Chapter 20 Student Book Answers

20.1 What you should already know

1 Absolute/Immediate – uses the operand

Direct – uses the contents of the memory location specified by the operand

Indirect – uses the contents of the contents of the memory location specified by the operand

Indexed – uses the value found at the memory location specified by adding operand to the contents of the index register

2
```
LDM #7
ADD #5
```

3 a) A function has a type and always returns a value.

b) Parameters are values/references included in a call to a procedure or function. Parameters can be passed by reference (value of variable used can be changed by the procedure/function) or by value (no change can be made).

c) A procedure definition is made once but the code specified can be used many times in a program, every time the procedure is called.

4 See answer to Activity 11G

Activity 20A

Address	Opcode	Operand	ACC	IX
500	LDM	#230	#230	
501	LDD	230	#231	
502	LDI	230	#5	
503	LDR	#1		#1
504	LDX	230	#5,#7,#9,#11,#0	
505	CMP	#0		
506	JPE	509		
507	INC	IX		#2,#3,#4,#5
508	JMP	504		
509	JMP	509		

Activity 20B

Pseudocode – without procedures and functions

```
DECLARE base, height, radius : INTEGER
DECLARE area : REAL
DECLARE CONSTANT Pi = 3.142
OUTPUT "Please enter base, height and radius"
INPUT base, height, radius
area = base * base
OUTPUT "Area of square is ", area
area = base * height
OUTPUT "Area of rectangle is ", area
area = base * base / 2
OUTPUT "Area of triangle is ", area
area = base * height
OUTPUT "Area of parallelogram is ", area
area = Pi * radius * radius
OUTPUT "Area of circle is ", area
```

Pseudocode – with procedures and functions

```
DECLARE base, height, radius : INTEGER
DECLARE area : REAL

FUNCTION square (side : INTEGER) RETURNS REAL
    RETURN side * side
ENDFUNCTION

FUNCTION rectangle (side : INTEGER, otherSide) RETURNS REAL
    RETURN side * otherSide
ENDFUNCTION

FUNCTION triangle (triBase : INTEGER, triHeight) RETURNS REAL
    RETURN triBase * triHeight / 2
ENDFUNCTION

FUNCTION parallelogram (parBase : INTEGER, parHeight) RETURNS
REAL
    RETURN parBase * parBase
ENDFUNCTION

FUNCTION circle(circRad : INTEGER) RETURNS REAL
    DECLARE CONSTANT Pi = 3.142
    RETURN circRad * circRad * Pi
ENDFUNCTION

PROCEDURE printArea (shapeName : String : areaToPrint : REAL)
    OUTPUT "Area of ", shapeName, " is ", areaToPrint
ENDPROCEDURE

OUTPUT "Please enter base, height and radius"
INPUT base, height, radius
area = square(base)
printArea ("square", area)

area = rectangle(base, height)
printArea ("rectangle", area)

area = triangle(base, height)
printArea ("triangle", area)
```

```
area = parallelogram(base, height)
printArea ("parallelogram", area)

area = circle(radius)
printArea ("circle", area)
```

Python – without procedures and functions

```
#areas iterative program
Pi = 3.142

base = int(input ("Please enter base "))
height = int(input ("Please enter height "))
radius = int(input ("Please enter radius "))

area = base * base
print ("Area of square is ", area)

area = base * height
print ("Area of rectangle is ", area)

area = base * height / 2
print ("Area of triangle is ", area)

area = base * height
print ("Area of parallelogram is ", area)

area = radius * radius * Pi
print ("Area of rectangle is ", area)
```

Python – with procedures and functions

```
#areas using  prodedures and functions

def square(side):
    return side * side

def rectangle(side, otherSide):
    return side * otherSide

def triangle(triBase, triHeight):
    return triBase * triHeight / 2

def parallelogram(parBase, parHeight):
    return parBase * parHeight

def circle(circRad):
    Pi = 3.142
    return circRad * circRad * Pi

def printArea (shapeName, areaToPrint):
    print ("Area of ", shapeName, " is ", areaToPrint)

base = int(input ("Please enter base "))
height = int(input ("Please enter height "))
radius = int(input ("Please enter radius "))

area = square(base)
printArea ("square", area)

area = rectangle(base, height)
printArea ("rectangle", area)

area = triangle(base, height)
printArea ("triangle", area)

area = parallelogram(base, height)
printArea ("parallelogram", area)

area = circle(radius)
printArea ("circle", area)
```

VB – without procedures and functions

```
' areas imperative program
Module Module1
    Public Sub Main()
        Dim base, height, radius As Integer
        Dim area As Decimal
        Const Pi As Decimal = 3.14

        Console.Write("Please enter base ")
        base = Integer.Parse(Console.ReadLine())
        Console.Write("Please enter height ")
        height = Integer.Parse(Console.ReadLine())
        Console.Write("Please enter radius ")
        radius = Integer.Parse(Console.ReadLine())
        area = base * base
        Console.WriteLine("Area of a square is " & area)
        area = base * height
        Console.WriteLine("Area of a rectangle is " & area)
        area = base * height / 2
        Console.WriteLine("Area of a triangle is " & area)
        area = base * height
        Console.WriteLine("Area of a parallelogram is " & area)
        area = radius * radius * Pi
        Console.WriteLine("Area of circle is " & area)

        Console.ReadKey()

    End Sub

End Module
```

