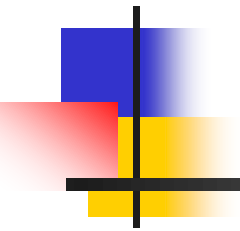


Random File

경복대학교
소프트웨어융합과
배희호 교수

Random Access File

- 대부분의 입출력 Stream들은 File에 순차적(Sequential)으로 입출력 작업을 수행
- Random Access File은 File 내용에 대한 비순차적(Non Sequential) 또는 임의 Access를 허용
- File을 임의로 Access하려면 File을 열고 특정 위치를 찾은 다음 해당 File에서 읽거나 씀
- Random Access File은 Byte의 큰 배열처럼 동작
- File Pointer라는 배열에 암시 된 Cursor가 있으며, Cursor를 이동하여 읽기 쓰기 작업을 수행
- EoF보다 원하는 Byte 수를 읽기 전에 File 끝에 도달하면 EoFException이 발생하는데, 이것은 IOException의 한 유형

Random Access File

일반 파일 열기

파일내용	h	e	l	l	o	w	o	r	l	d
인덱스	0	1	2	3	4	5	6	7	8	9

파일쓰기 완료 후 현재 파일 포인터 위치 (EOF)
앞으로 되돌아갈 수가 없음

RandomAccessFile 파일 열기

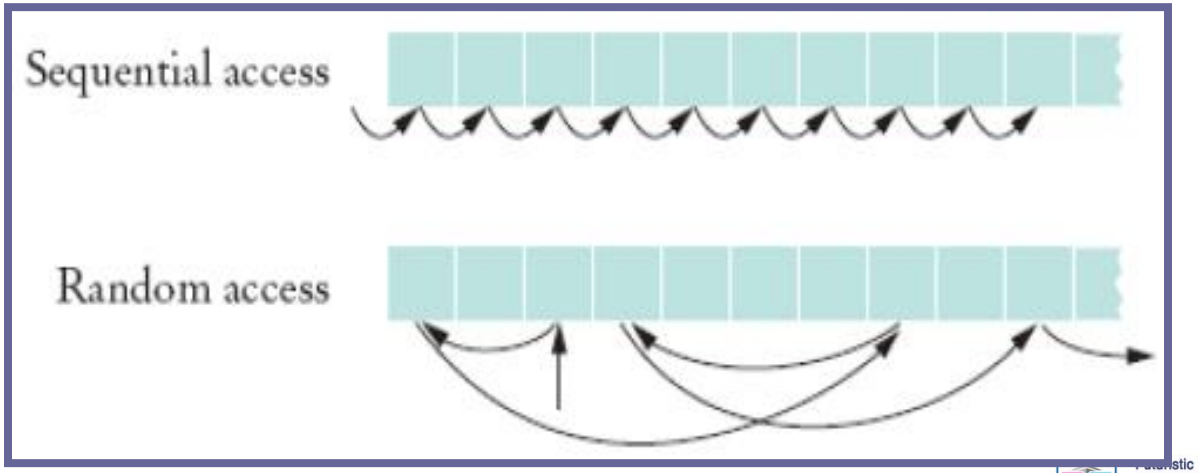
파일내용	h	e	l	l	o	w	o	r	l	d
인덱스	0	1	2	3	4	5	6	7	8	9

파일 포인터 위치 재지정 (seek)

파일쓰기 완료 후 현재 파일 포인터 위치 (EOF)
임의 위치로 되돌아갈 수 있음

Random Access File

- File을 읽고 쓰는 현재 위치를 원하는 곳으로 변경하여 임의의 위치에서 입 출력 가능
  - 직접 액세스 파일(Direct Access File)이라고도 함
  - 순차 액세스 파일(Sequential Access File)보다 효율적임
- 일반적으로 Binary Mode에서만 임의 접근을 사용
  - Text Mode에서는 특정문자에 대해 변환이 일어나므로 정확한 이동이 보장 안됨
- RandomAccessFile 클래스 사용



