WIPI JAVA 강좌 – 5

org.kwis.msp.db/io pacakge(II)
– file & db

1. 개 요
2. FileSystem
3. File
4. Database

박정환(tmozi@xce.co.kr)

1. 개요

- WIPI에서 FileSystem의 특징
  - 위피는 FileSystem을 지원하기 위한 클래스를 가지고 있으며 이를 위해 File과 FileSystem클래스를 가지고 있다.
  - 3가지의 접근방법을 제공하여 사용자, 관리자가 따로 파일을 관리할 수 있다.
  - 디렉토리 구조를 가지며 삭제 및 추가가 가능하다.
  - byte 배열로 저장된다.
  - File은 입력, 출력, 입출력 모드로 파일을 열 수 있고 그에 따라 사용방법을 결정할 수 있다.
  - 파일의 마지막에서 read시 -1(EOF)를 발생
- DataBase기능 구현
  - 위피에서는 Database를 구현하여 MIDP의 rms와 비슷하게 record 저장 방식을 가지고 있다.
  - 필터와 비교방법을 가지고 있어 특정 레코드를 검색하거나 정렬을 쉽게 해준다.

2. FileSystem 클래스(I)

- 클래스 설명
  - 파일의 생성, 삭제, 변경등을 관리해 주는 클래스
  - 인스턴스 없이 static으로 직접 사용
  - 3가지 File접근 방법이 정의되어 있다.
    - PRIVATE_ACCESS : 응용프로그램 자신만이 쓰는 접근방법
    - SHARD_ACCESS : 다른 프로그램과 공유하기 위한 접근 방법
    - SYSTEM_ACCESS : 시스템 디렉토리로 접근하기 위한 방법 (사용 제한되어 있다.)
  - 파일의 최대 길이가 정의 (MAX_FILENAME_LENGTH)
  - 파일의 경로 : 파일의 경로는 구현된 VM마다 차이가 있을 수 있다. 하지만, 주로 classpath를 따른다.

2. FileSystem 클래스(II)

- 주요메소드 : 모두 static메소드로 선언되어 있으며 파일(디렉토리)이름과 접근모드값을 들어간다.
  - exists(String, int)
  - isDirectory(String, int)
  - isFile(String, int)
  - list(String, int)
  - mkdir(String, int)
  - rmdir(String, int)
  - remove(String, int)
  - rename(String, int)

실제사용예)

```
try{
  boolean b = FileSystem.exists(
          “filename.txt”,
          FileSystem.PRIVATE_ACCESS);
  if(b) FileSystem.remove(
          “filename.txt”,
          FileSystem.PRIVATE_ACCESS);
}catch(IOException ie){}
```

3. File 클래스(I)

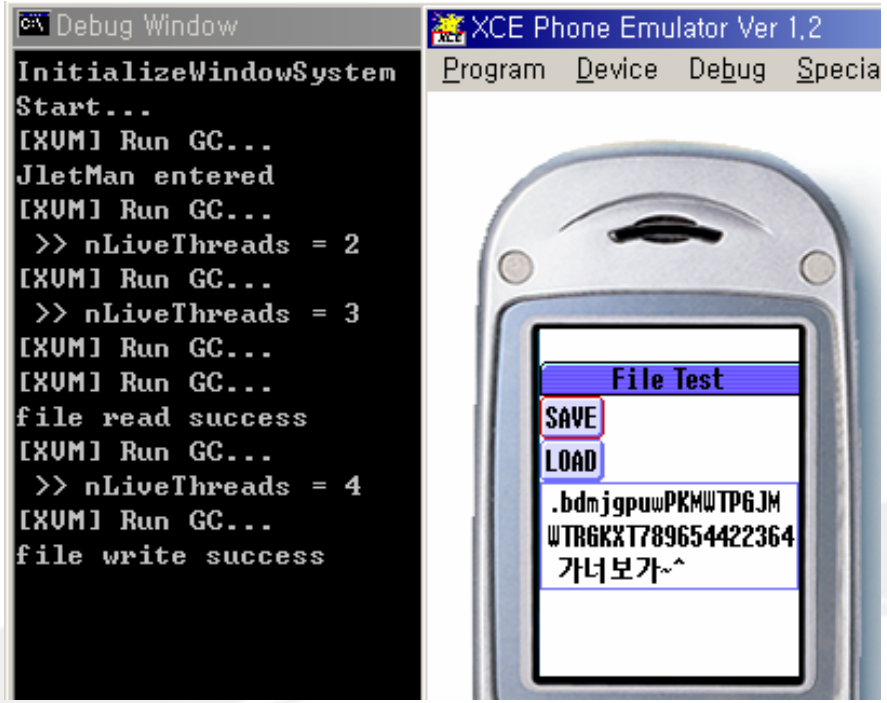
- 클래스 설명
  - 파일을 생성하거니 기존의 파일을 읽을때 사용
  - read, write명령을 가지며 Stream으로 동작한다.
  - seek값을 가지고 있어 file에서 이동이 가능
  - 파일 객체 생성시 파일을 여는 모드가 존재
    - READ_ONLY
    - WRITE
    - WRITE_TRUNC
    - READ_WRITE

실제사용예)

```
try{
  File file = new File("test.txt", File.WRITE);
  file.wirte(byteBuffer, 0, byteBuffer.length);
}catch(IOException ie){}
```