VB – with procedures and functions

```
'areas program with functions
Module Module1
    Public Sub Main()
        Dim base, height, radius As Integer
        Dim area As Decimal
        Const Pi As Decimal = 3.14

        Console.Write("Please enter base ")
        base = Integer.Parse(Console.ReadLine())
        Console.Write("Please enter height ")
        height = Integer.Parse(Console.ReadLine())
        Console.Write("Please enter radius ")
        radius = Integer.Parse(Console.ReadLine())
        area = square(base)
        printArea("square", area)
        area = rectangle(base, height)
        printArea("rectangle", area)
        area = triangle(base, height)
        printArea("trangle", area)
        area = parallelogram(base, height)
        printArea("parallelogram", area)
        area = circle(radius)
        printArea("circle", area)

        Console.ReadKey()

    End Sub

    Function square(side As Integer) As Decimal
        Return side * side
    End Function

    Function rectangle(side As Integer, otherSide As Integer) As Decimal
        Return side * otherSide
    End Function

    Function triangle(triBase As Integer, triHeight As Integer) As
Decimal
        Return triBase * triHeight / 2
    End Function

    Function parallelogram(parBase As Integer, parHeight As Integer) As
Decimal
        Return parBase * parHeight
    End Function

    Function circle(radius As Integer) As Decimal
        Const Pi As Decimal = 3.14
        Return radius * radius * Pi
    End Function

    Sub printArea(ByVal shapeName As String, ByVal areaToPrint As
Decimal)
        Console.WriteLine("Area of a " & shapeName & " is " &
areaToPrint)
    End Sub
End Module
```

Java – without procedures and functions

```
import java.util.Scanner;
class ACTIVITY20B
{
   public static void main(String args[])
    {
       Scanner myObj = new Scanner(System.in);
       final double PI = 3.142;
       double base, height, radius, area;
       System.out.println("Please enter base ");
       base = myObj.nextDouble();
       System.out.println("Please enter height ");
       height = myObj.nextDouble();
       System.out.println("Please enter radius ");
      radius = myObj.nextDouble();

      area = base * base;
      System.out.println("Area of a square is " + area);
      area = base * height;
      System.out.println("Area of a rectangle is " + area);
      area = base * height / 2;
      System.out.println("Area of a triangle is " + area);
      area = base * height;
      System.out.println("Area of a parallelogram is " + area);
      area = PI * radius * radius;
      System.out.println("Area of a circle is " + area);

    }
}
```

Java – with procedures and functions

```
import java.util.Scanner;
class ACTIVITY20Bfunction
{
   static void printArea (String shapeName, double areaToPrint)
   {
     System.out.println("Area of a " + shapeName + " is " +
areaToPrint);
    }
    static double square(double side)
    {
        return side * side;
    }

    static double rectangle(double side, double otherSide)
    {
        return side * otherSide;
    }
    static double triangle(double triBase, double triHeight)
    {
        return triBase * triHeight / 2;
    }

    static double parallelogram(double parBase, double parHeight)
    {
        return parBase * parHeight;
    }

    static double circle(double radius)
    {
        final double PI = 3.142;
        return radius * radius * PI;
    }
    public static void main(String args[])
    {
       Scanner myObj = new Scanner(System.in);

       double base, height, radius, area;
       System.out.println("Please enter base ");
       base = myObj.nextDouble();
       System.out.println("Please enter height ");
       height = myObj.nextDouble();
       System.out.println("Please enter radius ");
      radius = myObj.nextDouble();

      area = square(base);
      printArea("square", area);
      area = rectangle(base, height);
      printArea("rectangle", area);
      area = triangle(base, height);
      printArea("triangle", area);
      area = base * height;
      printArea("parallelogram", area);
      area = circle(radius);
      printArea("circle", area);

    }
}
```

Activity 20C

Python

```
#ACTIVITY 20C classes and objects
class student:
  def __init__(self, name, dateOfBirth, examMark):
    self.name = name
    self.dateOfBirth = dateOfBirth
    self.examMark = examMark;
  def displayExamMark(self):
    print("Student Name " + self.name)
    print("Exam Mark " , self.examMark)

myStudent = student("Mary Wu", 12/10/2012, 67)

myStudent.displayExamMark()
```

VB

```
'ACTIVITY 20C classes and objects
Module Module1
    Public Sub Main()
        Dim myStudent As New student("Mary Wu", #10/12/2012#, 67)
        myStudent.displayExamMark()
    End Sub

    Class student
        Dim name As String
        Dim dateOfBirth As Date
        Dim examMark As Integer
        Public Sub New(ByVal n As String, ByVal d As Date, ByVal e As
Integer)
            name = n
            dateOfBirth = d
            examMark = e
        End Sub

        Public Sub displayExamMark()
            Console.WriteLine("Student Name " & name)
            Console.WriteLine("Exam Mark " & examMark)
            Console.ReadKey()
        End Sub
    End Class

End Module
```

Java

```
//ACTIVITY 20C classes and objects
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
class student{
    private String name;
    private LocalDate dateOfBirth;
    private int examMark;

    student(String n, LocalDate d, int e){
        name = n;
        dateOfBirth = d;
        examMark = e;
        }
    public void displayExamMark (){
        System.out.println("Student Name " + name);
        System.out.println("ExamMark " + examMark);
        }
}
public class ATIVITY20C{
  public static void main(String[] args) {
        student myStudent = new student("Mary Wu", LocalDate.parse("2012-12-
10"), 67);
        myStudent.displayExamMark();

  }
}
```

Activity 20D

Python

```
#ACTIVITY 20D classes, objects and inheritance
class student:
  def __init__(self, name, dateOfBirth, examMark):
    self.name = name
    self.dateOfBirth = dateOfBirth
    self.examMark = examMark;
  def displayExamMark(self):
    print("Student Name " + self.name)
    print("Exam Mark " , self.examMark)

class partTimeStudent(student):
   def __init__(self, name, dateOfBirth, examMark):
        student.__init__(self, name, dateOfBirth, examMark)
        self.__fullTimeStudent = False

class fullTimeStudent(student):
   def __init__(self, name, dateOfBirth, examMark):
        student.__init__(self, name, dateOfBirth, examMark)
        self.__fullTimeStudent = True

myFullTimeStudent = fullTimeStudent("Mary Wu", 12/10/2012, 67)
myFullTimeStudent.displayExamMark()
myPartTimeStudent = partTimeStudent("Janet Yo", 23/5/2012, 95)
myPartTimeStudent.displayExamMark()
```

