SetDateOfLastContact(d :
TDateTime);
```

```
          Function GetCompanyName() : String;
          Function GetEmailAddress() : String;
          Function GetDateOfLastContact() : TDateTime;
      End;

constructor Company.Create(n , e : string);
begin
  CompanyName := n;
  EmailAddress := e;
  DateOfLastContact := StrToDate('1/1/1900');
End;

Procedure Company.SetDateOfLastContact(d : TDateTime);
begin
  DateOfLastContact := d
End;

Function Company.GetCompanyName() : String;
begin
  GetCompanyName := CompanyName
End;

Function Company.GetEmailAddress() : String;
begin
  GetEmailAddress := EmailAddress
End;

Function Company.GetDateOfLastContact() : TDateTime;
begin
  GetDateOfLastContact := DateOfLastContact
End;

var ThisCompany : Company;
begin
  ThisCompany := Company.Create('SLimited',
'abc@slimited.cie');

ThisCompany.SetDateOfLastContact(StrToDate('1/2/2016')
);

WriteLn(DateToStr(ThisCompany.GetDateOfLastContact()))
;
  ReadLn
end.
```

Task 27.02

Python

```
import datetime
class LibraryItem:

   def __init__(self, t, a, i):           # initialiser
method
      self.__Title = t
      self.__Author_Artist = a
      self.__ItemID = i
      self.__OnLoan = False
      self.__DueDate = datetime.date.today()

   def GetTitle(self):
      return(self.__Title)

   def GetAuthor_Artist(self):
      return(self.__Author_Artist)

   def GetItemID(self):
      return(self.__ItemID)

   def GetOnLoan(self):
      return(self.__OnLoan)

   def GetDueDate(self):
      return(self.__DueDate)

   def Borrowing(self):
      self.__OnLoan = True
      self.__DueDate = self.__DueDate +
datetime.timedelta(weeks=3)

   def Returning(self):
      self.__OnLoan = False

   def PrintDetails(self):
      print(self.__Title, ' ; ', self.__Author_Artist,
end='')
      print(' ; ', self.__ItemID, ' ; ', self.__OnLoan,
end='')
      print(' ; ', self.__DueDate)

class Book(LibraryItem):

   def __init__(self, t, a, i):           # initialiser
method
      LibraryItem.__init__(self, t, a, i)
      self.__IsRequested = False

   def GetIsRequested(self):
      return(self.__IsRequested)

   def SetIsRequested(self):
      self.__IsRequested = True

class CD(LibraryItem):

   def __init__(self, t, a, i):           # initialiser
method
      LibraryItem.__init__(self, t, a, i)
      self.__Genre = ""

   def GetGenre(self):
      return(self.__Genre)

   def SetGenre(self, g):
      self.__Genre = g

def main():
   ThisBook = Book("Computing", "Sylvia", 1234)
   ThisCD = CD("Let it be", "Beatles", 2345)
   ThisBook.PrintDetails()
   ThisCD.PrintDetails()

main()
```

VB.NET

```
Module Module1
    Class LibraryItem
        Private Title As String
        Private Author_Artist As String
        Private ItemID As Integer
        Private OnLoan As Boolean = False
        Private DueDate As Date = Today

        Sub Create(ByVal t As String, ByVal a As String, ByVal i As
Integer)
            Title = t
            Author_Artist = a
            ItemID = i
        End Sub

        Public Function GetTitle() As String
            Return (Title)
        End Function

        Public Function GetAuthor_Artist() As String
            Return (Author_Artist)
        End Function

        Public Function GetItemID() As Integer
            Return (ItemID)
        End Function

        Public Function GetOnLoan() As Boolean
            Return (OnLoan)
        End Function

        Public Function GetDueDate() As Date
            Return (DueDate)
        End Function


        Public Sub Borrowing()
            OnLoan = True
            DueDate = DateAdd(DateInterval.Day, 21, Today()) '3 weeks
from today
        End Sub

        Public Sub Returning()
            OnLoan = False
        End Sub

        Public Sub PrintDetails()
            Console.Write(Title & "; " & ItemID & "; " & OnLoan & ";
")
            Console.WriteLine(DueDate)
        End Sub
    End Class

    Class Book
        Inherits LibraryItem
        Private IsRequested As Boolean = False

        Public Function GetIsRequested() As Boolean
            Return (IsRequested)
        End Function

        Public Sub SetIsRequested()
            IsRequested = True
        End Sub
    End Class

    Class CD
        Inherits LibraryItem
        Private Genre As String = ""

        Public Function GetGenre() As String
            Return (Genre)
        End Function

        Public Sub SetGenre(ByVal g As String)
            Genre = g
        End Sub
    End Class

    Sub Main()
        Dim ThisBook As New Book()
        Dim ThisCD As New CD()
        ThisBook.Create("Computing", "Sylvia", 1234)
        ThisCD.Create("Let it be", "Beatles", 2345)
        ThisBook.PrintDetails()
        ThisCD.PrintDetails()
        Console.ReadLine()
    End Sub

End Module
```

Pascal

```
type
   LibraryItem = class
      private
         Title : STRING;
         Author_Artist : STRING;
         ItemID : INTEGER;
         OnLoan : BOOLEAN;
         DueDate : TDATETIME;
      public
         constructor Create(t, a : STRING; i :
INTEGER);  virtual;
         function GetTitle : STRING;
         function GetAuthor_Artist : STRING;
         function GetItemID : INTEGER;
         function GetOnLoan : BOOLEAN;
         function GetDueDate : TDATETIME;
         procedure Borrowing;
         procedure Returning;
         procedure PrintDetails; virtual;
      end;

   Book = class(LibraryItem)
```

```
      private
         IsRequested : BOOLEAN;
      public
         constructor Create(t, a : STRING; i :
INTEGER);  override;
         function GetIsRequested : BOOLEAN;
         procedure SetIsRequested;
         procedure PrintDetails; override;
      end;

   CD = class(LibraryItem)
      private
         Genre : STRING;
      public
         constructor Create(t, a : STRING; i :
INTEGER);  override;
         function GetGenre : STRING;
         procedure SetGenre(g : STRING);
      end;

// implementation of methods

constructor LibraryItem.Create(t, a : STRING; i :
INTEGER);
begin
   Title := t;
   Author_Artist := a;
   ItemID := i;
   OnLoan := FALSE;
   DueDate := 0;
end;

function LibraryItem.GetTitle : STRING;
begin
   GetTitle := Title;
end;

function LibraryItem.GetAuthor_Artist : STRING;
begin
   GetAuthor_Artist := Author_Artist;
end;

function LibraryItem.GetItemID : INTEGER;
begin
   GetItemID := ItemID;
end;

function LibraryItem.GetOnLoan : BOOLEAN;
begin
   GetOnLoan := OnLoan;
end;

function LibraryItem.GetDueDate : TDATETIME;
begin
   GetDueDate := DueDate;
```

```
end;

procedure LibraryItem.Borrowing;
begin
  OnLoan := TRUE;
  DueDate := Date() + 21;
end;

procedure LibraryItem.Returning;
begin
  OnLoan := FALSE;
end;

procedure LibraryItem.PrintDetails;
begin
  WriteLn(Title, ' ; ', ItemID:7, ' ; ', OnLoan, ' ;
', DateToStr(DueDate))
end;

constructor Book.Create(t, a : STRING; i : INTEGER);
begin
   inherited Create(t, a, i);
   IsRequested := FALSE;
end;

procedure Book.SetIsRequested;
begin
  IsRequested := TRUE;
end;

function Book.GetIsRequested : BOOLEAN;
begin
   GetIsRequested := IsRequested;
end;

constructor CD.Create(t, a : STRING; i : INTEGER);
begin
   inherited Create(t, a, i);
   Genre := '';
end;

function CD.GetGenre : STRING;
begin
   GetGenre := Genre;
end;

procedure CD.SetGenre(g : STRING);
begin
   Genre := g;
end;

var ThisBook : Book; ThisCD : CD;
begin
ThisBook := Book.Create('Computing','Sylvia',1234);
writeln(ThisBook.GetIsRequested);
```