Random Access File

- 필요성
  - 실시간 응용 Program에는 즉각적인 응답이 필요
    - 예) 청구서에 대한 고객 문의에 응답
  - 계정 기록을 통한 시퀀싱은 시간 집약적임
  - 무작위(즉각적인) 액세스가 실시간 요구를 충족시킴
  - 원하는 Record를 직접 읽거나 쓸 수 있음

RandomAccessFile

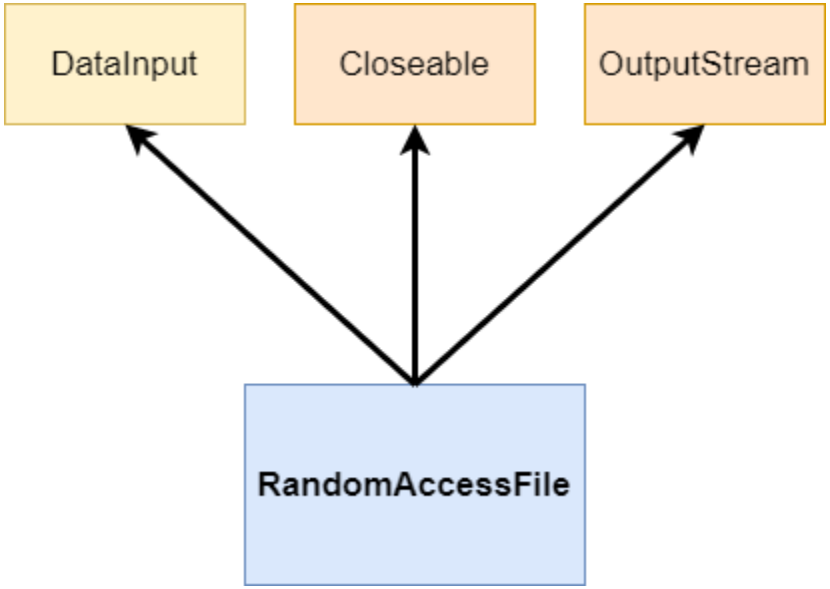
- 장점
  - BufferedReader의 skip()과 동일한 기능을 제공하는 RandomAccessFile의 seek()는 pointer만을 이동시킴으로써 건너 뛰는 Data들을 Memory에 올리지 않음
  - getChannel() 함수를 통해서 FileChannel을 얻을 수 있어 NIO 방식의 File IO 지원이 가능
- 단점
  - 성능이 떨어지고 한글 처리가 어려움
  - byte array를 그대로 전달하는 측면에서는 해당 단점이 구현하는 입장에서는 상관없지만, readLine()을 통해서 데이터를 읽을 경우에는 인코딩 타입이 ISO-8859-1 이 되기 때문에 추가적인 인코딩 처리가 필요
  - 예) UTF-8 방식의 인코딩으로 처리를 하기 위해서는 아래와 같이 처리를 해줘야 하기 때문에 성능이 떨어짐

RandomAccessFile 클래스

- File에 임의로(random) 접근할 수 있음
- DataInput, DataOutput Interface를 사용(포함)
  - Interface에 선언된 모든 메소드가 구현되어 있으므로 사용 가능

RandomAccessFile 클래스

- 입력 Stream과 출력 Stream의 2가지 기능을 가지고 있는 Stream
  - RandomAccessFile은 예외적으로 RandomAccessFile 이라는 객체 하나로 쓰기와 읽기 둘 다 할 수 있음
  - RandomAccessFile 클래스는 DataInput, 그리고 OutputStream을 둘 다 implements 했기 때문에 읽기, 쓰기가 가능



RandomAccessFile 클래스

- File의 임의의 위치에 직접 접근할 수 있는 방법을 제공
- RandomAccessFile 클래스는 File만을 대상으로 하며, 임의의 지점에서 입출력을 동시에 수행할 수 있음
- RandomAccessFile 클래스는 사실상 JAVA IO의 일부가 아님
- 컨트롤의 대상이 File이기 때문에 IO와 함께 언급되는 것이 일반적 임
- 편의상 Stream으로 분류하기도 하지만, 엄밀히 말해서 Stream은 아님
- Stream(w/흐름, 순서)은 임의의 위치에 Data를 읽고 쓸 수 없음