VB

```
'ACTIVITY 20D classes, objects and inheritance
Module Module1
    Public Sub Main()
        Dim myFullTimeStudent As New fullTimeStudent("Mary Wu", #10/12/2012#, 67)
        myFullTimeStudent.displayExamMark()
        Dim myPartTimeStudent As New fullTimeStudent("Janet Yo", #05/23/2012#, 95)
        myPartTimeStudent.displayExamMark()
        Console.ReadKey()
    End Sub

    Class student
        Dim name As String
        Dim dateOfBirth As Date
        Dim examMark As Integer
        Public Sub New(ByVal n As String, ByVal d As Date, ByVal e As Integer)
            name = n
            dateOfBirth = d
            examMark = e
        End Sub

        Public Sub displayExamMark()
            Console.WriteLine("Student Name " & name)
            Console.WriteLine("Exam Mark " & examMark)
        End Sub
    End Class

    Class partTimeStudent : Inherits student
        Private ReadOnly fullTimeStudent = False

        Public Sub New(ByVal n As String, ByVal d As Date, ByVal e As Integer)
            MyBase.New(n, d, e)
        End Sub

    End Class

    Class fullTimeStudent : Inherits student
        Private ReadOnly fullTimeStudent = True

        Public Sub New(ByVal n As String, ByVal d As Date, ByVal e As Integer)
            MyBase.New(n, d, e)
        End Sub

    End Class

End Module
```

Java

```
//ACTIVITY 20D classes, objects and inheritance
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
class student{
    private String name;
    private LocalDate dateOfBirth;
    private int examMark;

    student(String n, LocalDate d, int e){
        name = n;
        dateOfBirth = d;
        examMark = e;
        }
    public void displayExamMark (){
        System.out.println("Student Name " + name);
        System.out.println("ExamMark " + examMark);
        }
}

class partTimeStudent extends student {
    private boolean fullTimeStudent = false;
        partTimeStudent (String n, LocalDate d, int e){
        super (n, d, e);
        }
}

class fullTimeStudent extends student {
    private boolean fullTimeStudent = true;
        fullTimeStudent (String n, LocalDate d, int e){
        super (n, d, e);
        }
    }

public class ATIVITY20D{
  public static void main(String[] args) {
        student myFullTimeStudent = new fullTimeStudent("Mary Wu", LocalDate.parse("2012-12-10"), 67);
        myFullTimeStudent.displayExamMark();
        student myPartTimeStudent = new partTimeStudent("Janet Yo", LocalDate.parse("2012-07-05"), 95);
        myPartTimeStudent.displayExamMark();

  }
}
```

Activity 20E

Python

```
# ACTIVITY 20E polymorphism
class shape:
  def __init__(self):
    self.__areaValue = 0
    self.__perimeterValue = 0

  def area(self):
    print("Area ", self.__areaValue)

class square(shape):
    def __init__(self, side):
        shape.__init__(self)
        self.__side = side

    def area (self):
      self.__areaValue = self.__side * self.__side
      print("Area ", self.__areaValue)

class rectangle(shape):
    def __init__(self, length, breadth):
      shape.__init__(self)
      self.__length = length
      self.__breadth = breadth

    def area (self):
      self.__areaValue = self.__length * self.__breadth
      print("Area ", self.__areaValue)

class circle(shape):
    def __init__(self, radius):
      shape.__init__(self)
      self.__radius = radius

    def area (self):
      self.__areaValue = self.__radius * self.__radius * 3.142
      print("Area ", self.__areaValue)

mySquare = square(10)
mySquare.area()

myCircle = circle(20)
myCircle.area()

myRectangle = rectangle (10,17)
myRectangle.area()
```

VB

```
'VB ACTIVITY 20E polymorphism
Module Module1
    Public Sub Main()
        Dim myCircle As New circle(20)
        myCircle.Area()
        Dim myRectangle As New rectangle(10, 17)
        myRectangle.Area()
        Dim mySquare As New square(10)
        mySquare.Area()

        Console.ReadKey()
    End Sub

    Class shape
        Protected areaValue As Decimal
        Protected perimeterValue As Decimal

        Overridable Sub area()
            Console.WriteLine("Area " & areaValue)
        End Sub
    End Class

    Class square : Inherits shape
        Private side As Decimal

        Public Sub New(ByVal s As Decimal)
            side = s
        End Sub
        Overrides Sub Area()
            areaValue = side * side
            Console.WriteLine("Area " & areaValue)
        End Sub
    End Class

    Class rectangle : Inherits shape
            Private length As Decimal
            Private breadth As Decimal

            Public Sub New(ByVal l As Decimal, ByVal b As Decimal)
                length = l
                breadth = b
            End Sub

            Overrides Sub Area()
            areaValue = length * breadth
            Console.WriteLine("Area " & areaValue)
        End Sub

    End Class

    Class circle : Inherits shape
        Private radius As Decimal

        Public Sub New(ByVal r As Decimal)
            radius = r
        End Sub

        Overrides Sub Area()
            areaValue = radius * radius * 3.142
            Console.WriteLine("Area " & areaValue)
        End Sub
    End Class

End Module
```

Java

```
// ACTIVITY20E polymorphism

class shape {
    protected double areaValue;

    public void area (){
        System.out.println("Area " + areaValue);
        }
}

class square extends shape {
    private double side;

    square (double s){
        side = s;
    }
    public void area (){
        areaValue = side * side;
        System.out.println("Area " + areaValue);
        }
}

class rectangle extends shape {
    private double length;
    private double breadth;

    rectangle(double l, double b){
        length = l;
        breadth = b;
    }
    public void area (){
        areaValue = length * breadth;
        System.out.println("Area " + areaValue);
        }
}
class circle extends shape {
    private double radius;

    circle (double r){
        radius = r;
    }
    public void area (){
        areaValue = radius * radius * 3.142;
        System.out.println("Area " + areaValue);
        }
}
public class ACTIVITY20E{
  public static void main(String[] args) {
    square mySquare = new square(20);
    mySquare.area();
    circle myCircle = new circle(20);
    myCircle.area();
    rectangle myRectagle = new rectangle(10, 17);
    myRectagle.area();

  }
}
```

