Chapter 26 Further Programming: Answers to coursebook questions and tasks

Syllabus sections covered: 4.3

Task 26.01 sequential file handling

Python

```
import pickle # this library is required to create
binary files
from datetime import date

class CarRecord :
   def __init__(self) :
      self.VehicleID = "dummy"
      self.Registration = ""
      self.DateOfRegistration = date(1990,1,1)
      self.EngineSize = 0
      self.PurchasePrice = 0.0



def SaveData(Car) :
   # file channel for car records
   CarFile = open('CarFile.DAT','wb')
   for i in range(100):  # loop for each array element
      # write a whole record to the binary file
      pickle.dump(Car[i], CarFile)
   CarFile.close()  # close file

def LoadData() :
   CarFile = open('CarFile.DAT','rb')  # open file for
binary read
   Car = []  # start with empty list
   EoF = False
   while not EoF :  # check for end of file
      try :
         Car.append(pickle.load(CarFile))  # append
record from file to end of list
      except :
         EoF = True
   CarFile.close()
   return Car

def OutputRecords(Car) :
   for i in range(100):  # loop for each array element
      print(Car[i].VehicleID)  # write one field

def main() :
   ThisCar = CarRecord()
   #Car =[ThisCar for i in range(100)] # only run this
1st time
   #SaveData(Car)  # only run this first time
   Car = LoadData() # from existing file
   OutputRecords(Car)
   # add more records
```

```
    i = int(input('Record Number? '))
    while i != 0 :
        Car[i].VehicleID = input('Vehicle ID: ')
        Car[i].Registration = input('Registration: ')
        Car[i].DateOfregistration = (input('Registration
Date: '));
        Car[i].EngineSize = int(input('Engine size: '))
        Car[i].PurchasePrice = float(input('Purchase
price: '))
        i = int(input('next Record Number? '))
    OutputRecords(Car)
    SaveData(Car)

main()
```

VB.NET

```
Option Explicit On
Imports System.IO

Module Module1
    Structure CarRecord
        Dim VehicleID As String
        Dim Registration As String
        Dim DateOfRegistration As Date
        Dim EngineSize As Integer
        Dim PurchasePrice As Decimal
    End Structure

    Dim CarFileWriter As BinaryWriter
    Dim CarFileReader As BinaryReader
    Dim CarFile As FileStream
    Dim Car(100) As CarRecord ' declare an array of CarRecord type

    Sub SaveData()
        'link file to filename
        CarFile = New FileStream("CarFile.DAT", FileMode.Create)
        ' create a new file and open it for writing
        CarFileWriter = New BinaryWriter(CarFile)
        For i = 1 To 100  ' loop for each array element
            CarFileWriter.Write(Car(i).VehicleID)  ' write a field to
the binary file
            CarFileWriter.Write(Car(i).Registration)
            CarFileWriter.Write(Car(i).DateOfRegistration)
            CarFileWriter.Write(Car(i).EngineSize)
            CarFileWriter.Write(Car(i).PurchasePrice)
        Next
        CarFileWriter.Close()  'close file channel
        CarFile.Close()
    End Sub

    Sub LoadData()
        Dim i As Integer
        CarFile = New FileStream("CarFile.DAT", FileMode.Open)  'link
to filename
        ' create a new file and open it for reading
        CarFileReader = New BinaryReader(CarFile)
        i = 1
        ' loop until end of binary file reached
        Do While CarFile.Position < CarFile.Length
            ' read fields from the binary file
            Car(i).VehicleID = CarFileReader.ReadString()
            Car(i).Registration = CarFileReader.ReadString()
```

```
            Car(i).DateOfRegistration = CarFileReader.ReadString()
            Car(i).EngineSize = CarFileReader.ReadInt32()
            Car(i).PurchasePrice = CarFileReader.ReadDecimal()
            i = i + 1
        Loop
        CarFileReader.Close()  'close file channel
        CarFile.Close()
    End Sub

    Sub OutputRecords()
        For i = 1 To 100
            Console.WriteLine(Car(i).VehicleID)
        Next
    End Sub

    Sub Main()
        Dim i As Integer
        LoadData()
        OutputRecords()
        Console.Write("Record number? ")
        i = Console.ReadLine()
        Do While i <> 0
            Console.Write("Vehicle ID: ")
            Car(i).VehicleID = Console.ReadLine()
            Console.Write("Registration: ")
            Car(i).Registration = Console.ReadLine()
            Console.Write("Registration Date: ")
            Car(i).DateOfRegistration = Console.ReadLine()
            Console.Write("Engine Size: ")
            Car(i).EngineSize = Console.ReadLine()
            Console.Write("Purchase Price: ")
            Car(i).PurchasePrice = Console.ReadLine()
            Console.Write("next Record Number: ") : i =
Console.ReadLine()
        Loop
        SaveData()
        Console.ReadLine()
    End Sub

End Module
```