RandomAccessFile 클래스

- RandomAccessFile 클래스에는 Input 및 OutputStream과 동일한 read(), write() 및 close() 메소드가 포함되어 있음
- Data를 읽거나 쓰기 전에 파일 내의 시작 위치를 선택할 수 있는 seek() 메소드가 있음
- 원시 타입 값, 바이트 배열 및 문자열 읽기 및 쓰기 기능을 포함하고 있음
- stream / reader/ writer 모델과 호환되지 않음
- 무작위 액세스 파일을 사용하면 원하는 위치를 찾고 바이트 수를 읽고 쓸 수 있음
- 파일의 시작 부분을 기준으로 탐색만 지원
  - 파일 포인터의 현재 위치와는 관련이 없음
  - 그러나 현재 위치를 보고하는 방법이 있음

RandomAccessFile 클래스

- 기존 입출력 스트림 클래스 vs RandomAccessFile 클래스

구분	설명
입출력 스트림 클래스	✓ 파일 내용을 순차적으로만 처리할 수 있음 ✓ 입출력 스트림으로 파일의 전체 내용을 출력하거나 복사하는 경우 처음부터 끝까지 순차적으로 읽어내는 과정을 거치게 됨
RandomAccessFile 클래스	✓ RandomAccessFile 클래스는 파일의 임의의 위치에 직접 접근할 수 있는 방법을 제공 ✓ 파일의 중간 부분에 특정 내용을 변화시키는 처리만 할 경우 RandomAccessFile 클래스를 이용하면 파일의 필요한 부분에 바로 접근하여 원하는 처리를 할 수 있음

RandomAccessFile 클래스

- RandomAccessFile class는 입출력 클래스 중 유일하게 파일에 대한 입력과 출력을 동시에 할 수 있는 클래스
- RandomAccessFile 클래스는 파일만을 대상으로 함
- 파일 포인터가 있어서 읽고 쓰는 위치의 이동이 가능 (seek 메소드를 활용)
- 순차적인 접근이 아닌 임의의 지점에 접근하여 작업을 수행하고 싶다면 사용하기 좋음
- RandomAccessFile 클래스 생성자는 인수로 File 이름뿐만 아니라 File Mode까지 함께 전달해야 함
- 파일 모드
  - r : 파일을 오로지 읽는 것만 가능한 모드로 개방
  - rw : 파일을 읽고 쓰는 것이 모두 가능한 모드로 개방함. 만약 파일이 없으면 새로운 파일을 생성

RandomAccessFile 클래스

- RandomAccessFile 클래스는 파일을 랜덤하게 읽고 쓸수 있는 클래스
- RandomAccessFile 클래스를 생성시 mode 옵션으로 읽기 전용 또는 읽고 쓰기가 가능한 클래스

- 생성자

생성자	설명
RandomAccessFile (File file, String mode)	File 인수로 지정된 파일에서 읽고 선택 사항으로 줄 수 있는 임의 액세스 파일 스트림을 생성
RandomAccessFile (String name, String mode)	지정된 이름의 파일을 읽고 선택적으로 쓰는 임의 액세스 파일 스트림을 생성

RandomAccessFile 클래스

- mode 속성

모드	설명
r	읽기 전용으로 연다
rw	읽기와 쓰기 동시에 가능 하도록 연다
rwd	데이터가 변형되었을 때 자동으로 강제적으로 데이터를 집어 넣는 방식 // file만 변한 경우
rws	데이터가 변형되었을 때 자동으로 강제적으로 데이터를 집어 넣는 방식 // file + metadata(파일의 주변 데이터) 동시에 변한 경우

- RandomAccessFile이 읽기 전용 모드로 작성되면 FileNotFoundException이 발생
- RandomAccessFile이 읽기 – 쓰기로 생성되면 길이가 0인 파일이 자동 생성