Activity 20F

Python

```python
#ACTIVITY 20F overloading
class greeting:

    def hello(self, firstName = None,lastName = None):

        if firstName is not None and lastName is not None:
            print ("Hello " + firstName + " " + lastName)
        elif firstName is not None:
            print ("Hello " + firstName)
        else:
            print ("Hello")

myGreeting = greeting()

myGreeting.hello()

myGreeting.hello("Christopher")

myGreeting.hello("Christopher", "Robin")
```

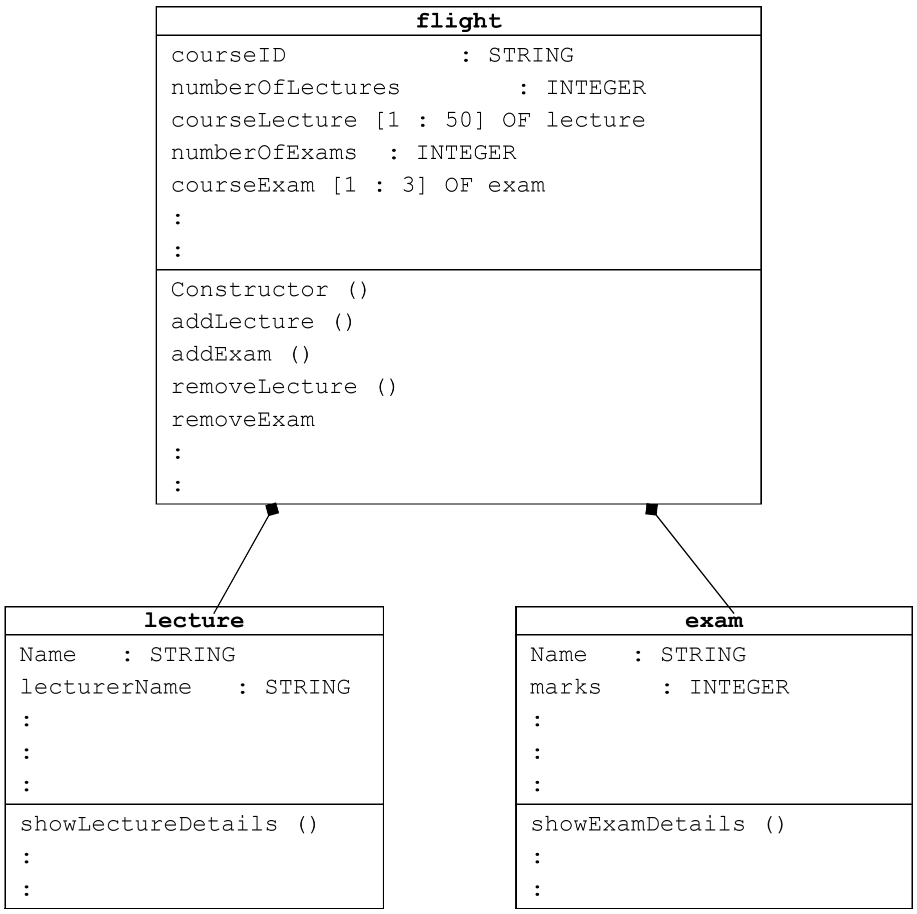
VB

```vb
'ACTIVITY 20F overloading
Module Module1
    Public Sub Main()
        Dim myGreeting As New greeting
        myGreeting.hello()
        myGreeting.hello("Christopher")
        myGreeting.hello("Christopher", "Robin")
        Console.ReadKey()
    End Sub

    Class greeting

        Public Overloads Sub hello()
            Console.WriteLine("Hello")
        End Sub

        Public Overloads Sub hello(ByVal name As String)
            Console.WriteLine("Hello " & name)
        End Sub

        Public Overloads Sub hello(ByVal firstName As String, ByVal
lastName As String)
            Console.WriteLine("Hello " & firstName & " " & lastName)
        End Sub
    End Class

End Module
```

Java

```
//ACTIVITY 20F overloading

class greeting{
    public void hello(){
         System.out.println("Hello");
    }

    public void hello(String name){
         System.out.println("Hello " + name);
    }

    public void hello(String firstName, String secondName){
         System.out.println("Hello " + firstName + " " + secondName);
    }
}

class ACTIVITY20F{
   public static void main(String[] args){
         greeting myGreeting = new greeting();
         myGreeting.hello();
         myGreeting.hello("Christopher");
         myGreeting.hello("Christopher","Robin");
   }
}
```

Activity 20G



Containment diagram for a university course

Activity 20H

Sample Answer in Python

```
#ACTIVITY20H
class Node:

    def __init__(self, item):

        self.leftPointer = None
        self.rightPointer = None
        self.item = item

    def insert(self, item):
# Compare the new item with the parent node
        if self.item:
            if item < self.item:
                if self.leftPointer is None:
                    self.leftPointer = Node(item)
                else:
                    self.leftPointer.insert(item)
            elif item > self.item:
                if self.rightPointer is None:
                    self.rightPointer = Node(item)
                else:
                    self.rightPointer.insert(item)
        else:
            self.item = item

    def search(self, item):
        while self.item != item:
            if item < self.item:
                self.item = self.leftPointer
            else:
                self.item = self.rightPointer
            if self.item is None:
                return None
        return self.item

# Print the tree in order
    def PrintInOrder(self):
        if self.leftPointer:
            self.leftPointer.PrintInOrder()
        print( self.item)
        if self.rightPointer:
            self.rightPointer.PrintInOrder()

# Preorder traversal
    def PreorderTraversal(self, tree):
        result = []
        if tree:
            result.append(tree.item)
            result = result +
self.PreorderTraversal(tree.leftPointer)
            result = result +
self.PreorderTraversal(tree.rightPointer)
        return result

# set up the root node
tree = Node(27)

# Use the insert method to add nodes to
tree.insert(19)
tree.insert(36)
tree.insert(42)
tree.insert(16)

#print(tree.search(16))

tree.PrintInOrder()
print(tree.PreorderTraversal(tree)
```

Activity 20I

```
language(england,Language).

language(Country,japanese).
```

Activity 20J

```
savingsRate(laila, X).

bankAccount(X,savings,Y).


bankAccount(robert,savings,300.00).


interest(sevenPercent,savings,2000.00).
```

Activity 20K

1 Absolute/Immediate – uses the operand, e.g. `LDM #20` stores the denary value 20 in the accumulator.

Direct – uses the contents of the memory location specified by the operand, e.g. `LDD 20` stores the value stored in the location with address 20 in the accumulator.