```
ThisCD := CD.Create('Let it be', 'Beatles', 2345);
ThisBook.PrintDetails;
ThisCD.PrintDetails;
Readln;
ThisCD.Free;
ThisBook.Free;
end.
```

Task 27.03

Python

```python
class Borrower() :
   def __init__(self, n, e, b) :
      self.__BorrowerName = n
      self.__EmailAddress = e
      self.__BorrowerID = b
      self.__ItemsOnLoan = 0

   def GetBorrowerName(self) :
      return(self.__BorrowerName)

   def GetEmailAddress(self) :
      return (self.__EmailAddress)

   def GetBorrowerID(self) :
      return (self.__BorrowerID)

   def GetItemsOnLoan(self) :
      return(self.__ItemsOnLoan)

   def UpdateItemsOnLoan(self, n) :
      self.__ItemsOnLoan += n

   def PrintDetails(self) :
      print("Borrower     : ", self.__BorrowerName)
      print("ID           : ", self.__BorrowerID)
      print("email        : ", self.__EmailAddress)
      print("Items on loan: ", self.__ItemsOnLoan)

def main():
   NewBorrower = Borrower("Sylvia", "adc@cie", 123)
   NewBorrower.UpdateItemsOnLoan(3)
   NewBorrower.PrintDetails()
   NewBorrower.UpdateItemsOnLoan(-1)
   NewBorrower.PrintDetails()
```

VB.NET

```vb
Class Borrower
       Private BorrowerName As String
       Private EmailAddress As String
       Private BorrowerID As Integer
       Private ItemsOnLoan As Integer
       Public Sub Create(n As String, e As String, b As Integer)
           BorrowerName = n
           EmailAddress = e
           BorrowerID = b
           ItemsOnLoan = 0
```

```
        End Sub

        Public Function GetBorrowerName() As String
            GetBorrowerName = BorrowerName
        End Function

        Public Function GetEmailAddress() As String
            GetEmailAddress = EmailAddress
        End Function

        Public Function GetBorrowerID() As Integer
            GetBorrowerID = BorrowerID
        End Function

        Public Function GetItemsOnLoan() As Integer
            GetItemsOnLoan = ItemsOnLoan
        End Function

        Public Sub UpdateItemsOnLoan(n As Integer)
            ItemsOnLoan += n
        End Sub

        Public Sub PrintDetails()
            Console.WriteLine("Borrower     : " & BorrowerName)
            Console.WriteLine("ID           : " & BorrowerID)
            Console.WriteLine("email        : " & EmailAddress)
            Console.WriteLine("Items on loan: " & ItemsOnLoan)
        End Sub
    End Class

    Sub Main()
        Dim NewBorrower As New Borrower()
        NewBorrower.Create("Sylvia", "adc@cie", 123)
        NewBorrower.UpdateItemsOnLoan(3)
        NewBorrower.PrintDetails()
        NewBorrower.UpdateItemsOnLoan(-1)
        NewBorrower.PrintDetails()
        Console.ReadLine()
    End Sub
```

Pascal

```
  Borrower = class
      private
         BorrowerName : string;
         EmailAddress : string;
         BorrowerID : integer;
         ItemsOnLoan : integer;
      public
         constructor Create(n, e : string; b :
integer);
         function GetBorrowerName : string;
         function GetEmailAddress : string;
         function GetBorrowerID : integer;
         function GetItemsOnLoan : integer;
         procedure UpdateItemsOnLoan(n : integer);
         procedure PrintDetails;
      end;
constructor Borrower.Create(n, e : string; b :
integer);
```

```
begin
   BorrowerName := n;
   EmailAddress := e;
   BorrowerID := b;
   ItemsOnLoan := 0;
end;

function Borrower.GetBorrowerName : string;
begin
   GetBorrowerName := BorrowerName;
end;

function Borrower.GetEmailAddress : string;
begin
   GetEmailAddress := EmailAddress;
end;

function Borrower.GetBorrowerID : integer;
begin
  GetBorrowerID := BorrowerID;
end;

function Borrower.GetItemsOnLoan : integer;
begin
   GetItemsOnLoan := ItemsOnLoan;
end;

procedure Borrower.UpdateItemsOnLoan(n : integer);
begin
   ItemsOnLoan := ItemsOnLoan + n
end;

procedure Borrower.PrintDetails;
begin
   WriteLn('Borrower     : ', BorrowerName);
   WriteLn('ID           : ', BorrowerID);
   WriteLn('email        : ', EmailAddress);
   WriteLn('Items on loan: ', ItemsOnLoan);
end;
var Newborrower : Borrower;
begin
   NewBorrower := Borrower.Create('Sylvia', 'adc@cie',
123);
   NewBorrower.UpdateItemsOnLoan(3);
   NewBorrower.PrintDetails();
   NewBorrower.UpdateItemsOnLoan(-1);
   NewBorrower.PrintDetails();
   readln
end.
```

Task 27.04

Python

```
def PrintDetails(self):
    print("Book Details")
    LibraryItem.PrintDetails(self)
```

	```
	        print(self.__IsRequested)

	    def PrintDetails(self):
	        print("CD Details")
	        LibraryItem.PrintDetails(self)
	        print(self.__Genre)
	```

VB.NET	
	```
	'in base class add the keyword overridable to method to be
	redefined
	Public Overridable Sub PrintDetails()
	   Console.WriteLine(Title & "; " & ItemID & "; " & OnLoan & ";
	" & DueDate)
	End Sub

	' in subclass add the re-defined method:
	Public Overrides Sub PrintDetails()
	   Console.WriteLine("Book Details")
	   MyBase.PrintDetails()
	   Console.WriteLine(IsRequested)
	End Sub

	Public Overrides Sub PrintDetails()
	   Console.WriteLine("CD Details")
	   MyBase.PrintDetails()
	   Console.WriteLine(Genre)
	End Sub
	```

Pascal	
	```
	// in class definition add virtual:
	procedure PrintDetails; virtual;

	// in subclass implementation add method definition:
	procedure Book.PrintDetails;
	begin
	   WriteLn('Book Details');
	   inherited;
	   WriteLn( IsRequested);
	end;

	procedure CD.PrintDetails;
	begin
	   WriteLn('CD Details');
	   inherited;
	   WriteLn( Genre);
	end;
	```

Task 27.05

Python	
	```
	class LibraryItem:

	   def __init__(self, t, a, i):           #
	initialiser method
	       self.__Title = t
	       self.__Author_Artist = a
	       self.__ItemID = i
	```

```python
      self.__OnLoan = False
      self.__DueDate = datetime.date.today()
      self.__BorrowerID = 0

…
   def Borrowing(self, b):
      self.__OnLoan = True
      self.__DueDate = self.__DueDate + 
datetime.timedelta(weeks=3)
      self.__BorrowerID = b

   def PrintDetails(self):
      print(self.__Title, ' ; ', self.__Author_Artist, 
' ; ', end='')
      print(self.__ItemID, ' ; ', self.__OnLoan)
      print(self.__DueDate, ' ; Borrower: ', 
self.__BorrowerID)

def main():
   ThisBook = Book("Computing", "Sylvia", 1234)
   ThisBook.PrintDetails()
   NewBorrower = Borrower("Fred", "adc@cie", 123)
   ThisBook.Borrowing(123)
   NewBorrower.UpdateItemsOnLoan(1)
   ThisBook.PrintDetails()
   NewBorrower.PrintDetails()
```

VB.NET

```vbnet
 Class LibraryItem
        Private Title As String
        Private Author_Artist As String
        Private ItemID As Integer
        Private OnLoan As Boolean = False
        Private DueDate As Date = Today
        Private BorrowerID As Integer = 0

        Sub Create(ByVal t As String, ByVal a As String, ByVal i As 
Integer)
            Title = t
            Author_Artist = a
            ItemID = i
        End Sub