Pascal

```
program Project2;

{$APPTYPE CONSOLE}

uses
  SysUtils;

type
   CarRecord = record
      VehicleID : string[20];
      Registration : string[10];
      DateOfRegistration : TDateTime;
      EngineSize : integer;
      PurchasePrice : currency;
   end;

var
   CarFile : file of CarRecord;  // file channel for
```

```
car records
   Car : array[1..100] of CarRecord; // array of
CarRecord type

 procedure SaveData;
 var i : integer;
 begin
    AssignFile(CarFile, 'CarFile.DAT');  // link to the
filename
    Rewrite(CarFile);  // create a new file and open
for writing

    for i := 1 to 100 do  // loop for each array
element
       Write(CarFile, Car[i]);  // write a whole record

    CloseFile(CarFile);  // close file channel
 end;

 procedure LoadData;
 var i : integer;
 begin
    AssignFile(CarFile, 'CarFile.DAT');  // link to the
filename
    Reset(CarFile);  // open file for reading
    i := 1;
    while not Eof(CarFile) do  // check for end of file
       begin
          Read(CarFile, Car[i]);  // read a record from
file
          i := i + 1;
       end;
    CloseFile(CarFile);  // close file channel
 end;

 procedure OutputRecords;
 var i : integer;
 begin
     for i := 1 to 100 do  // loop for each array
element
        WriteLn(Car[i].VehicleID);  // write one field
 end;

 var i : integer; datestring : string;
begin
   LoadData; // from existing file
   OutputRecords;
   // add more records
   write('Record Number? '); readln(i);
   while i <> 0 do
      begin
         write('Vehicle ID: ');
readLn(Car[i].VehicleID);
         write('Registration: ');
readLn(Car[i].Registration);
```

```
         write('Registration Date: ');
readLn(datestring);
         Car[i].DateOfregistration :=
StrToDate(datestring);
         write('Engine size: ');
readLn(Car[i].EngineSize);
         write('Purchase price: ');
readLn(Car[i].PurchasePrice);
         write('next Record Number? '); readln(i);
      end;
   SaveData;
   readln;
end.
```

Task 26.02 random-access file handling

Python

```
import pickle # this library is required to create
binary files
from datetime import date
RECORDSIZE = 50  # 20 + 10 +  8 + 4 + 8
class CarRecord :
   def __init__(self) :
      VehicleID = "dummy"
      VehicleID = VehicleID.ljust(20)
      self.VehicleID = VehicleID.encode('utf-8')
      Registration = " "
      Registration = Registration.ljust(10)
      self.Registration = Registration.encode('utf-
8')
      self.DateOfRegistration = date(1990,1,1)
      self.EngineSize = 0
      self.PurchasePrice = 0.0

def InitialiseFile() :
   CarFile = open('CarFile.DAT','wb') # file for car
records
   for i in range(100):  # loop for each array
element
      Address = i * RECORDSIZE + 1
      CarFile.seek(Address, 0)
      # write a whole record to the binary file
      pickle.dump(CarRecord(), CarFile)
   CarFile.close()  # close file

def InputNewRecordData() :
   ThisCar = CarRecord()
   VehicleID = input('Vehicle ID: ')
   VehicleID = VehicleID.ljust(20)
   ThisCar.VehicleID = VehicleID.encode('utf-8')
   Registration = input('Registration: ')
   Registration = Registration.ljust(10)
   ThisCar.Registration = Registration.encode('utf-
8')
   ThisCar.DateOfregistration = (input('Registration
```

```
Date: '));
   ThisCar.EngineSize = int(input('Engine size: '))
   ThisCar.PurchasePrice = float(input('Purchase
price: '))
   return ThisCar

def Hash(reg) :
   result = ord(reg[0]) * RECORDSIZE + 1
   print('Hashed to ',result)
   return result

def SaveToFile(ThisCar, CarFile) :
   Address = Hash(ThisCar.Registration.decode('utf-
8'))
   CarFile.seek(Address, 0)
   pickle.dump(ThisCar, CarFile)# write a whole
record to the binary file

def OpenFileForUpdate() :
   CarFile = open('CarFile.DAT','rb+') # open file
for update
   return CarFile

def FindRecord(reg, CarFile) :
   Address = Hash(reg)
   CarFile.seek(Address, 0)
   ThisCar = pickle.load(CarFile) # load record from
file
   return ThisCar

def OutputData(ThisCar) :
   print(ThisCar.VehicleID)  # write one field

def main() :
   InitialiseFile() # only run this procedure the
first time
   CarFile = OpenFileForUpdate()
   ThisCar = CarRecord()
   # add records
   Answer = input('add a record? (Y/N) ')
   while Answer != 'N' :
      ThisCar = CarRecord()
      ThisCar = InputNewRecordData()
      SaveToFile(ThisCar, CarFile)
      Answer = input('add a record? (Y/N) ')

   # find records
   Answer = input('find a record? (Y/N) ')
   while Answer != 'N' :
      Reg = input('Give vehicle registration: ')
      ThisCar = FindRecord(Reg, CarFile)
      OutputData(ThisCar)
      Answer = input('find a record? (Y/N) ')
   CarFile.close()
```