RandomAccessFile 클래스

- 메소드

메소드	설명
public int read()	현재 파일 포인터에서 내용을 읽음
public int read(byte[] b, int off, int len)	데이터를 len 만큼을 읽어 byte[] b의 off 위치에 저장하고 읽은 바이트 수를 반환
public final int readInt( )	현재의 파일 포인터에서 데이터를 읽어 int 형으로 반환
final String readLine()	현재의 파일 포인터에서 한 라인을 읽음
public final double readDouble( )	현재의 파일 포인터에서 데이터를 읽어 double형으로 반환
public void write(int b)	현재의 파일 포인터에서 int형을 씀
public void write(byte[] b, int off, int len)	현재의 파일 포인터에서 바이트 배열을 씀

RandomAccessFile 클래스

- 메소드

메소드	설명
public final void writeInt (int v)	현재의 파일 포인터에서 정수형을 씀
public final void writeDouble (double v)	현재의 파일 포인터에서 실수형을 씀
public long getFilePointer( )	파일의 위치정보를 얻는 메소드 파일 포인터(파일 내에서 현재 제어되고 있는 부분)의 위치를 반환
public void seek(long pos)	인자로 지정한 위치로 파일 포인터의 위치를 변경하는 메소드
skipBytes(int)	지정된 바이트 만큼 파일 포인터를 앞쪽으로 이동
long length()	파일의 길이를 바이트 단위로 반환
void close()	파일 객체 닫기

RandomAccessFile 클래스

- File Pointer
  - RandomAccessFile은 File의 현재 위치를 나타내는 파일 포인터를 지원
  - File을 처음 만들 때 File Pointer는 0으로 설정되어 File의 시작을 나타냄
  - 읽기 및 쓰기 메소드 호출은 읽거나 쓴 Byte 수로 파일 포인터를 조정
  - 파일의 끝은 EOF(-1)의 값을 갖음

RandomAccessFile 클래스

- File Pointer 조작 메소드
  - RandomAccessFile에는 File Pointer를 명시적으로 조작하는 3가지 메소드
  - int skipBytes(int n)
    - File Pointer를 지정된 Byte 수만큼 앞으로 이동
  - void seek (long pos)
    - File Pointer를 지정된 Byte 바로 앞에 배치
  - long getFilePointer()
    - File Pointer의 현재 Byte 위치를 반환

RandomAccessFile 클래스

- java.io.RandomAccessFile.skipBytes(int n) 메소드
  - 건너뛴 Byte를 삭제하는 n Byte의 입력을 건너뛰려고 시도
  - 이 메소드는 일부 적은 수의 Byte를 건너뛰고 0일 수 있음
  - n Byte를 건너 뛰었기 전에 File의 끝에 도달하는 것은 하나의 가능성일 뿐임
  - 이 메소드는 EOF예외를 던지지 않음
  - 건너뛰는 실제 Byte 수가 반환
  - n이 음수인 경우 Byte를 건너 뛰지 않음

RandomAccessFile 클래스

- seek(long pos) 메소드
  - pos
    - File Pointer를 설정하는 것으로 File의 시작 부분에서 Byte 단위로 측정된 Offset 위치
  - File에서 다음 읽기 또는 쓰기가 발생하는 이 파일의 시작 부분에서 측정된 File Pointer Offset을 설정함
  - Offset은 File의 끝을 넘어 설정될 수도 있음
  - Offset을 File 끝 이상으로 설정해도 File 길이가 변경되지 않음
  - File 길이는 Offset이 File의 끝을 넘어 설정된 후에 작성하면 변경됨

RandomAccessFile 클래스

- File Pointer를 특정 Byte로 이동하려면

```
file.seek(n)
```

- File Pointer의 현재 위치를 얻으려면

```
long n = file.getFilePointer()
```

- File의 Byte 수를 찾으려면

```
long filelength = file.length()
```

RandomAccessFile 클래스

- RandomAccessFile도 readLine()을 지원
  - readLine()을 통해서 Data를 읽을 경우 Data의 Encoding Type이 ISO-8859-1이 됨
  - 즉, ISO-8859-1외의 Encoding 방식을 사용하고 있는 경우에는 이의 변환이 필요하게 됨

```
RandomAccessFile reader = new RandomAccessFile
                                        (strFileName, "r");
String line = new String(reader.readLine().
                              getBytes("ISO-8859-1"),"UTF-8");
```