…
        Public Sub Borrowing(b As Integer)
            OnLoan = True
            DueDate = DateAdd(DateInterval.Day, 21, Today()) '3 
weeks from today
            BorrowerID = b
        End Sub

        Public Overridable Sub PrintDetails()
            Console.WriteLine(Title & "; " & ItemID & "; " & OnLoan 
& "; " & DueDate)
            Console.Writeline("Borrower: " & BorrowerID)
        End Sub
    End Class

    Sub Main()
        Dim ThisBook As New Book()
        Dim ThisBorrower As New Borrower()
```

	```
	        ThisBook.Create("Computing", "Sylvia", 1234)
	        ThisBorrower.Create("Fred", "adc@cie", 456)
	        ThisBook.PrintDetails()
	        ThisBorrower.PrintDetails()
	        ThisBook.Borrowing(456)
	        ThisBorrower.UpdateItemsOnLoan(1)
	        ThisBook.PrintDetails()
	        ThisBorrower.PrintDetails()
	        Console.ReadLine()
	    End Sub
	```

Pascal	
	```
	type
	   LibraryItem = class
	      private
	         Title : STRING;
	         Author_Artist : STRING;
	         ItemID : INTEGER;
	         OnLoan : BOOLEAN;
	         DueDate : TDATETIME;
	         BorrowerID : integer;
	      public
	         constructor Create(t, a : STRING; i :
	INTEGER);  virtual;
	         function GetTitle : STRING;
	         function GetAuthor_Artist : STRING;
	         function GetItemID : INTEGER;
	         function GetOnLoan : BOOLEAN;
	         function GetDueDate : TDATETIME;
	         procedure Borrowing(b : integer);
	         procedure Returning;
	         procedure PrintDetails; virtual;
	      end;

	constructor LibraryItem.Create(t, a : STRING; i :
	INTEGER);
	begin
	   Title := t;
	   Author_Artist := a;
	   ItemID := i;
	   OnLoan := FALSE;
	   DueDate := 0;
	   BorrowerID := 0;
	end;

	procedure LibraryItem.Borrowing(b : integer);
	begin
	  OnLoan := TRUE;
	  DueDate := Date() + 21;
	  BorrowerID := b;
	end;

	procedure LibraryItem.PrintDetails;
	begin
	  WriteLn(Title, ' ; ', ItemID:7, ' ; ', OnLoan);
	  WriteLn(DateToStr(DueDate),'  Borrower: ',
	BorrowerID)
	```

```
end;

var ThisBook : Book; Newborrower : Borrower;
begin
   NewBorrower := Borrower.Create('Fred', 'adc@cie',
123);
   NewBorrower.PrintDetails();
   ThisBook := Book.Create('Computing', 'SL & DD',
111);
   ThisBook.PrintDetails();
   ThisBook.Borrowing(123);
   NewBorrower.UpdateItemsOnLoan(1);
   ThisBook.PrintDetails();
   NewBorrower.PrintDetails();
   readln
end.
```

Task 27.06

Python

```
class Book(LibraryItem):

   def __init__(self, t, a, i):           #
initialiser method
      LibraryItem.__init__(self, t, a, i)
      self.__IsRequested = False
      self.__RequestedBy = 0


   def GetIsRequested(self):
      return(self.__IsRequested)

   def SetIsRequested(self, b):
      self.__IsRequested = True
      self.__RequestedBy = b

   # print details method for Book

   def PrintDetails(self):
      print("Book Details")
      LibraryItem.PrintDetails(self)
      if self.__IsRequested :
         print('Requested by ', self.__RequestedBy)
      else :
         print('no requests')

   ThisBook.SetIsRequested(345)
   ThisBook.PrintDetails()
```

VB.NET

```
   Class Book
        Inherits LibraryItem
        Private IsRequested As Boolean = False
        Private RequestedBy As Integer

        '  Overrides Sub Create(ByVal t As String, ByVal a As String,
ByVal i As Integer)
```

```
        '      MyBase.Create(t, a, i)
        ' End Sub

        Public Function GetIsRequested() As Boolean
            Return (IsRequested)
        End Function

        Public Sub SetIsRequested(b As Integer)
            IsRequested = True
            RequestedBy = b
        End Sub
        ' in subclass add the re-defined method:
        Public Overrides Sub PrintDetails()
            Console.WriteLine("Book Details")
            MyBase.PrintDetails()
            Console.WriteLine(IsRequested)
            Console.WriteLine("Requested by " & RequestedBy)
        End Sub
    End Class

    Sub Main()
        Dim ThisBook As New Book()
        ThisBook.Create("Computing", "Sylvia", 1234)
        ThisBook.PrintDetails()
        ThisBook.SetIsRequested(890)
        ThisBook.PrintDetails()
        Console.ReadLine()
    End Sub
```

Pascal

```
   Book = class(LibraryItem)
      private
         IsRequested : BOOLEAN;
         RequestedBy : integer;
      public
         constructor Create(t, a : STRING; i :
INTEGER);  override;
         function GetIsRequested : BOOLEAN;
         procedure SetIsRequested(b : integer);
         procedure PrintDetails; override;
      end;

procedure Book.SetIsRequested(b : integer);
begin
  IsRequested := TRUE;
  RequestedBy := b;
end;

function Book.GetIsRequested : BOOLEAN;
begin
   GetIsRequested := IsRequested;
end;

procedure Book.PrintDetails;
begin
   WriteLn('Book Details');
   inherited;
   if IsRequested
     then writeln('Requested by ', RequestedBy)
```

	```
	      else Writeln('no requests');
	end;
	…
	   ThisBook.SetIsRequested(123);
	   ThisBook.PrintDetails();
	```

Task 27.07

Python

```
import datetime

# Borrower class ***************************************************
class TBorrower:
   def __init__(self, n, e, i):   # constructor
      self.__BorrowerName = n
      self.__EmailAddress = e
      self.__BorrowerID = i
      self.__ItemsOnLoan = 0

   def getBorrowerName(self):
      return(self.__BorrowerName)

   def getEmailAddress(self):
      return(self.__EmailAddress)

   def getBorrowerID(self):
      return(self.__BorrowerID)

   def getItemsOnLoan(self):
      return(self.__ItemsOnLoan)

   def updateItemsOnLoan(self, n):
      self.__ItemsOnLoan = self.__ItemsOnLoan + n

   def printDetails(self):
      print(self.__BorrowerName, ';', self.__BorrowerID, ';', end='')
      print(self.__EmailAddress, ';', self.__ItemsOnLoan)



# Library Item class ***********************************************
class TLibraryItem:

   def __init__(self, t, a, i):          # initialiser method
      self.__Title = t
      self.__Author_Artist = a
      self.__ItemID = i
      self.__OnLoan = False
      self.__BorrowerID = 0
      self.__DueDate = datetime.date.today()

   def getTitle(self):
      return(self.__Title)

   def getAuthor_Artist(self):
      return(self.__Author_Artist)

   def getItemID(self):
      return(self.__getItemID)

   def getOnLoan(self):
      return(self.__OnLoan)

   def getBorrowerID(self):
      return(self.__BorrowerID)

   def getDueDate(self):
      return(self.__DueDate)

   def Borrowing(self, i, x):
      if  x.getItemsOnLoan() < 5:
         self.__OnLoan = True
         self.__BorrowerID = x.getBorrowerID()
         self.__DueDate = self.__DueDate + datetime.timedelta(weeks=3)
         x.updateItemsOnLoan(1)
      else:
         print("too many books on loan")

   def Returning(self, i, x):
      self.__OnLoan = False
      x.updateItemsOnLoan(-1)

   def printDetails(self):
      print(self.__Title, ';', self.__Author_Artist, ';', end =' ')
      print(self.__ItemID, ';', self.__OnLoan, ';', end = ' ')
      print(self.__BorrowerID, ';', self.__DueDate)