main()

VB.NET

```
Option Explicit On
Imports System.IO

Module Module1
    Structure CarRecord
        <VBFixedString(20)> Dim VehicleID As String
        <VBFixedString(10)> Dim Registration As String
        Dim DateOfRegistration As Date
        Dim EngineSize As Integer
        Dim PurchasePrice As Decimal
    End Structure

    Const RecordSize = 88 ' 40 + 20 + 8 + 4 + 16
    Dim ThisCar As CarRecord ' declare an array of CarRecord type

    Sub InitialiseFile()
        ' set up a dummy record and store in each record position
in file
        ThisCar.VehicleID = ""
        ThisCar.Registration = ""
        ThisCar.DateOfRegistration = #1/1/1900#
        ThisCar.EngineSize = 0
        ThisCar.PurchasePrice = 0.0
        For i = 1 To 100 ' loop for each array element
            FilePut(1, ThisCar, i)
        Next
    End Sub

    Sub InputNewRecordData()
        Console.Write("Vehicle ID: ")
        ThisCar.VehicleID = Console.ReadLine()
        Console.Write("Registration: ")
        ThisCar.Registration = Console.ReadLine()
        Console.Write("Registration Date: ")
        ThisCar.DateOfRegistration = Console.ReadLine()
        Console.Write("Engine Size: ")
        ThisCar.EngineSize = Console.ReadLine()
        Console.Write("Purchase Price: ")
        ThisCar.PurchasePrice = Console.ReadLine()
    End Sub

    Function Hash(r) As Integer
        Dim Position As Integer
        Position = Asc(r(1))
        Return Position
    End Function

    Sub SaveToFile()
        Dim Position As Integer
        Position = Hash(ThisCar.Registration)
        FilePut(1, ThisCar, Position)
    End Sub

    Sub OpenFileForUpdate()
        'link the file to the filename
        FileOpen(1, "CarFile.DAT", OpenMode.Random, , , RecordSize)
    End Sub

    Sub FindRecord(reg)
        Dim Position As Integer
        Position = Hash(reg)
```

```
        FileGet(1, ThisCar, Position)
    End Sub

    Sub OutputData()
        Console.WriteLine(ThisCar.VehicleID) ' write one field
    End Sub

    Sub Main()
        Dim Answer, Reg As String
        OpenFileForUpdate()
        InitialiseFile() ‘ only use this first time round
        'add records
        Console.Write("add a record? (Y/N) ")
        Answer = Console.ReadLine()
        Do While Answer <> "N"
            InputNewRecordData()
            SaveToFile()
            Console.Write("add a record: (Y/N) ")
            Answer = Console.ReadLine()
        Loop
        ' find records
        Console.Write("find a record? (Y/N) ")
        Answer = Console.ReadLine()
        Do While Answer <> "N"
            Console.Write("Give vehicle registration: ")
            Reg = Console.ReadLine()
            FindRecord(Reg)
            OutputData()
            Console.Write("find a record: (Y/N) ")
            Answer = Console.ReadLine()
        Loop
        FileClose(1)
        Console.ReadLine()
    End Sub

End Module
```