Indirect - uses the contents of the contents of the memory location specified by the operand, e.g. `LDI 20` stores the value stored in the location with the address stored at location 20 in the accumulator.

Indexed – uses the value found at the memory location specified by adding operand to the contents of the index register, e.g. `LDX 20` stores the value stored in the location with address 20 plus the contents of the index register in the accumulator.

2 Use of procedures and functions for procedural programming. For example, each procedure or function is developed separately.

```
def square(side):
    return side * side

def rectangle(side, otherSide):
    return side * otherSide
:
:
def printArea (shapeName, areaToPrint):
    print ("Area of ", shapeName, " is ", areaToPrint)
```

Use of polymorphism in object-oriented programming where classes can be derived from other classes. For example, the use of the base class shape and the derived classed for specific shapes.

```
class shape:
  def __init__(self):
    self.__areaValue = 0
    self.__perimeterValue = 0

  def area(self):
    print("Area ", self.__areaValue)

class square(shape):
    def __init__(self, side):
        shape.__init__(self)
        self.__side = side

    def area (self):
      self.__areaValue = self.__side * self.__side
      print("Area ", self.__areaValue)
```

3 a) `language(java,highLevel).`
`language(java,oop).`

b) i)
```
fortran
cobol
visualBasic
visualBasic
python
python
```
ii) `false`

c) `translator(assembler,X).`

20.2 What you should already know

1

Python

```
#set up file with one record
class TStudentRecord:
    def _init_(self):
        self.name =""
        self.address = ""
        self.className =""

studentFile = open('student.TXT','w')

myStudent = TStudentRecord()
myStudent.name = "Ahmad Sayed"
myStudent.address = "My House"
myStudent.className ="5X"

print ("Name ", myStudent.name)
print ("Address ", myStudent.address)
print ("Class ", myStudent.className)

studentFile.write(myStudent.name + " " + myStudent.address + " " +
myStudent.className + "\n")
studentFile.close()

studentFile = open('student.TXT','r')
print (studentFile.read()) #print all the records in a file
studentFile.close()

#append another record
studentFile = open('student.TXT','a')

myStudent = TStudentRecord()
myStudent.name = "Frank Yang"
myStudent.address = "My bungalow"
myStudent.className ="5X"

print ("Name ", myStudent.name)
print ("Address ", myStudent.address)
print ("Class ", myStudent.className)

studentFile.write(myStudent.name + " " + myStudent.address + " " +
myStudent.className + "\n")
```

```
studentFile.close()

studentFile = open('student.TXT','r')
print (studentFile.read())
studentFile.close()

#delete record
studentFile = open('student.TXT','r')
newStudentFile = open('newStudent.TXT', 'w')

recordRead = studentFile.readline()

while recordRead != '':
        if (recordRead != "Frank Yang" + " " + "My bungalow" + " " + "5X" + "\n"
):
                newStudentFile.write(recordRead)
        recordRead = studentFile.readline()

studentFile.close()
newStudentFile.close()

newStudentFile = open('newStudent.TXT', 'r')
print (newStudentFile.read())
newStudentFile.close()
os.remove('student.TXT')
os.rename('newStudent.TXT','student.TXT')
```

VB

```
'student record
 Imports System.IO
 Module Module1
     Sub Main()
        'set up file with one record
        Dim recordWrite, recordRead As String
        Dim studentFileWrite As StreamWriter
        Dim studentFileRead As StreamReader
        studentFileWrite = New StreamWriter("student.txt")
        Dim myStudent As TStudentRecord

        myStudent.name = "Ahmad Sayed"
        myStudent.address = "My House"
        myStudent.className = "5X"
        Console.WriteLine("Name " + myStudent.name)
        Console.WriteLine("Address " + myStudent.address)
        Console.WriteLine("Class " + myStudent.className)
        recordWrite = myStudent.name + " " + myStudent.address + " " +
myStudent.className
        studentFileWrite.WriteLine(recordWrite)
        studentFileWrite.Close()

        studentFileRead = New StreamReader("student.txt")
        Do
            recordRead = studentFileRead.ReadLine
            Console.WriteLine(recordRead)
        Loop Until recordRead Is Nothing

        studentFileRead.Close()

        'append another record
        myStudent.name = "Frank Yang"
        myStudent.address = "My bungalow"
        myStudent.className = "5X"
        Console.WriteLine("Name " + myStudent.name)
        Console.WriteLine("Address " + myStudent.address)
        Console.WriteLine("Class " + myStudent.className)
        studentFileWrite = New StreamWriter("student.txt", True)
        recordWrite = myStudent.name + " " + myStudent.address + " " +
myStudent.className
        studentFileWrite.WriteLine(recordWrite)
        studentFileWrite.Close()

        studentFileRead = New StreamReader("student.txt")
        Do
            recordRead = studentFileRead.ReadLine
            Console.WriteLine(recordRead)
        Loop Until recordRead Is Nothing

        studentFileRead.Close()

        'delete record
        studentFileRead = New StreamReader("student.txt")
        studentFileWrite = New StreamWriter("newStudent.txt")

        Do
            recordRead = studentFileRead.ReadLine
            If recordRead <> "Frank Yang" + " " + "My bungalow" + " " + "5X"
Then
                recordWrite = recordRead
                studentFileWrite.WriteLine(recordWrite)
            End If
        Loop Until recordRead Is Nothing
        studentFileRead.Close()
        studentFileWrite.Close()


        studentFileRead = New StreamReader("newStudent.txt")
        Do
            recordRead = studentFileRead.ReadLine
            Console.WriteLine(recordRead)
        Loop Until recordRead Is Nothing
        studentFileRead.Close()


        My.Computer.FileSystem.DeleteFile("student.txt")
        My.Computer.FileSystem.RenameFile("newStudent.txt", "student.txt")

        Console.ReadKey()
    End Sub
    Structure TStudentRecord
        Dim name As String
        Dim address As String
        Dim className As String
    End Structure
End Module
```

