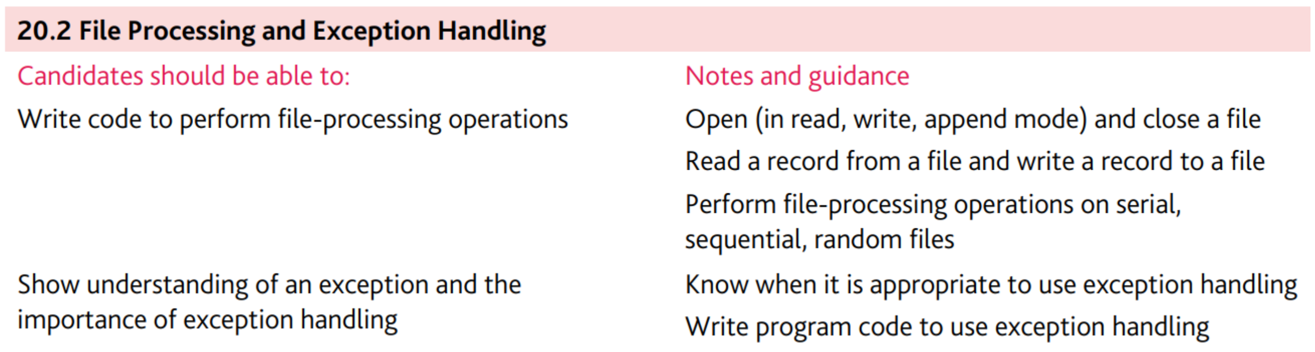
1. File Processing



20.2 File Processing and Exception Handling	
Candidates should be able to:	Notes and guidance
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
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

---

File Processing

Pseudocode

Program code

`Scanner`

> API: [https://docs.oracle.com/javase/17/docs/api/java/util/Scanner.html]

`Scanner` 和其他读入的一个不同点就是他是存在于 `java.util` 包下面的 而其他大多数是在 `java.io` 包下面。

有一个好处就是 `scanner.hasNext` 可以用来检查文件是否读完

Declaration/construction

```
Scanner sca1 = new Scanner(System.in);//从console输入
Scanner sca2 = new Scanner(new File("SamuelJie.txt"));//can be used to read a file
Scanner sca3 = new Scanner("Samuel is a hardworking boy");//can read a string
//then you can carry out other operations
int number = sca1.nextInt();
```

Methods:

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various `next` methods.

简单来说 被空格、换行符所分割的 每一个部分 都算一个`token` 比如`"1 2 3 Samuel"` 中 `1`，`2`，`3`，`"Samuel"`都算`token`. `1` 可以用`nextInt()` 来读取，`"Samuel"`可以用 `next()`来读取。

- `hasNext()`
  - Returns true if this scanner has another token in its input.
- `next()`
  - Finds and returns the next complete token from this scanner.
- `nextLine()`
  - Advances this scanner past the current line and returns the input that was skipped.
- `nextInt()`
  - Scans the next token of the input as an `int`.

`FileReader`

> API: [https://docs.oracle.com/javase/17/docs/api/java/io/FileReader.html]

通常在考试中FileReader都是作为创建BufferedReader的一个前置来使用。

Declaration/construction

```
FileReader(File file);//can construct with a file
FileReader(String Filename);//can construct with the filename
//for example
String fileName="input.txt";
FileReader fr = new FileReader(filename);
```

`BufferedReader`

> API: [https://docs.oracle.com/javase/17/docs/api/java/io/BufferedReader.html]

Buffered Reader 相对于 传统Reader的好处就是 当读入一个文件的时候 能够先把文件的内容存在Buffer(缓存)里面 然后每次从Buffer里面读取，这样比传统Reader直接从文件里读取更加高效

Declaration/construction

```
File fi = new File(fileName);
FileReader fr = new FileReader(fi);
BufferedReader bfr = new BufferedReader(fr);
```

Methods(考试会用到的):

`readLine()`: Reads a line of text. A line is considered to be terminated by any one of a line feed (‘\n’), a carriage return (‘\r’), or a carriage return followed immediately by a linefeed.

```
public String readLine()
//Usage
String dataRead = bfr.readLine();//bfr is the BufferedReader declared previously
```

`RandomAccessFile`

> API: [https://docs.oracle.com/javase/17/docs/api/java/io/RandomAccessFile.html]

Instances of this class support both reading and writing to a random access file. A random access file behaves like a large array of bytes stored in the file system. There is a kind of cursor, or index into the implied array, called the `file pointer`; input operations read bytes starting at the file pointer and advance the file pointer past the bytes read. If the random access file is created in read/write mode, then output operations are also available; output operations write bytes starting at the file pointer and advance the file pointer past the bytes written. Output operations that write past the current end of the implied array cause the array to be extended. The file pointer can be read by the `getFilePointer` method and set by the `seek` method.