# Book class ****************************************************
class TBook(TLibraryItem):

   def __init__(self, t, a, i):           # initialiser method
      TLibraryItem.__init__(self, t, a, i)
      self.__IsRequested = False
      self.__RequestedBy = 0


   def getIsRequested(self):
      return(self.__IsRequested)

   def getRequestedBy(self):
      return(self.__RequestedBy)

   def RequestBook(self, i, x):
      self.__IsRequested = True
      self.__RequestedBy = x.getBorrowerID()

   def printDetails(self):
      TLibraryItem.printDetails(self)
      print(self.__IsRequested, ';', self.__RequestedBy)
```

```
# CD class ************************************************
class T_CD(TLibraryItem):

   def __init__(self, t, a, i):           # initialiser method
      TLibraryItem.__init__(self, t, a, i)
      self.__Genre = ""

   def getGenre(self):
      return(self.__Genre)

   def SetGenre(self, g):
      self.__Genre = g

   def printDetails(self):
      TLibraryItem.printDetails(self)
      print(self.__Genre)




# Display menu *****************************************************
def DisplayMenu():
   print('1 - Add a new borrower')
   print('2 - Add a new book')
   print('3 - Add a new CD')
   print('4 - Borrow book')
   print('5 - Return book')
   print('6 - Borrow CD')
   print('7 - Return CD')
   print('8 - Request book')
   print('9 - Print all details')
   print('99 - Exit program')
   print
   print('Enter your menu choice: ')


# main program *****************************************************

def main():
   Finish = False
   NextBorrowerID = 1
   NextBookID = 1
   NextCD_ID = 1

   while Finish == False:
      DisplayMenu()
      MenuChoice = int(input())
      if MenuChoice == 1:     # new borrower
         BName = input("Name: ")
         Email = input("email address: ");
         BorrowerID = NextBorrowerID
         NextBorrowerID = NextBorrowerID + 1
         Borrower = TBorrower(BName, Email, BorrowerID)
      elif MenuChoice == 2:     # new book
         Title = input("Title: ")
```

```python
         Author = input("Author: ")
         ItemID = NextBookID
         NextBookID = NextBookID + 1
         Book = TBook(Title, Author, ItemID)
      elif MenuChoice == 3:     # new CD
         Title = input("Title: ")
         Artist = input("Artist: ")
         ItemID = NextCD_ID
         NextCD_ID = NextCD_ID + 1
         CD = T_CD(Title, Artist, ItemID)
      elif MenuChoice == 4:     # borrow book
         BorrowerID = input("Borrower ID: ")
         ItemID = input("Book ID: ")
         Book.Borrowing(ItemID, Borrower)
      elif MenuChoice == 5:     # return book
         BorrowerID = input("Borrower ID: ")
         ItemID = input("Book ID: ")
         Book.Returning(ItemID, Borrower)
      elif MenuChoice == 6:     # borrow CD
         BorrowerID = input("Borrower ID: ")
         ItemID = input("CD ID: ")
         CD.Borrowing(ItemID, Borrower)
      elif MenuChoice == 7:     # return CD
         BorrowerID = input("Borrower ID: ")
         ItemID = input("CD ID: ")
         CD.Returning(ItemID, Borrower)
      elif MenuChoice == 8:     # request book
         BorrowerID = input("Borrower ID: ")
         ItemID = input("Book ID: ")
         Book.RequestBook(ItemID, Borrower)
      elif MenuChoice == 9:     # print all details
         print("Borrower Details")
         Borrower.printDetails()
         print("Book Details")
         Book.printDetails()
         print("CD Details")
         CD.printDetails()
      elif MenuChoice == 99:    # end program
         Finish = True
      else:
         print("wrong input")
   input()

main()
```

VB.NET

```vb
Module Module1

    Class TBorrower
        Private BorrowerName As String
        Private BorrowerID As Integer
        Private EmailAddress As String
        Private ItemsOnLoan As Integer = 0

        Public Sub Create(ByVal n As String, ByVal e As String, ByVal i As Integer)
            BorrowerName = n
            EmailAddress = e
            BorrowerID = i
        End Sub
        Public Function GetBorrowerName() As String
            Return (BorrowerName)
        End Function
        Public Function GetBorrowerID() As Integer
            Return (BorrowerID)
        End Function
        Public Function GetEmailAddress() As String
            Return (EmailAddress)
        End Function
        Public Function GetItemsOnLoan() As Integer
            Return (ItemsOnLoan)
        End Function
        Public Sub UpdateItemsOnLoan(ByVal n As Integer)
            ItemsOnLoan = ItemsOnLoan + n
        End Sub
        Public Sub PrintDetails()
            Console.WriteLine("Borrower Details")
            Console.Write(BorrowerName & ";" & BorrowerID & ";")
            Console.WriteLine(EmailAddress & ";" & ItemsOnLoan)
            Console.WriteLine()
        End Sub
    End Class

    Class TLibraryItem
        Private Title As String
        Private Author_Artist As String
        Private ItemID As Integer
        Private OnLoan As Boolean = False
        Private BorrowerID As Integer = 0
        Private DueDate As Date = Today
        Overridable Sub Create(ByVal t As String, ByVal a As String, ByVal i As Integer)
            Title = t
            Author_Artist = a
            ItemID = i
        End Sub
        Public Function GetTitle() As String
            Return (Title)
        End Function
        Public Function GetItemID() As Integer
            Return (ItemID)
        End Function
        Public Function GetOnLoan() As Boolean
            Return (OnLoan)
        End Function
        Public Function GetBorrowerID() As Integer
            Return (BorrowerID)
        End Function
        Public Function GetDueDate() As Date
            Return (DueDate)
        End Function
        Public Sub Borrowing(ByVal i As Integer, ByVal x As TBorrower)
            If x.GetItemsOnLoan < 5 Then
                OnLoan = True
                BorrowerID = x.GetBorrowerID()
                DueDate = DateAdd(DateInterval.Day, 21, Today())
                x.UpdateItemsOnLoan(1)
            Else
                Console.WriteLine("too many books on loan")
            End If
        End Sub
        Public Sub Returning(ByVal i As Integer, ByVal x As TBorrower)
```

```
            OnLoan = False
            x.UpdateItemsOnLoan(-1)
        End Sub
        Overridable Sub PrintDetails()
            Console.Write(Title & ";" & ItemID & ";" & OnLoan & ";")
            Console.WriteLine( BorrowerID & ";" & DueDate)
        End Sub
    End Class

    Class TBook
        Inherits TLibraryItem
        Private Author As String
        Private IsRequested As Boolean = False
        Private RequestedBy As Integer = 0

        Public Function GetIsRequested() As Boolean
            Return (IsRequested)
        End Function
        Public Function GetRequestedBy() As Integer
            Return (RequestedBy)
        End Function
        Public Sub RequestBook(ByVal i As Integer, ByVal x As TBorrower)
            IsRequested = True
            RequestedBy = x.GetBorrowerID()
        End Sub
        Public Overrides Sub PrintDetails()
            Console.WriteLine("Book Details")
            MyBase.PrintDetails()
            Console.WriteLine(IsRequested & ";" & RequestedBy)
            Console.WriteLine()
        End Sub
    End Class

    Class T_CD
        Inherits TLibraryItem
        Private Genre As String

        Overrides Sub Create(ByVal t As String, ByVal a As String, ByVal i As Integer)
            MyBase.Create(t, a, i)
        End Sub
        Public Function GetGenre() As String
            Return (Genre)
        End Function
        Public Sub SetGenre(ByVal g As String)
            Genre = g
        End Sub
        Overrides Sub PrintDetails()
            Console.WriteLine("CD Details")
            MyBase.PrintDetails()
            Console.WriteLine(Genre)
            Console.WriteLine()
        End Sub
    End Class