  - 위에서 Bytes변환 후 다시 String으로 만들어야 함
  - 이는 당연히 엄청난 성능 저하를 가져옴(BufferedReader에 비해 10배). 그러나 위와 같은 변환을 거치지 않는다면 한글 처리가 불가능함
  - 즉, 성능적인 측면 때문에 BufferedReader의 대체 클래스로는 사용이 불가능

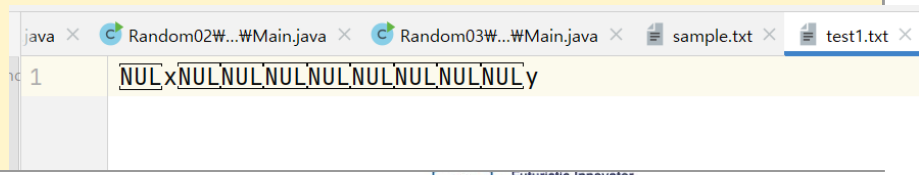
Random Access File 예제 1

```
public static void main(String[] args) {
    String inputfile = ".₩₩data₩₩test.txt";
    File file = new File(inputfile);
    if (file.exists()) {
        try {
            RandomAccessFile input = new RandomAccessFile(file, "r");
            input.seek(3);
            System.out.println((char) input.read());
            input.seek(6);
            System.out.println((char) input.read());
            input.seek(0);
            System.out.println((char) input.read());
            input.close();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    } else {
        System.out.println(file + "이 존재하지 않아요");
    }
}
```

```
    ----+----1----+----2----+
1   Kyungbok University
2   Smart IT
```

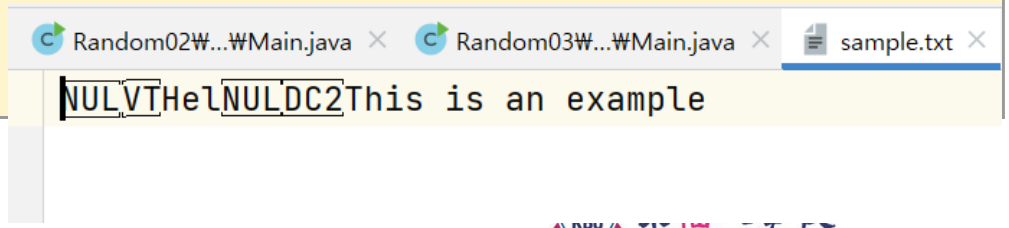
Random Access File 예제 2

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩";

    try {
        RandomAccessFile file = new RandomAccessFile(path +"test1.txt", "rw");
        System.out.println(file.length());
        System.out.println("file pointer : " + file.getFilePointer());
        file.seek(1);
        file.write('x');
        file.seek(10);
        file.write('y');
        System.out.println(file.length());
        System.out.println("file pointer : " + file.getFilePointer());
        file.close();
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```



Random Access File 예제 3

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩";

    try {
        RandomAccessFile random =
                                    new RandomAccessFile(path + "sample.txt", "rw");
        random.writeUTF("Hello World");
        random.seek(0);
        System.out.println("" + random.readUTF());
        random.seek(5);
        random.writeUTF("This is an example");
        random.seek(0);
        System.out.println("" + random.readUTF());
    } catch (IOException ex) {
        System.out.println(ex.getMessage());
    }
}
```



Random Access File 예제 4(1/3)

```
public static void main(String[] args) {
    String page = ".₩₩data₩₩";

    try {
        RandomAccessFile random  = new RandomAccessFile(
                                                    page + "member.txt","rw");
        boolean isUse = true;
        String name = "홍길동";
        char gender = 'M';
        String address = "전북 전주";
        String homeTel = "063-123-1234";
        String phoneTel = "010-123-1234";
        int group = 4;
        String memo = "정보영상진흥원 거시기 머시기 저시기 가시기 나시기";
```

Random Access File 예제 4(2/3)