2. File 클래스(II)

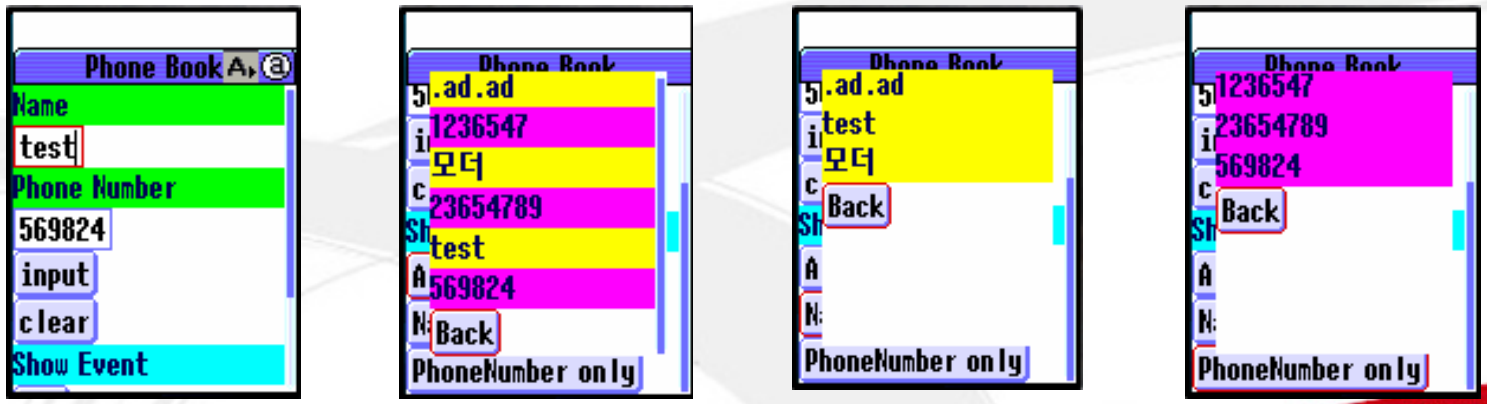
- 주요 메소드
  - File(String, int)
  - read(byte[])
  - read(byte[], int, int)
  - write(byte[])
  - write(byte[], int, int)
  - write(int)
  - seek(int)
  - sizeOf()




```
00000000h: 2E 62 64 6D 6A 67 70 75 77 50 4B 4D 57 54 50 47 ; .bdmjgpuwPKMWTPG
00000010h: 4A 4D 57 54 52 47 4B 58 54 37 38 39 36 35 34 34 ; JMWTRGKXT7896544
00000020h: 32 32 33 36 34 20 B0 A1 B3 CA BA B8 B0 A1 7E 5E ; 22364 가너보가~^
```

4. Database 클래스(I)

- 클래스 설명
  - 레코드 단위로 저장
  - 레코드는 바이트 배열로 한 데이터 베이스의 모든 레코드의 길이는 일정
  - 레코드 ID를 int로 가지며 0부터 사작하여 1씩 증가한다.
  - DataFilter와 DataComparator 인터페이스를 구현하여 정렬 및 특정 레코드를 골라낼 수 있다.
  - 레코드 길이가 항상일정하여 빈공간이 남는다.
  - 하나의 Jlet당 여러 개의 db를 생성할수 있다.



4. Database 클래스(II)

- 레코드 저장예(DataBaseTest.java 참조)
  - 레코드는 12바이트로 설정
  - 6개의 레코드가 존재
  - 첫번째 바이트는 이름일 경우 N, 전화번호일 경우 P
  - 첫번째 바이트는 Data를 filter할때 사용
  - 이름만 선택할 경우 DataComporatorString으로 2번째 바이트부터 비교
  - 번호만 선택할 경우 DataComporatorInteger로 2번째 바이트부터 비교

```
00000000h: 4E 2E 61 64 2E 61 64 00 00 00 00 00 50 31 32 33 ; N.ad.ad.....P123
00000010h: 36 35 34 37 00 00 00 00 4E B8 F0 B4 F5 00 00 00 ; 6547....N모더...
00000020h: 00 00 00 00 50 32 33 36 35 34 37 38 39 00 00 00 ; ....P23654789...
00000030h: 4E 74 65 73 74 00 00 00 00 00 00 00 50 35 36 39 ; Ntest.......P569
00000040h: 38 32 34 00 00 00 00 00                         ; 824.....
```

5. DataFilter, DataComparator interface

- 인터페이스 설명
  - DataFilter : 데이터 중에 조건에 맞는 레코드만 골라내기 위해 DB안에 설정이 가능
    - public boolean filter(byte[] data) : 레코드가 순서대로 data로 넘어오는데 반환값이 true이면 record가 선택되고 false이면 record를 선택에서 제외시킨다.
    - DataFilterInteger class : wipi에서 만들어둔 Filter로 숫자범위로 필터링을 수행
  - DataComparator : 데이터의 배열순서를 정하는데 사용
    - public int compare(byte[] data1, byte[] data2) : 반환값으로 EQUIVALENT, FOLLOWS, PRECEDES중에 하나가 된다.
    - DataComparatorInteger class : 데이터를 숫자로 오름차순으로 배열
    - DataComparatorString class : string 순으로 배열
  - DataBase의 sortRecord(dataFilter, dataComparator)에서 database에 설정

- DataFilter구현예

```
public boolean filter(byte[] data){
    if(data[0] = 'A') return true;
    return false;
} // 레코드의 첫번째가 'A'인 것들만 필터링한다.
```

- DataComaparotor 구현 예

```
public int compare(byte[] data1, byte[] data2){
    if(data1[0] == data2[0]) return DataComparator.EQUIVALENT;
    else if(data1[0] > data2[0]) return DataComparator.FLLOWS;
    return DataComparator.PRECEDES;
} // 첫번째 바이트만 비교하여 오름차순으로 정렬
```