    Dim Borrower As New TBorrower
    Dim Book As New TBook
    Dim CD As New T_CD

    Dim NextBorrowerID, NextBookID, NextCD_ID As Integer
    Dim Finish As Boolean

    Sub DisplayMenu()
```

```
        Console.WriteLine("1 - Add a new borrower")
        Console.WriteLine("2 - Add a new book")
        Console.WriteLine("3 - Add a new CD")
        Console.WriteLine("4 - Borrow book")
        Console.WriteLine("5 - Return book")
        Console.WriteLine("6 - Borrow CD")
        Console.WriteLine("7 - Return CD")
        Console.WriteLine("8 - Request book")
        Console.WriteLine("9 - Print all details")
        Console.WriteLine("99 - Exit program")
        Console.WriteLine()
        Console.Write("Enter your menu choice: ")
    End Sub

    Sub ProcessMenuChoice()
        Dim MenuChoice, ItemID, BorrowerID As Integer
        Dim BName, Email, Title, Author, Artist As String

        MenuChoice = Console.ReadLine()
        Select Case MenuChoice
            Case 1                 'new borrower
                Console.Write("Name: ")
                BName = Console.ReadLine()
                Console.Write("email address: ")
                Email = Console.ReadLine()
                BorrowerID = NextBorrowerID
                NextBorrowerID = NextBorrowerID + 1
                Borrower.Create(BName, Email, BorrowerID)
            Case 2                  ' new book
                Console.Write("Title: ")
                Title = Console.ReadLine()
                Console.Write("Author: ")
                Author = Console.ReadLine()
                ItemID = NextBookID
                NextBookID = NextBookID + 1
                Book.Create(Title, Author, ItemID)
            Case 3                     ' new CD
                Console.Write("Title: ")
                Title = Console.ReadLine()
                Console.Write("Artist: ")
                Artist = Console.ReadLine()
                ItemID = NextCD_ID
                NextCD_ID = NextCD_ID + 1
                CD.Create(Title, Artist, ItemID)
            Case 4                     ' borrow book
                Console.Write("Borrower ID: ")
                BorrowerID = Console.ReadLine()
                Console.Write("Book ID: ")
                ItemID = Console.ReadLine()
                Book.Borrowing(ItemID, Borrower)
            Case 5                     ' return book
                Console.Write("Borrower ID: ")
                BorrowerID = Console.ReadLine()
                Console.Write("Book ID: ")
                ItemID = Console.ReadLine()
                Book.Returning(ItemID, Borrower)
            Case 6                     ' borrow CD
                Console.Write("Borrower ID: ")
                BorrowerID = Console.ReadLine()
                Console.Write("CD ID: ")
                ItemID = Console.ReadLine()
                CD.Borrowing(ItemID, Borrower)
            Case 7                  ' return CD
                Console.Write("Borrower ID: ")
                BorrowerID = Console.ReadLine()
                Console.Write("CD ID: ")
                ItemID = Console.ReadLine()
                CD.Returning(ItemID, Borrower)
            Case 8                  ' request book
                Console.Write("Borrower ID: ")
                BorrowerID = Console.ReadLine()
                Console.Write("Book ID: ")
                ItemID = Console.ReadLine()
                Book.RequestBook(ItemID, Borrower)
            Case 9                  ' print all details
                Borrower.PrintDetails()
                Book.PrintDetails()
                CD.PrintDetails()
            Case 99
                Finish = True
            Case Else
                Console.WriteLine("wrong input")
        End Select
    End Sub

    Sub Main()
        Finish = False
        NextBorrowerID = 1
        NextBookID = 1
        NextCD_ID = 1
        Do
            DisplayMenu()
            ProcessMenuChoice()
        Loop Until Finish
        Console.ReadLine()
    End Sub

End Module
```

Pascal

```
program OOP2;

{$APPTYPE CONSOLE}

uses
  SysUtils;

type TBorrower =    class
        private
           BorrowerName : STRING;
           BorrowerID   : INTEGER;
           EmailAddress : STRING;
           ItemsOnLoan  : INTEGER;
        public
           constructor Create(n, e : STRING; i : INTEGER);
           function GetBorrowerName : STRING;
           function GetBorrowerID   : INTEGER;
           function GetEmailAddress : STRING;
           function GetItemsOnLoan  : INTEGER;
           procedure UpdateItemsOnLoan(n : INTEGER);
           procedure PrintDetails;
     end;

     TLibraryItem = class
        private
```

```
            Title : STRING;
            ItemID : INTEGER;
            OnLoan : BOOLEAN;
            BorrowerID : INTEGER;
            DueDate : TDATETIME;
         public
            constructor Create(t : STRING; i : INTEGER);   virtual;
            function GetTitle : STRING;
            function GetItemID : INTEGER;
            function GetOnLoan : BOOLEAN;
            function GetBorrowerID : INTEGER;
            function GetDueDate : TDATETIME;
            procedure Borrowing(i : INTEGER; x : TBorrower);
            procedure Returning(i : INTEGER; x : TBorrower);
            procedure PrintDetails; virtual;
      end;

      TBook = class(TLibraryItem)
         private
            Author : STRING;
            ISBN : STRING;
            IsRequested : BOOLEAN;
            RequestedBy : INTEGER;
         public
            constructor Create(t : STRING; i : INTEGER);  override;
            function GetAuthor : STRING;
            function GetISBN : STRING;
            function GetIsRequested : BOOLEAN;
            function GetRequestedBy : INTEGER;
            procedure RequestBook(i : INTEGER; x : TBorrower);
            procedure PrintDetails; override;
      end;

      T_CD = class(TLibraryItem)
         private
            Genre : STRING;
         public
            constructor Create(t : STRING; i : INTEGER);  override;
            function GetGenre : STRING;
            procedure SetGenre(g : STRING);
            procedure PrintDetails; override;
      end;

{ ****  TBorrower Methods ***************************}

constructor TBorrower.Create(n, e : STRING; i : INTEGER);
begin
   BorrowerName := n;
   EmailAddress := e;
   BorrowerID   := i;
   ItemsOnLoan := 0;
end;
function TBorrower.GetBorrowerName : STRING;
begin
   GetBorrowerName := BorrowerName;
end;
function TBorrower.GetBorrowerID   : INTEGER;
begin
   GetBorrowerID := BorrowerID;
end;
function TBorrower.GetEmailAddress : STRING;
```

```
begin
   GetEmailAddress := EmailAddress;
end;
function TBorrower.GetItemsOnLoan : INTEGER;
begin
   GetItemsOnLoan := ItemsOnLoan;
end;
procedure TBorrower.UpdateItemsOnLoan(n : INTEGER);
begin
   ItemsOnLoan := ItemsOnLoan + n;
end;
procedure TBorrower.PrintDetails;
begin
   WriteLn('Borrower Details');
   Write(BorrowerName, ';', BorrowerID, ';');
   WriteLn(EmailAddress, ';', ItemsOnLoan);
   WriteLn;
end;

{ ****  TLibraryItem Methods ***************************}

constructor TLibraryItem.Create(t : STRING; i : INTEGER);
begin
   Title := t;
   ItemID := i;
   OnLoan := FALSE;
   BorrowerID := 0;
   DueDate := 0;
end;
function TLibraryItem.GetTitle : STRING;
begin
   GetTitle := Title;
end;
function TLibraryItem.GetItemID   : INTEGER;
begin
   GetItemID := ItemID;
end;
function TLibraryItem.GetOnLoan : BOOLEAN;
begin
   GetOnLoan := OnLoan;
end;
function TLibraryItem.GetBorrowerID : INTEGER;
begin
   GetBorrowerID := BorrowerID;
end;
function TLibraryItem.GetDueDate : TDATETIME;
begin
   GetDueDate := DueDate;
end;
procedure TLibraryItem.Borrowing(i : INTEGER; x : TBorrower);
begin
  if x.GetItemsOnLoan <  5
     then
        begin
           OnLoan := TRUE;
           BorrowerID := x.GetBorrowerID;
           DueDate := Date() + 21;
           x.UpdateItemsOnLoan(1);
        end
     else
        WriteLn('too many books on loan');
```