Pascal

```
program Project2;

{$APPTYPE CONSOLE}

uses
  SysUtils;

type
   CarRecord = record
      VehicleID : string[20];
      Registration : string[10];
      DateOfRegistration : TDateTime;
      EngineSize : integer;
      PurchasePrice : currency;
   end;

var
   // declare a file channel to take car records
   CarFile : file of CarRecord;
   ThisCar : CarRecord; // declare a variable of
CarRecord type
```

```
 procedure InitialiseFile;
 var i : integer;
 begin
    // link the file channel to the filename
    AssignFile(CarFile, 'CarFile2.DAT');
    Rewrite(CarFile);  // create a new file and open
it for writing
    // create a dummy record
    ThisCar.VehicleID := '';
    ThisCar.Registration := '';
    ThisCar.DateOfRegistration :=
StrToDate('01/01/1900');
    ThisCar.EngineSize := 0;
    ThisCar.PurchasePrice := 0.0;

    for i := 1 to 100 do  // loop for each array
element
       // write dummy record to the binary file
       Write(CarFile, ThisCar);
    CloseFile(CarFile);  // close file channel
 end;

 procedure InputNewRecordData;
 var datestring : string;
      begin
         write('Vehicle ID: ');
readLn(ThisCar.VehicleID);
         write('Registration: ');
readLn(ThisCar.Registration);
         write('Registration Date: ');
readLn(datestring);
         ThisCar.DateOfregistration :=
StrToDate(datestring);
         write('Engine size: ');
readLn(ThisCar.EngineSize);
         write('Purchase price: ');
readLn(ThisCar.PurchasePrice);
      end;

 function Hash(r : string) : integer;
 begin
    result := ord(r[1]); // a very simple hashing
function
 end;

 procedure SavetoFile;
 var Address : integer;
 begin
    Address := Hash(ThisCar.Registration);
    Seek(CarFile, Address);
    Write(CarFile, ThisCar);  // write record to file
 end;

 procedure OpenFileForUpdate;
 begin
```

```
    // link the file channel to the filename
    AssignFile(CarFile, 'CarFile2.DAT');
    Reset(CarFile);  // open file for update
 end;

 procedure FindRecord(reg : string);
 var Address : integer;
 begin
    Address := Hash(reg);
    Seek(CarFile, Address);
    Read(CarFile, ThisCar);  // read record from file
 end;

 procedure OutputData;
 begin
     WriteLn(ThisCar.VehicleID);  // write one field
 end;

 // ********** main program starts here
******************

var answer, reg : string;
begin
   InitialiseFile;
   OpenFileForUpdate;

   // add records
   write('add a record? (Y/N) '); readln(answer);
   while answer <> 'N' do
      begin
         InputNewRecordData;
         SaveToFile;
         write('add a record? (Y/N) ');
readln(answer);
      end;

   // find records
   write('find a record? (Y/N) '); readln(answer);
   while answer <> 'N' do
      begin
         write('Give vehicle registration: ');
readln(reg);
         FindRecord(reg);
         OutputData;
         write('find a record? (Y/N) ');
readln(answer);
      end;

   CloseFile(CarFile);
   readln;
end.
```

Task 26.03 exception handling

Python

```
    EoF = False
    while not EoF :  # check for end of file
       try :
          Car.append(pickle.load(CarFile))
       except :
          EoF = True
```

VB.NET

```
    Sub SaveToFile()
        Dim Position As Integer
        Position = Hash(ThisCar.Registration)
        Try
            FilePut(1, ThisCar, Position)
        Catch ex As Exception
            Console.WriteLine("File not open")
        End Try
    End Sub
```

Pascal

```
procedure SavetoFile;
 var Address : integer;
 begin
    Address := Hash(ThisCar.Registration);
    try
       Seek(CarFile, Address);
       Write(CarFile, ThisCar);  // write record to
file
    except
       Writeln('File is not open');
    end;
 end;
```

Exam style Questions

1 a i

Python

```
class CustomerRecord :
   def __init__(self) :
      self.CustomerID = 0
      self.CustomerName = ''
      self.TelNumber = ''
      self.TotalOrders = 0
```

VB.NET

```
    Structure CustomerRecord
        Dim CustomerID As Integer
        Dim CustomerName As String
        Dim TelNumber As String
        Dim TotalOrders As Decimal
    End Structure
```

Pascal

```
type
   CustomerRecord = record
      CustomerID : integer;
      CustomerName : string[30];
      TelNumber : string[14];
      TotalOrders : currency;
```

	end;

ii

Python	`CustomerData = [CustomerRecord() for I in range(1000)]`
VB.NET	`Dim CustomerData(999) As CustomerRecord ' declare an array of CustomerRecord type`
Pascal	`var CustomerData[0..999] of CustomerRecord;`

b i

Python	`def Hash(ID) :`
	`  Address = ID % 1000`
	`  return(Address)`

VB.NET	`Function Hash(ID) As Integer`
	`  Dim Address As Integer`
	`  Address = ID Mod 1000`
	`  Return Address`
	`End Function`