```
System.out.println(random.getFilePointer());
random.writeBoolean(isUse);
System.out.println(random.getFilePointer());
random.writeUTF(name);
System.out.println(random.getFilePointer());
random.writeChar(gender);
System.out.println(random.getFilePointer());
random.writeUTF(address);
System.out.println(random.getFilePointer());
random.writeUTF(homeTel);
System.out.println(random.getFilePointer());
random.writeUTF(phoneTel);
System.out.println(random.getFilePointer());
random.writeInt(group);
System.out.println(random.getFilePointer());
random.writeUTF(memo);
System.out.println(random.getFilePointer());
```

Random Access File 예제 4(3/3)

```
            System.out.println("크기 : "+ random.length());   //134
            random.seek(0); //찾아가는 위치는 절대값(양수 위치만 존재함)
            System.out.println(random.readBoolean());    //true
            System.out.println(random.readUTF());         //홍길동
            System.out.println(random.readChar());        //M
            System.out.println(random.getFilePointer());  //14
            random.seek(random.length());                      //132로 이동
            System.out.println(random.getFilePointer()); //132
            random.seek(14);
            System.out.println(random.getFilePointer()); //14
            System.out.println(random.readUTF());        //전북 전주
            System.out.println(random.getFilePointer()); //29
            random.writeUTF("끝에 추가함 AAA");
            System.out.println(random.getFilePointer()); //50
            random.close();
        } catch (FileNotFoundException e) {
            System.out.println("파일 없음");
        } catch (IOException e) {
            System.out.println("오류");
        }
    }
```

Random Access File 예제 5(1/2)

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩";

    try {
        File imsiFile = File.createTempFile("data", ".tmp", new File(path));
        System.out.println(imsiFile.getAbsolutePath());
        imsiFile.deleteOnExit();
        FileOutputStream output = new FileOutputStream(imsiFile);
        for (int i = 0; i <= 100; i++) {
            output.write(i);
        }
        output.close();
```

Random Access File 예제 5(2/2)

```
        RandomAccessFile input = new RandomAccessFile(imsiFile, "r");
        for (int i = 100; i >= 0 ; i -= 10) {
            input.seek(i);
            System.out.print(input.readByte());
            System.out.print(" ");
        }
        System.out.println();
        input.close();
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

Random Access File 예제 6(1/2)

```
public static void main(String[] args) {
    String path = ".₩₩data₩₩";
    int seekSize = 5;

    File file = new File(path + "test.txt");
    if (file.exists()) {
        try {
            RandomAccessFile random = new RandomAccessFile(file, "r");
            String line;
            while ((line = random.readLine()) != null) {
                System.out.println(line);
            }
            System.out.println("total length : " + random.length());
```

Random Access File 예제 6(2/2)

```
            long size = random.length() / seekSize +
                    (random.length() % seekSize == 0 ? 0 : 1);
            for (int i = 0; i < size; i++) {
                byte[] data = new byte[seekSize];
                random.seek((long) i * seekSize);
                random.read(data);
                System.out.printf("pointer : %02d str : %s₩n",
                        random.getFilePointer(), new String(data).trim());
            }
            random.close();
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    } else {
        System.out.println(file + "이 존재하지 않아요");
    }
}
```

Random Access File 예제 5

```
public class  ReverseFile {
    public static void main(String[] args) throws IOException {
        RandomAccessFile file = new
                                RandomAccessFile(".//data//test.txt","r");
        int ch;
        long fileSize = file.length();
        long pos = fileSize - 1;  //뒤로 부터(맨뒤는 EOF)
        while (true) {
            if (pos < 0) {
                        break;
           }
            file.seek(pos);   // 파일 포인터 이동
            ch = file.readByte();
           System.out.printf("%c", ch);
            pos--;                  // 파일 포인터 위치값 감소 (앞으로)
        }
        file.close();
    }
}
```