Java

```
//student record
import java.util.*;
import java.io.*;

class ShouldKnow_20_2_Q1 {
    static class TStudentRecord {
        String name;
        String address;
        String className;
        String gender;

        public void TstudentRecord() {
            name = "";
            address = "";
            className ="";
        }
    }

   public static void main(String args[]){
      TStudentRecord myStudentRecord = new TStudentRecord();
      myStudentRecord.name = "Ahmad Sayed";
      myStudentRecord.address = "My House";
      myStudentRecord.className = "5X";

      System.out.println("Name  " + myStudentRecord.name);
      System.out.println("Address  " + myStudentRecord.address);
      System.out.println("Class " + myStudentRecord.className);

      String studentFileWrite;
      studentFileWrite = myStudentRecord.name + " " + myStudentRecord.address + "
" + myStudentRecord.className;
        try {
            FileWriter studentFileWriter = new FileWriter("student.txt", false);
            PrintWriter studentWriter = new PrintWriter(studentFileWriter);
            studentWriter.printf(studentFileWrite + "\n");
            studentWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
      myStudentRecord.name = "Frank Yang
      myStudentRecord.address = "My Bungalow";
      myStudentRecord.className = "5X";

      System.out.println("Name  " + myStudentRecord.name);
      System.out.println("Address  " + myStudentRecord.address);
      System.out.println("Class " + myStudentRecord.className);

      studentFileWrite = myStudentRecord.name + " " + myStudentRecord.address + "
" + myStudentRecord.className;
        try {
            FileWriter studentFileWriter = new FileWriter("student.txt", true);
            PrintWriter studentWriter = new PrintWriter(studentFileWriter);
            studentWriter.printf(studentFileWrite + "\n");
            studentFileWriter.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
              try {
                  FileReader studentFileReader  = new FileReader("student.txt");
                  BufferedReader studentReader = new
BufferedReader(studentFileReader);
                  String studentFileRead;
                  while ((studentFileRead = studentReader.readLine()) != null) {
                      System.out.println(studentFileRead);
                    }
             studentFileReader.close();
            } catch (IOException e) {
            e.printStackTrace();
            }

            try {
                FileReader studentFileReader  = new FileReader("student.txt");
                BufferedReader studentReader = new
BufferedReader(studentFileReader);
                FileWriter studentFileWriter = new FileWriter("NewStudent.txt",
false);
                PrintWriter studentWriter = new PrintWriter(studentFileWriter);

                String studentFileRead;

                while ((studentFileRead = studentReader.readLine()) != null) {
                    if (!(studentFileRead.equals("Frank Yang My Bungalow 5X")) ){
                        studentWriter.printf(studentFileRead + "\n");
                    }
                }
                studentFileReader.close();
                studentFileWriter.close();
            } catch (IOException e) {
            e.printStackTrace();
            }
        try {
                  FileReader studentFileReader  = new
FileReader("newStudent.txt");
                  BufferedReader studentReader = new
BufferedReader(studentFileReader);
                  String studentFileRead;

                  while ((studentFileRead = studentReader.readLine()) != null) {
                      System.out.println(studentFileRead);
                    }

            } catch (IOException e) {
            e.printStackTrace();
            }
        }
     }
```

2 **Serial** – records are stored in a file one after another in the order they were added to the file.

Sequential – records are stored in a file in a given order based on the key field

Random – records are stored in a file in any available position based a hashing algorithm used on the key field.

3 **Direct** – a record is found using a hashing algorithm without searching any other records.

Sequential – a record is found by searching each record in turn starting at the beginning of the file.

Activity 20L

Python

```
#ACTIVITY 20L
import datetime
import pickle

class student:
   def __init__(self):
         self.name = ""
         self.registerNumber = 0
         self.dateOfBirth = datetime.datetime.now()
         self.fullTime = True

studentRecord = student()

studentFile = open('students.DAT','wb')

print("Please enter student details")
studentRecord.name = input("Please enter student name  ")
studentRecord.registerNumber = int(input("Please enter student's
register number  "))
year = int(input("Please enter student's year of birth YYYY "))
month = int(input("Please enter student's month of birth MM "))
day = int(input("Please enter student's day of birth DD "))
studentRecord.dateOfBirth = datetime.datetime(year, month, day)
studentRecord.fullTime = bool(input("Please enter True for full-time or
False for part-time  "))
pickle.dump (studentRecord, studentFile)
print(studentRecord.name, studentRecord.registerNumber,
studentRecord.dateOfBirth, studentRecord.fullTime)


studentFile.close()

studentFile = open('students.DAT','rb')

studentRecord = pickle.load(studentFile)

print(studentRecord.name, studentRecord.registerNumber,
studentRecord.dateOfBirth, studentRecord.fullTime)

studentFile.close()
```

