Section 20.2 - File Processing and Exception Handling

Layer 6: High-Order Language

Syllabus Content Section 20: Further Programming

S20.2.1 Write code to perform file-processing operations

- Open (in read, write, append mode) and close a file
- Read a record from a file and write a record to a file
- Perform file-processing operations on serial, sequential, random files

---

> Handling text files

```
OPENFILE <file identifier> FOR <file mode>
READFILE <file identifier>, <variable>
WRITEFILE <file identifier> , <data>
CLOSEFILE <file identifier>
```

READ	for data to be read from the file
WRITE	for data to be written to the file. A new file will be created and any existing data in the file will be lost.
APPEND	for data to be added to the file, after any existing data.

- Example - handling text files

```
//This example uses the operations together, to copy all the lines from FileA.txt to
FileB.txt, replacing any blank lines by a line of dashes.
DECLARE LineOfText : STRING
OPENFILE "FileA.txt" FOR READ
OPENFILE "FileB.txt" FOR WRITE
WHILE NOT EOF("FileA.txt")
        READFILE "FileA.txt", LineOfText
        IF LineOfText = "" THEN
            WRITEFILE "FileB.txt", " --------------------------- "
        ELSE
            WRITEFILE "FileB.txt", LineOfText
        ENDIF
ENDWHILE
CLOSEFILE "FileA.txt"
CLOSEFILE "FileB.txt"
```

> Handling random files

```
OPENFILE <file identifier> FOR RANDOM
SEEK <file identifier>, <address>
GETRECORD <file identifier>, <variable>
PUTRECORD <file identifier>, <variable>
```

- Example – handling random files

```
//The records from positions 10 to 20 of a file StudentFile.Dat are moved to the next position
and a new record is inserted into position 10.
DECLARE Pupil : Student
DECLARE NewPupil : Student
DECLARE Position : INTEGER

NewPupil.LastName ← "Johnson"
NewPupil.Firstname ← "Leroy"
NewPupil.DateOfBirth ← 02/01/2005
NewPupil.YearGroup ← 6
NewPupil.FormGroup ← 'A'

OPENFILE "StudentFile.Dat" FOR RANDOM
FOR Position ← 20 TO 10 STEP -1
        SEEK "StudentFile.Dat", Position
        GETRECORD "StudentFile.Dat", Pupil
        SEEK "StudentFile.Dat", Position + 1
        PUTRECORD "StudentFile.Dat", Pupil
NEXT Position

SEEK "StudentFile.Dat", 10
PUTRECORD "StudentFile.Dat", NewPupil

CLOSEFILE "StudentFile.dat"
```

S20.2.2 Show understanding of an exception and the importance of exception handling

- Know when it is appropriate to use exception handling
- Write program code to use exception handling

```
TRY
        <statementsA>
EXCEPT
        <statementsB>
ENDTRY
```

> Python distinguishes between different types of exception, such as:

- IOError: for example, a file cannot be opened
- ImportError: Python cannot find the module
- ValueError: an argument has an inappropriate value
- KeyboardInterrupt: the user presses Ctrl+C or Ctrl+Del
- EOFError: a file-read meets an end-of-file condition
- ZeroDivisionError: a division by zero has been attempted

> Java distinguishes between different types of exception, such as:

- IOException: for example, a file cannot be opened
- ArithmeticException: an arithmetic error occurred, such as division by zero

- ArrayIndexOutOfBoundsException: the program tries to access an array element outside the boundary