Pascal	`Function Hash(ID) : Integer;`
	`  var Address : Integer;`
	`  Address := ID Mod 1000;`
	`  Hash := Address;`
	`End;`

ii

Python	`def AddRecord(CustomerData, Customer)`
	`  Address = Hash(Customer.CustomerID)`
	`  while CustomerData(Address).CustomerID != 0 :`
	`  Address += 1`
	`  if Address = 1000 :`
	`  Address = 0`
	`  CustomerData[Address] = Customer`

VB.NET	`Sub AddRecord(Customer As CustomerRecord)`
	`  Dim Address As Integer`
	`  Address = Hash(Customer.CustomerID)`
	`  Do While CustomerData(Address).CustomerID <> 0`
	`  Address += 1`
	`  If Address = 1000 Then Address = 0`
	`  Loop`
	`  CustomerData(Address) = Customer`
	`End Sub`

Pascal	`procedure AddRecord(Customer : CustomerRecord)`
	`var Address : integer;`
	`begin`
	`  while CustomerData[Address].CustomerID <> 0 do`

```
        begin
           Address := Address + 1;
           if Address = 1000 then Address = 0
        end;
    CustomerData[Address] = Customer
end;
```

iii

Python

```
def FindRecord(CustomerData, ID)
    Address = Hash(ID)
    while CustomerData[Address].CustomerID != ID :
        Address += 1
        if Address = 1000 :
            Address = 0
    return (Address)
```

VB.NET

```
    Function FindRecord(ID) As Integer
        Dim Address As Integer
        Address = Hash(ID)
        Do While CustomerData(Address).CustomerID <> ID
            Address += 1
            If Address = 1000 Then Address = 0
        Loop
        Return Address
    End Function
```

Pascal

```
Function FindRecord(ID : Integer) : Integer;
var Address : Integer;
begin
    Address := Hash(ID);
    while CustomerData[Address].CustomerID <> 0 do
        begin
           Address := Address + 1;
           if Address = 1000 then Address = 0;
        end;
    FindRecord := Address;
End;
```

c

Python

```
import pickle
def SaveData(CustomerData) :
    CustomerFile = open('CustomerData.Dat','wb')
    for i in range(1000) :
        pickle.dump(CustomerData[i], CustomerFile)
    CustomerFile.close()
```

VB.NET

```
    Sub SaveData()
        'link file to filename
        CustomerFile = New IO.FileStream("CustomerData.DAT",
IO.FileMode.Create)
        ' create a new file and open it for writing
        CustomerFileWriter = New IO.BinaryWriter(CustomerFile)
        For i = 0 To 999  ' loop for each array element
```

```
            ' write a field to the binary file
            CustomerFileWriter.Write(CustomerData(i).CustomerID)
            CustomerFileWriter.Write(CustomerData(i).CustomerName)
            CustomerFileWriter.Write(CustomerData(i).TelNumber)
            CustomerFileWriter.Write(CustomerData(i).TotalOrders)
        Next
        CustomerFileWriter.Close()  'close file channel
        CustomerFile.Close()
    End Sub
```

Pascal

```
procedure SaveData;
var CustomerFile : file of CustomerRecord;
begin
   assignFile(CustomerFile, 'CustomerData.Dat');
   rewrite(CustomerFile);
   for i := 0 to 999 do           // loop for each
array element
      write(CustomerFile, CustomerData[i]);
   closeFile(CustomerFile);
End;
```

d Need to set up fixed length dummy records and save them to a random file. AddRecord needs to update the correct record in the random file. FindRecord needs to read the random file. Don't need the SaveData procedure at the end of program execution

2

Python

```
def OpenFile() :
   FileName = input("Which file do you want to use?
")
   try:
      Channel = open(FileName, 'rb+')
   except :
      print("File does not exist")
```

VB.NET

```
    Sub OpenFile()
        Dim FileName As String
        Console.Write("Which file do you want to use? ")
        FileName = Console.ReadLine()
        Try
            FileOpen(1, FileName, OpenMode.Random)
        Catch ex As Exception
            Console.WriteLine("This file does not exist")
        End Try
    End Sub
```

Pascal

```
procedure OpenFile;
var FileName : string;
    Channel : file of recordType;
begin
  Write('Which file do you want to use? ');
  ReadLn(FileName);
  try
     AssignFile(Channel, FileName);
     Reset(Channel);
  except
```

```
      WriteLn('This file does not exist');
  end;
end;
```