VB

```
' ACTIVITY 20L
Option Explicit On
Imports System.IO
Module Module1

    Public Sub Main()
         Dim studentFileWriter As BinaryWriter
         Dim studentFileReader As BinaryReader
         Dim studentFile As FileStream

         Dim year, month, day As Integer
         Dim studentRecord As New student()

         studentFile = New FileStream("studentFile.DAT",
FileMode.Create)
         studentFileWriter = New BinaryWriter(studentFile)

        Console.Write("Please enter student name ")
         studentRecord.name = Console.ReadLine()
         Console.Write("Please enter student's register number ")
         studentRecord.registerNumber =
Integer.Parse(Console.ReadLine())
         Console.Write("Please enter student's year of birth YYYY ")
         year =Integer.Parse(Console.ReadLine())
         Console.Write("Please enter student's month of birth MM ")
         month =Integer.Parse(Console.ReadLine())
         Console.Write("Please enter student's day of birth DD ")
         day =Integer.Parse(Console.ReadLine())
         studentRecord.dateOfBirth = DateSerial(year, month, day)
         Console.Write("Please enter True for full-time or False for
part-time ")
         studentRecord.fullTime = Boolean.Parse(Console.ReadLine())

         studentFileWriter.Write(studentRecord.name)
         studentFileWriter.Write(studentRecord.registerNumber)
         studentFileWriter.Write(studentRecord.dateOfBirth)
         studentFileWriter.Write(studentRecord.fullTime)

         studentFileWriter.Close()
         studentFile.Close()

         studentFile = New FileStream("studentFile.DAT", FileMode.Open)
         studentFileReader = New BinaryReader(studentFile)

         studentRecord.name = studentFileReader.ReadString()
         studentRecord.registerNumber = studentFileReader.ReadInt32()
         studentRecord.dateOfBirth = studentFileReader.ReadString()
         studentRecord.fullTime = studentFileReader.ReadBoolean()

         studentFileReader.Close()
         studentFile.Close()
```

```
            Console.WriteLine (studentRecord.name & " " &
studentRecord.registerNumber & " " & studentRecord.dateOfBirth & " " &
studentRecord.fullTime)
            Console.ReadKey ()

    End Sub

    class student:
            Public name As String
            Public registerNumber As Integer
            Public dateOfBirth As Date
            Public fullTime As Boolean
    End Class

End Module
```

Java

```
//ACTIVITY20L no data of birth
import java.io.*;
import java.util.*;


class student {
    private String name;
    private int registerNumber;
    private boolean fullTime;

    student( String n, int r, boolean f){
        name = n;
        registerNumber = r;
        fullTime = f;
    }
    public void showDetails(){
        System.out.println(name + " " + registerNumber + " " + fullTime
);
    }
}

public class ACTIVITY20L {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);

    String nameIn;
    int registerNumberIn;
    boolean fullTimeIn;

    System.out.println("Please student details");
    System.out.println("Please student name ");
    nameIn = myObj.next();
    System.out.println("Please student's register number ");
    registerNumberIn = myObj.nextInt();
    System.out.println("Please enter true for full-time or false
for part-time ");
    fullTimeIn = myObj.nextBoolean();

    student studentRecord = new student(nameIn, registerNumberIn,
fullTimeIn);
    studentRecord.showDetails();
    String registerNumberAsString =
Integer.toString(registerNumberIn);
    String fullTimeAsString = Boolean.toString(fullTimeIn);


    try {
        FileWriter studentFile = new FileWriter("student.txt");
        studentFile.write(nameIn + " " + registerNumberAsString +
" " + fullTimeAsString);
        studentFile.close();
     }
    catch (IOException e) {
      System.out.println("File write error");
      e.printStackTrace();
    }
    try {
      File myStudent = new File("student.txt");
      Scanner myReader = new Scanner(myStudent);
      while (myReader.hasNextLine()) {
        String data = myReader.nextLine();
        System.out.println(data);
      }
      myReader.close();
    } catch (FileNotFoundException e) {
      System.out.println("File read error");
      e.printStackTrace();
    }
  }

}
```

Activity 20M

Sample Python

```
#ACTIVITY 20L updated
import datetime
import pickle

class student:
   def __init__(self):
          self.name = ""
          self.registerNumber = 0
          self.dateOfBirth = datetime.datetime.now()
          self.fullTime = True

studentRecord = student()

studentFile = open('students.DAT','ab')  # a for append

print("Please enter student details")
studentRecord.name = input("Please enter student name  ")
studentRecord.registerNumber = int(input("Please enter student's
register number  "))
year = int(input("Please enter student's year of birth YYYY "))
month = int(input("Please enter student's month of birth MM "))
day = int(input("Please enter student's day of birth DD "))
studentRecord.dateOfBirth = datetime.datetime(year, month, day)
studentRecord.fullTime = bool(input("Please enter True for full-time or
False for part-time  "))
pickle.dump (studentRecord, studentFile)
print(studentRecord.name, studentRecord.registerNumber,
studentRecord.dateOfBirth, studentRecord.fullTime)


studentFile.close()

studentFile = open('students.DAT','rb')

studentRecord = pickle.load(studentFile)

print(studentRecord.name, studentRecord.registerNumber,
studentRecord.dateOfBirth, studentRecord.fullTime)

studentFile.close()
```

Activity 20N

Python

```
#ACTIVITY 20N
try:
    value = int(input('Please enter an integer value '))

except:
    print('Not an integer')
```

VB

```
'ACTIVITY 20N
Module Module1

    Public Sub Main()
        Dim value As Integer
        Try
            Console.Write("Please input an integer value ")
            value = Integer.Parse(Console.ReadLine())
        Catch e As System.FormatException
            Console.WriteLine("Not an integer")
        End Try
        Console.ReadLine()
    End Sub

End Module
```

Java

```
//ACTIVITY20N
import java.io.*;
import java.util.*;

public class ACTIVITY20N {
  public static void main(String[] args) {
        Scanner myObj = new Scanner(System.in);

        try {
            System.out.println("Please input an integer value ");
            int value = myObj.nextInt();
        }

        catch (InputMismatchException e) {
              System.out.println("Not an integer");
        }
    }
}
```

Activity 20O

Python

```
#checking file exists before trying to read from it
try:
        studentFile = open('students.DAT','rb')
except OSError:
        print("Could not open students.DAT")
        sys.exit()
```

Visual Basic

```
'checking file exists before trying to read from it
Try
            studentFile = New FileStream("studentFile.DAT", FileMode.Open)
Catch e As System.IO.IOException
            Console.WriteLine("File not found")
End Try
```