Data type	Size
`byte`	1 byte
`short`	2 bytes
`int`	4 bytes
`long`	8 bytes
`float`	4 bytes
`double`	8 bytes
`boolean`	1 **bit**
`char`	2 bytes

Declaration/construction

```
RandomAccessFile(File file, String mode);
RandomAccessFile(String fileName, String mode);
```

`mode`

`mode` 本质上就是读写模式，像是pseudocode 中 OPENFILE `<file Name>` FOR `<File mode>` 中的 `<File mode>` 一样。

- `"r"` open for read only.
- `"rw"` open for read and write

Methods(考试会用到的)

- `read()`
  - Reads a byte of data from this file.
- `readBoolean()`
  - Reads a `boolean` from this file.
- `readByte()`
  - Reads a signed eight-bit value from this file.
- `readChar()`
  - Reads a character from this file.
- `readDouble()`
  - Reads a `double` from this file.
- `readInt()`
- `readUTF()`

- `seek(long pos)`
  - Sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
  - Inputs the number, representing the position that you want the `file pointer` to be.
- `writeInt(int num)`
- `writeChars(String str)`
  - Writes a string to the file as a sequence of characters.
- `writeDouble(String str)`
  - Writes a string to the file using modified UTF-8 encoding in a machine-independent manner.

Example code(from Kamana):

```
import java.io.*;

public class RandomFileDemo {

        public static void main(String[] args)throws Exception {
                RandomAccessFile raf=new RandomAccessFile("file.dat",
"rw");
                raf.writeChar('a');
                raf.writeChar('b');
                raf.writeChar('c');
                raf.writeChar('d');
                raf.writeChar('e');
                raf.writeUTF("kamna");
                raf.writeInt(50);
                raf.writeDouble(50.555);
                raf.writeChar('f');
                raf.close();

                RandomAccessFile r=new RandomAccessFile("file.dat",
"r");
                r.seek(29);
                //System.out.println(r.readChar());
                //System.out.println(r.readChar());
                //r.seek(6);
                //System.out.println(r.readChar());
                //System.out.println(r.readChar());
                //System.out.println(r.readChar());
                //System.out.println(r.readUTF());
                //System.out.println(r.readInt());
                //System.out.println(r.readDouble());
                System.out.println(r.readChar());
                r.close();
        }

}


import java.io.*;
public class randomFileusingArray {


                public static void main(String args[]) throws
IOException {

                        //write the alphabet to a random access file
                        char[] alphabet = {'A', 'B', 'C', 'D', 'E', 'F',
'G', 'H', 'I', 'J',
                                                          'K', 'L',
'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
                                                          'U', 'V',
'W', 'X', 'Y', 'Z'};


                        RandomAccessFile random = new
RandomAccessFile("RandomTest.dat", "rw");

                        System.out.println("Writing alphabet to the
file!");

                        for(int i = 0; i < alphabet.length; i++)
                                random.writeChar(alphabet[i]);

                        System.out.println("Finished writing!");

                        random.close();

                        //reading characters from the alphabet off the
file
                        RandomAccessFile rand0m = new
RandomAccessFile("RandomTest.dat", "r");
```

```
                        System.out.println("Beginning to read!");

                        rand0m.seek(8); // 5th character (E)
                        System.out.println(rand0m.readChar());
                        rand0m.seek(24); //13th character (M)
                        System.out.println(rand0m.readChar());
                        rand0m.seek(40); //21st character (U)
                        System.out.println(rand0m.readChar());
                        rand0m.seek(19); //won't read correctly!
                        System.out.println(rand0m.readChar());
                        System.out.println("Done reading!");
                        rand0m.close();
                }
        }
}
```