```
end;
procedure TLibraryItem.Returning(i : INTEGER; x : TBorrower);
begin
   OnLoan := FALSE;
   x.UpdateItemsOnLoan(-1);
end;
procedure TLibraryItem.PrintDetails;
begin
   Write(Title, ';', ItemID, ';', OnLoan, ';');
   WriteLn(BorrowerID, ';', DateToStr(DueDate));
end;

{ ****  TBook Methods ***************************}

constructor TBook.Create(t{, a, b} : STRING; i : INTEGER);
begin
   inherited Create(t, i);
   //Author := a;
   //ISBN := b;
   IsRequested := FALSE;
   RequestedBy := 0;

end;
function TBook.GetAuthor : STRING;
begin
   GetAuthor := Author;
end;
function TBook.GetISBN : STRING;
begin
   GetISBN := ISBN;
end;
function TBook.GetIsRequested : BOOLEAN;
begin
   GetIsRequested := IsRequested;
end;
function TBook.GetRequestedBy : INTEGER;
begin
   GetRequestedBy := RequestedBy;
end;
procedure TBook.RequestBook(i : INTEGER; x : TBorrower);
begin
  IsRequested := TRUE;
  RequestedBy := x.GetBorrowerID;
end;
procedure TBook.PrintDetails;
begin
   WriteLn('Book Details');
   inherited;
   WriteLn(Author, ';', ISBN, ';', IsRequested, ';', RequestedBy);
   WriteLn;
end;
{ ****  T_CD Methods ***************************}

constructor T_CD.Create(t{, g} : STRING; i : INTEGER);
begin
   inherited Create(t, i);
   //Genre := g;

end;

function T_CD.GetGenre : STRING;
```

```
begin
   GetGenre := Genre;
end;

procedure T_CD.SetGenre(g : STRING);
begin
   Genre := g;
end;

procedure T_CD.PrintDetails;
begin
   WriteLn('CD Details');
   inherited;
   WriteLn(Genre);
   WriteLn;
end;
{**************** procedure declarations *******************}
var Borrower : array[1..20] of TBorrower;
    Book : array[1..100] of TBook;
    CD : array[1..20] of T_CD;
    NextBorrowerID, NextBookID, NextCD_ID : INTEGER;
    Finish : BOOLEAN;

procedure DisplayMenu;
begin
   WriteLn('1 - Add a new borrower');
   WriteLn('2 - Add a new book');
   WriteLn('3 - Add a new CD');
   WriteLn('4 - Borrow book');
   WriteLn('5 - Return book');
   WriteLn('6 - Borrow CD');
   WriteLn('7 - Return CD');
   WriteLn('8 - Request book');
   WriteLn('9 - Print all details');
   WriteLn('99 - Exit program');
   WriteLn;
   Write('Enter your menu choice: ');
end;

procedure ProcessMenuChoice;
var MenuChoice, ItemID, BorrowerID, i : INTEGER;
    BName, Email, Title, Author, ISBN, Genre : STRING;
begin
  ReadLn(MenuChoice);
    case MenuChoice of
      1: begin   // new borrower
         Write('Name: '); ReadLn(BName);
         Write('email address: '); ReadLn(Email);
         BorrowerID := NextBorrowerID;
         NextBorrowerID := NextBorrowerID + 1;
         Borrower[BorrowerID] := TBorrower.Create(BName, Email, BorrowerID);
         end;
      2: begin   // new book
         Write('Title: '); ReadLn(Title);
         Write('Author: '); ReadLn(Author);
         Write('ISBN: '); ReadLn(ISBN);
         ItemID := NextBookID; NextBookID := NextBookID + 1;
         Book[ItemID] := TBook.Create(Title, {Author, ISBN,} ItemID);
         end;
      3: begin  // new CD
         Write('Title: '); ReadLn(Title);
```

```
          Write('Genre: '); ReadLn(Genre);
          ItemID := NextCD_ID; NextCD_ID := NextCD_ID + 1;
          CD[ItemID] := T_CD.Create(Title, {Genre,} ItemID);
          end;
       4: begin  // borrow book
          Write('Borrower ID: '); ReadLn(BorrowerID);
          Write('Book ID: '); ReadLn(ItemID);
          Book[ItemID].Borrowing(ItemID, Borrower[BorrowerID]);
          end;
       5: begin   // return book
          Write('Borrower ID: '); ReadLn(BorrowerID);
          Write('Book ID: '); ReadLn(ItemID);
          Book[ItemID].Returning(ItemID, Borrower[BorrowerID]);
          end;
       6: begin   // borrow CD
          Write('Borrower ID: '); ReadLn(BorrowerID);
          Write('CD ID: '); ReadLn(ItemID);
          CD[ItemID].Borrowing(ItemID, Borrower[BorrowerID]);
          end;
       7: begin   // return CD
          Write('Borrower ID: '); ReadLn(BorrowerID);
          Write('CD ID: '); ReadLn(ItemID);
          CD[ItemID].Returning(ItemID, Borrower[BorrowerID]);
          end;
       8: begin  // request book
          Write('Borrower ID: '); ReadLn(BorrowerID);
          Write('Book ID: '); ReadLn(ItemID);
          Book[ItemID].RequestBook(ItemID, Borrower[BorrowerID]);
          end;
       9: begin  // print all details
          for i := 1 to NextBorrowerID - 1 do
          Borrower[i].PrintDetails;
          for i := 1 to NextBookID - 1 do
          Book[i].PrintDetails;
          for i := 1 to NextCD_ID - 1 do
          CD[i].PrintDetails;
          end;
      99: Finish := TRUE;
     else WriteLn('wrong input');
     end;
end;


{********** main program *************************************}
begin
   Finish := FALSE;
   NextBorrowerID := 1;
   NextBookID := 1;
   NextCD_ID := 1;

   repeat
      DisplayMenu;
      ProcessMenuChoice;
   until Finish;
   ReadLn;
end.
```

Task 27.08

Python

```
class Assessment:
   def __init__(self, t, m):
      self.__AssessmentTitle = t
      self.__MaxMarks = m

   def OutputAssessmentDetails(self):
      print(self.__AssessmentTitle, "  Marks: ",
self.__MaxMarks)


class Course:
   def __init__(self, t, m): # sets up a new course
      self.__CourseTitle = t
      self.__MaxStudents = m
      self.__NumberOfLessons = 0
      self.__CourseLesson = []
      self.__CourseAssessment = Assessment

   def AddLesson(self, t, d, r):
      self.__NumberOfLessons = self.__NumberOfLessons
+ 1
      self.__CourseLesson.append(Lesson(t, d, r))

   def AddAssessment(self, t, m):
      CourseAssessment = Assessment(t, m)


   def OutputCourseDetails(self):
      print(self.__CourseTitle, end=' ')
      print( "Maximum number of students: ",
self.__MaxStudents)
      for i in range(self.__NumberOfLessons):

print(self.__CourseLesson[i].OutputLessonDetails())

class Lesson:
   def __init__(self, t, d, r):
      self.__LessonTitle = t
      self.__DurationMinutes = d
      self.__requiresLab = r

   def OutputLessonDetails(self):
      print(self.__LessonTitle,
self.__DurationMinutes)

def Main():
   MyCourse = Course("Computing", 10) # sets up a new
course

   MyCourse.AddAssessment("Programming", 100) # adds
an assignment

   # add 3 lessons
```

```
    MyCourse.AddLesson("Problem Solving", 60, False)
    MyCourse.AddLesson("Programming", 120, True)
    MyCourse.AddLesson("Theory", 60, False)

    # check it all works
    MyCourse.OutputCourseDetails()


Main()
```

VB.NET

```
Module Module1

    Class Assessment
        Private AssessmentTitle As String
        Private MaxMarks As Integer

        Public Sub Create(ByVal t As String, ByVal m As Integer)
            AssessmentTitle = t
            MaxMarks = m
        End Sub