Sample see Java Activity 20L

End of chapter questions

1 a) `parent(ali, ahmed).`
`parent(meena, ahmed).`

b) `P =`
`ahmed`
`aisha`

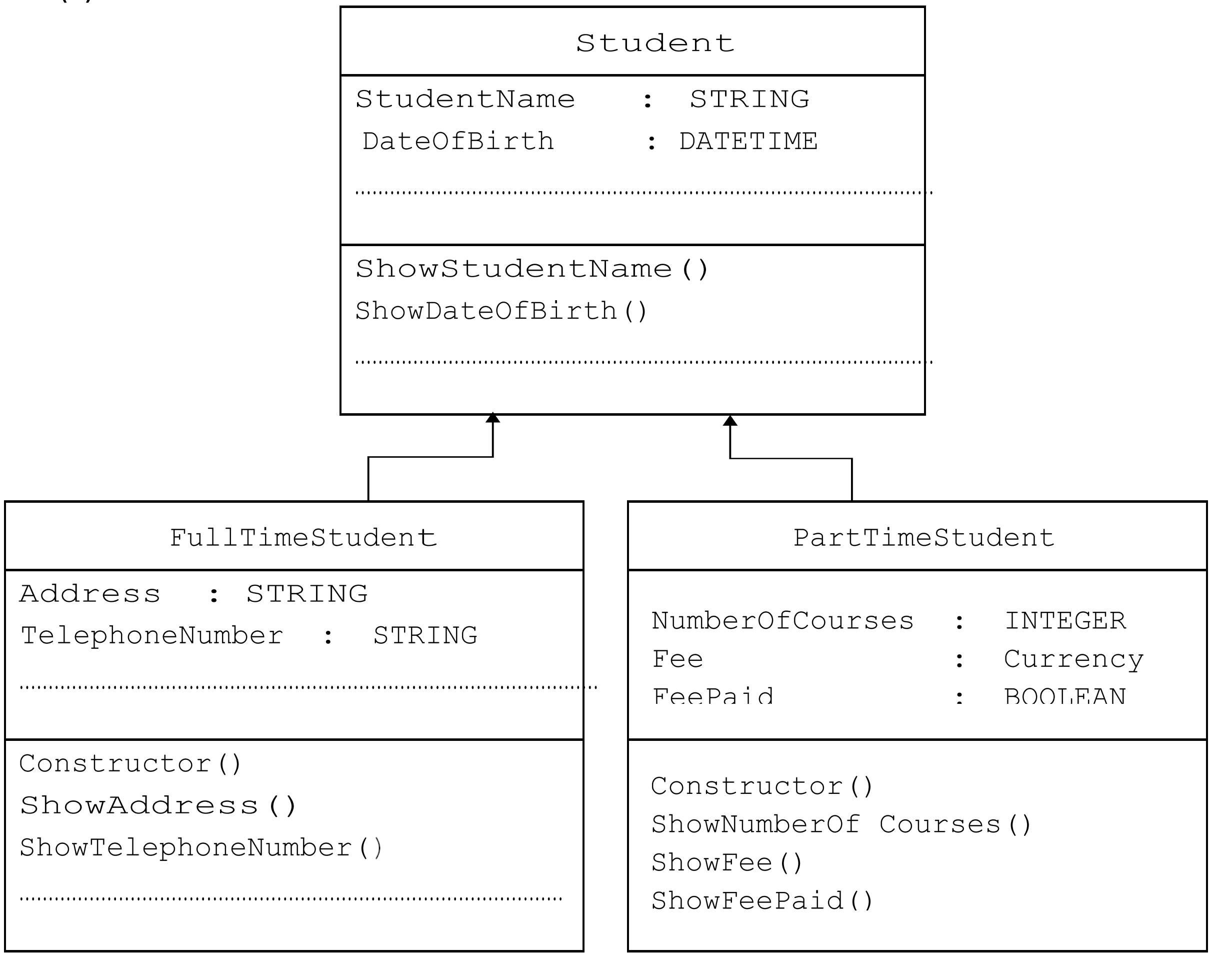
c) `mother(M, gina).`

d)
```
father(F, C)
IF
male(F) AND parent(F, C).
```

e)
```
brother(X, Y)
IF
male(X) AND parent(A, X) AND parent(A, Y) AND NOT X=Y.
```

2 a)

3 (a)


Student
StudentName : STRING
DateOfBirth : DATETIME
ShowStudentName()
ShowDateOfBirth()
FullTimeStudent
Address : STRING
TelephoneNumber : STRING
Constructor()
ShowAddress()
ShowTelephoneNumber()
PartTimeStudent
NumberOfCourses : INTEGER
Fee : Currency
FeePaid : BOOLEAN
Constructor()
ShowNumberOf Courses()
ShowFee()
ShowFeePaid()

b) i)

Python

```
class Student :     def __int__(self) :
        self.__StudentName = ""
          self.__DateOfBirth = ""  # date(1,1,2015)
     def ShowStudentName() :
         pass
     def ShowDateOfBirth() :
         pass
```

VB

```
Class Student
    Public Sub ShowStudentName()
    End Sub
    Public Sub ShowDateOfBirth()
    End Sub
    Private StudentName As String
    Private DateOfBirth As Date
End Class
```

ii)

Python

```
class FullTimeStudent(Student) :     def
__init__(self) :
         self.Address = ""
         self.__TelephoneNumber = ""
     def ShowAddress() :
        pass
     def ShowTelephoneNumber() :
        pass
```

VB.NET

```
Class FullTimeStudent : Inherits Student
    Sub ShowAddress()
    End Sub
    Sub ShowTelephoneNumber()
    End Sub
    Private Address As String
    Private TelephoneNumber As String
End Class
```

iii)

Python

```
NewStudent = FullTimeStudent()
NewStudent.StudentName = "A.Nyone"
NewStudent.DateOfBirth = "12/11/1990"
NewStudent.TelephoneNumber = "099111"
```

VB

```
Dim NewStudent As FullTimeStudent = New FullTimeStudent()
NewStudent.StudentName = "A.Nyone"
NewStudent.DateOfBirth = #11/12/1990#
NewStudent.TelephoneNumber = "099111"
```

3 a) Exception – situation causing a crash/run-time error/fatal error.

Exception handling – code which is called when a runtime error occurs to avoid the program terminating/crashing.

b) Opening a non-existent file.

Using a directory path that does not exist.

Attempting to read past the end of the file.

c) i) `09`

ii) Line 11 catches exceptions (only) between lines 05 and 10 and stops the program from crashing.

Line 12 outputs the exception message if required.