        Public Sub OutputAssessmentDetails()
            Console.Write(AssessmentTitle & "Marks: " & MaxMarks)
        End Sub
    End Class

    Class Lesson
        Private LessonTitle As String
        Private DurationMinutes As Integer
        Private RequiresLab As Boolean

        Public Sub Create(ByVal t As String, ByVal d As Integer,
ByVal r As Boolean)
            LessonTitle = t
            DurationMinutes = d
            RequiresLab = r
        End Sub

        Public Sub OutputLessonDetails()
            Console.WriteLine(LessonTitle & "  " & DurationMinutes)
        End Sub
    End Class

    Class Course
        Private CourseTitle As String
        Private MaxStudents As Integer
        Private NumberOfLessons As Integer = 0
        Private CourseLesson(50) As Lesson
        Private CourseAssessment As Assessment

        Public Sub Create(ByVal t As String, ByVal m As Integer)
            CourseTitle = t
            MaxStudents = m
        End Sub

        Sub AddLesson(ByVal t As String, ByVal d As Integer, ByVal r
As Boolean)
            NumberOfLessons = NumberOfLessons + 1
            CourseLesson(NumberOfLessons) = New Lesson
            CourseLesson(NumberOfLessons).Create(t, d, r)
        End Sub
        Public Sub AddAssessment(ByVal t As String, ByVal m As
Integer)
            CourseAssessment = New Assessment
            CourseAssessment.Create(t, m)
        End Sub


        Public Sub OutputCourseDetails()
            Console.Write(CourseTitle)
            Console.WriteLine("Maximum number of students: " &
MaxStudents)
            For i = 1 To NumberOfLessons
                CourseLesson(i).OutputLessonDetails()
            Next
        End Sub
    End Class

    Sub Main()
        Dim MyCourse As New Course
        MyCourse.Create("Computing", 10) ' sets up a new course

        MyCourse.AddAssessment("Programming", 100) ' adds an
assessment

        ' add 3 lessons
        MyCourse.AddLesson("Problem Solving", 60, False)
        MyCourse.AddLesson("Programming", 120, True)
        MyCourse.AddLesson("Theory", 60, False)

        'check it all works
        MyCourse.OutputCourseDetails()
        Console.ReadLine()
    End Sub

End Module
```

Pascal

```
program Project2;

{$APPTYPE CONSOLE}

uses
  SysUtils;

type
Lesson = class
   private
      LessonTitle : string;
      DurationMinutes : integer;
      RequiresLab : boolean;
   public
      constructor Create(t : string; d : integer; r :
boolean);
      procedure OutputLessonDetails;
end;

Assessment = class
   private
      AssessmentTitle : string;
```

	```
	      MaxMarks : integer;
	   public
	      constructor Create(t : string; m : integer);
	      procedure OutputAssessmentDetails;
	end;

	Course = class
	   private
	      CourseTitle : string;
	      MaxStudents : integer;
	      NumberOfLessons : integer;
	      CourseLesson : Array[1..50] of Lesson;
	      CourseAssessment : Assessment;

	   public
	      Constructor Create(t : string; m : integer);
	      procedure AddLesson(t : string; d : integer; r :
	boolean);
	      procedure AddAssessment(t : string; m :
	integer);
	      procedure OutputCourseDetails;
	end;

	// *** class implementation starts here *******
	constructor Lesson.Create(t : string; d : integer; r :
	boolean);
	begin
	   LessonTitle := t;
	   DurationMinutes := d;
	   RequiresLab := r;
	end;

	procedure Lesson.OutputLessonDetails;
	begin
	   WriteLn(LessonTitle, DurationMinutes);
	end;

	constructor Assessment.Create(t : string; m :
	integer);
	begin
	   AssessmentTitle := t;
	   MaxMarks := m;
	end;

	procedure Assessment.OutputAssessmentDetails;
	begin
	   WriteLn(AssessmentTitle, 'Marks: ', MaxMarks);
	end;

	constructor Course.Create(t : string; m : integer);
	   begin
	      CourseTitle := t;
	      MaxStudents := m;
	   end;
	```

```
procedure Course.AddLesson(t : string; d : integer; r
: boolean);
begin
   NumberOfLessons := NumberOfLessons + 1;
   CourseLesson[NumberOfLessons] := Lesson.Create(t,
d, r);
end;

procedure Course.AddAssessment(t : string; m :
integer);
begin
   CourseAssessment := Assessment.Create(t, m);
end;

procedure Course.OutputCourseDetails;
var i : integer;
begin
   Write(CourseTitle, '  Maximum number of students:
');
   Writeln(MaxStudents);
   For i := 1 to NumberOfLessons do
   WriteLn(CourseLesson[i].LessonTitle);
end;

// ************ main program starts here *********
var MyCourse : Course;
begin
  // sets up a new course
  MyCourse := Course.Create('Computing', 10);

  MyCourse.AddAssessment('Programming', 100); // adds
an assessment

  // add 3 lessons
  MyCourse.AddLesson('Problem Solving',60, FALSE);
  MyCourse.AddLesson('Programming',120, TRUE);
  MyCourse.AddLesson('Theory',60, FALSE);

  // check it all works
  MyCourse.OutputCourseDetails;
  ReadLn;
  MyCourse.Free;  // free memory
end.
```

Exam-style Questions

1 a

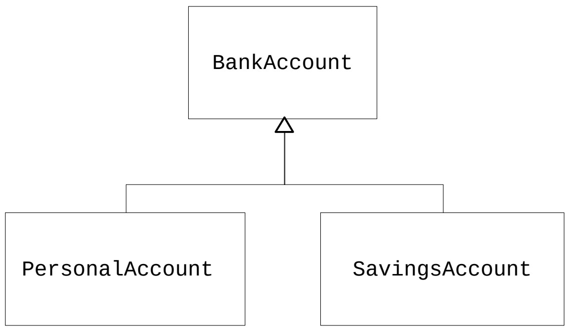

Figure 27.01

b

Python

```
class BankAccount :

   def __init__(self, i):           # initialiser
method
      self.__AccountHolderName = ''
      self.__IBAN = i

   def SetAccountHolderName(self, n):
      self.__AccountHolderName = n

   def GetAccountHolderName(self):
      return(self.__AccountHolderName)

   def GetIBAN(self):
      return(self.__IBAN)
```

VB.NET

```
    Class BankAccount
        Private AccountHolderName As String
        Private IBAN As Integer

        Sub Create(ByVal i As Integer)
            AccountHolderName = ""
            IBAN = i
        End Sub

        Public Sub SetAccountHolderName(ByVal n As String)
            AccountHolderName = n
        End Sub

        Public Function GetAccountHolderName() As String
            Return (AccountHolderName)
        End Function

        Public Function GetIBAN() As Integer
            Return (IBAN)
        End Function
    End Class
```

Pascal

```
type
   BankAccount = class
      private
         AccountHolderName : STRING;
         IBAN : INTEGER;
      public
         constructor Create(i : INTEGER);  virtual;
         function GetAccountHolderName : STRING;
         function GetIBAN : INTEGER;
         procedure SetAccountHolderName(n : string);
      end;

// implementation of methods

constructor BankAccount.Create(i : INTEGER);
begin
   AccountHolderName := '';
   IBAN := i;
end;

function BankAccount.GetAccountHolderName : STRING;
begin
   GetAccountHolderName := AccountHolderName;
end;

function BankAccount.GetIBAN : INTEGER;
begin
   GetIBAN:= IBAN;
end;

procedure BankAccount.SetAccountHolderName(n :
string);
begin
  AccountHolderName := n;
end;
```

c i

attributes for `PersonalAccount: MonthlyFee, OverDraftLimit`

methods: `Constructor, SetOverDraftLimit, GetOverDraftLimit, GetMonthlyFee`

ii

attributes for `SavingsAccount:  InterestRate`

methods: `Constructor, GetInterestRate, CalculateInterest`

iii encapsulation

2 a

Complete the class diagram showing the appropriate properties and methods.

SeasonTicketHolder
PRIVATE TicketHolderName: STRING

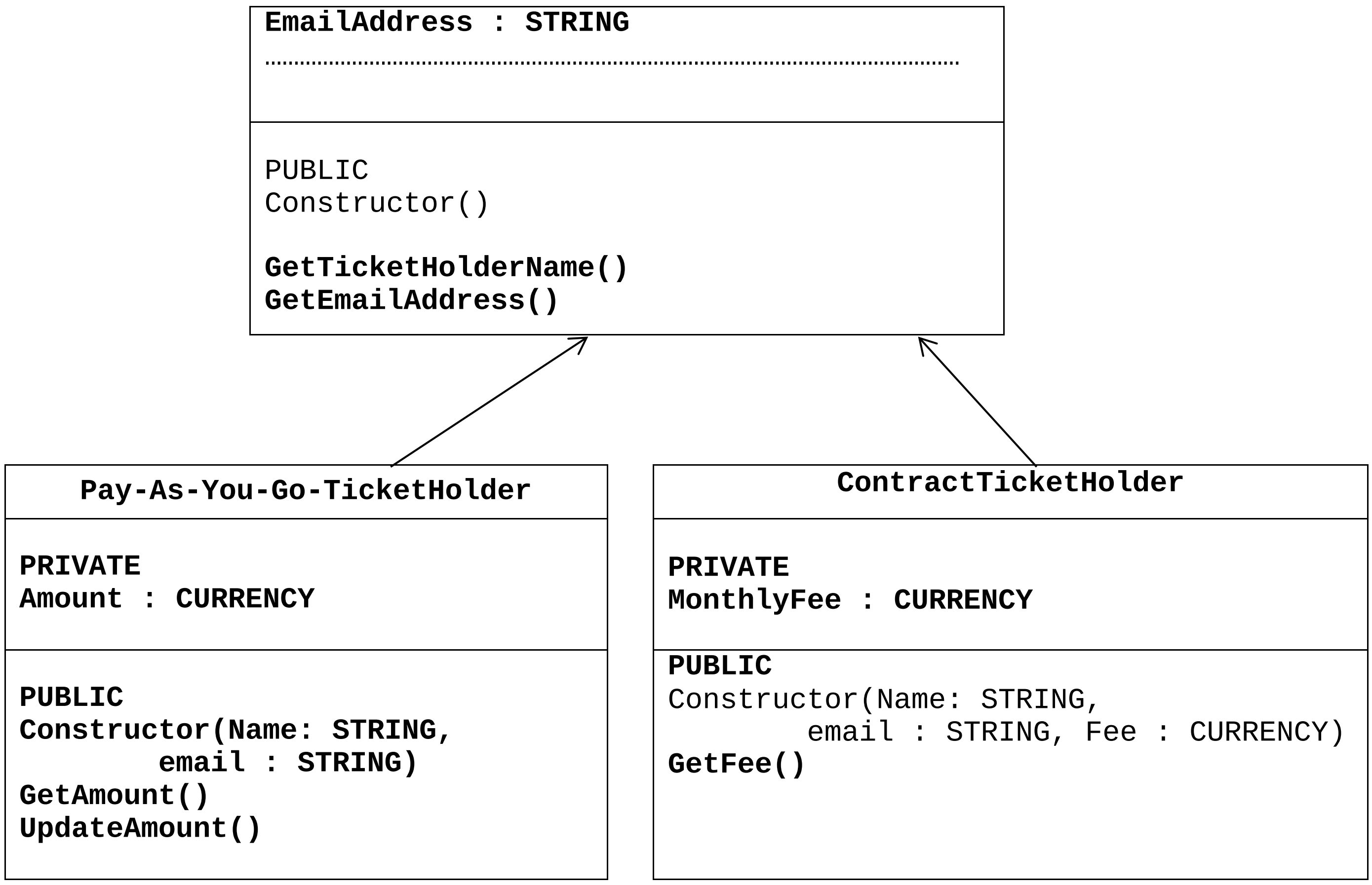

Figure 27.02

b i Attributes are declared as private so that they can only be changed through the class methods.

b ii Methods are declared as public so that they can be used to access the attributes.

c

Python	`NewCustomer = ContractTicketHolder("A. Smith", "xyz@abc.xx", 10)`

VB.NET	`Dim NewCustomer As New ContractTicketHolder()`
	`NewCustomer.Create("A. Smith", "xyz@abc.xx", 10)`

Pascal	`var NewCustomer : ContractTicketHolder;`
	
	`NewCustomer := ContractTicketHolder.Create('A. Smith','xyz@abc.xx',10);`

3 a Containment

b

Python	`class NodeClass :`

	```
	def __init__(self):          # initialiser method
	   self.__Data = ''
	   self.__Pointer = -1

	def SetData(self, d):
	   self.__Data = d

	def GetData(self):
	   return(self.__Data)

	def SetPointer(self, x):
	   self.__Data = x

	def GetPointer(self):
	   return(self.__Pointer)
	```

VB.NET	
	```
	Class NodeClass
	    Private Data As String
	    Private Pointer As Integer

	    Sub Create()
	        Data = ""
	        Pointer = -1
	    End Sub

	    Public Sub SetData(ByVal d As String)
	        Data = d
	    End Sub

	    Public Function GetData() As String
	        Return (Data)
	    End Function

	    Public Sub SetPointer(ByVal x As Integer)
	        Pointer = x
	    End Sub

	    Public Function GetPointer() As Integer
	        Return (Pointer)
	    End Function
	End Class
	```

Pascal	
	```
	type
	   NodeClass = class
	      private
	         Data : STRING;
	         Pointer : INTEGER;
	      public
	         constructor Create;
	         function GetData : STRING;
	         function GetPointer : INTEGER;
	         procedure SetData(d : STRING);
	         procedure SetPointer(x : INTEGER);

	      end;

	// implementation of methods
	```

```
constructor NodeClass.Create;
begin
   Data := '';
   Pointer := -1;
end;

function NodeClass.Getdata : STRING;
begin
   GetData := Data;
end;

function NodeClass.GetPointer : INTEGER;
begin
   GetPointer:= Pointer;
end;

procedure NodeClass.SetData(d : STRING);
begin
  Data := d;
end;

procedure NodeClass.SetPointer(x : INTEGER);
begin
  Pointer := x;
end;
```

c

Python

```
class QueueClass :

   def __init__(self):
      self.__Queue = [NodeClass() for i in range(51)]
      self.__Head = -1
      self.__Tail = -1
```

VB.NET

```
   Class QueueClass
      Private Queue(50) As Nodeclass
      Private Head As Integer
      Private Tail As Integer

      Sub Create()
         Head = -1
         Tail = -1
      End Sub
   End Class
```

Pascal

```
type
   QueueClass = class
      private
         Queue : array[0 .. 50] of NodeClass;
         Head : INTEGER;
         Tail : INTEGER;
      public
         constructor Create;
      end;


// implementation of methods

constructor QueueClass.Create;
begin
   Head := -1;
   Tail := -1;
end;
```

d

Python

```
  def JoinQueue(self, d):
      if Head == -1 :
          Head = 0
      self.__Tail += 1   # does not account for wrap
around
      i = self.__Tail
      self.__Queue[i].SetData(d)
```

VB.NET

```
        Sub JoinQueue(d)
            Dim i As Integer
            If Head = -1 Then
                Head = 0
            End If
            Tail += 1   ' does not account for wrap araound
            i = Tail
            Queue(i) = New NodeClass()
            Queue(i).Create()
            Queue(i).SetData(d)
        End Sub
```

Pascal

```
procedure QueuClass.JoinQueue(d : STRING);
var i : INTEGER;
begin
   Tail := Tail + 1; // does not account for wrap
around
   i := Tail;
   if Head = -1 then Head := 0; // first item to join
queue
   Queue[i] := NodeClass.Create();
   Queue[i].SetData(d);
end;
```