Bab 5
Persistence

MIDP menyediakan sebuah API dimana program dapat menyimpan data-data aplikasi secara lokal didalam device tersebut. MIDP Record Management System adalah sebuah fasilitas yang dimiliki oleh MIDlets untuk menyimpan data-data aplikasi pada saat MIDlet invocations. Data akan disimpan dalam non-volatile memory didalam device. Hal ini berarti, data-data program yang telah disimpan tidak akan hilang walaupun program di restart maupun device dimatikan.

5.1 Tujuan

Pada akhir pembelajaran, siswa diharapkan dapat:

- Memahami mengenai konsep dari Record Store
- Membuat dan membuka sebuah Record Store
- Menambah, memanggil kembali, mengupdate, dan mendelete record
- Memanggil record satu persatu (enumerate) record dengan menggunakan RecordEnumerate
- Membuat sebuah Record Comparator
- Membuat sebuah Record Filter

5.2 Record Store

Sebuah Record Store adalah sebuah koleksi daripada record-record. Record Id didalam Record Store selalu unique. Record Id akan secara otomatis dialokasikan pada saat pembentukan sebuah record dan bertindak sebagai index atau primary key. Pemberian record Id dilaksanakan secara sekuensial dan nilai yang diberikan kepada record Id pertama pada setiap Record Store adalah 1 (satu).

Pada saat sebuah record dihapus, record id-nya tidak akan bisa digunakan kembali. Jika kita membuat empat buah record dan menghapus record ke-empat, maka record Id selanjutnya yang akan diberikan oleh system adalah 5 (lihat gambar)

Record ID	Byte array
1	Data dari record #1
2	Data dari record #2
3	Data dari record #3
5	Data dari record #5

MIDlets dapat menciptakan lebih dari satu Record Store. Nama dari sebuah record store didalam MIDlet suite haruslah unique. Nama dari record store juga case sensitive dan memiliki panjang maksimal 32 karakter.

Pada saat MIDlet suite dihapus dari sebuah device, maka semua record store yang terkoneksi dengan MIDlet didalam suite tersebut juga akan terhapus.

Membuat dan membuka sebuah Record Store

Method-method dibawah ini digunakan untuk membuat dan membuka sebuah record store:

`static` RecordStore	openRecordStore`(String recordStoreName, boolean createIfNecessary)`
`static` RecordStore	openRecordStore`(String recordStoreName, boolean createIfNecessary, int authmode, boolean writable)`
`static` RecordStore	openRecordStore`(String recordStoreName, String vendorName, String suiteName)`

Jika createIfNecessary di-set menjadi true dan Record Store belum ada, maka Record Store akan dibangun. Jika createIfNecessary di-set menjadi false dan Record Store tersebut belum dibuat, maka sebuah RecordStoreNotFoundException akan dijalankan.

Authmode paramater dapat di-set menjadi RecordStore.AUTHMODE_PRIVATE atau RecordStore.AUTHMODE_ANY. Penggunaan AUTHMODE_PRIVATE akan menyebabkan Record Store hanya mampu diakses oleh MIDlet suite si pemilik MIDlet. Sedangkan setting authmode ke AUTHMODE_ANY akan menyebabkan Record Store untuk diakses oleh MIDlet manapun. Access mode ini dispesifikasikan oleh sebuah writable boolean parameter. Untuk memperbolehkan MIDlet yang lain (diluar MIDlet suite) untuk menggunakan record store tersebut, parameter ini harus diubah menjadi true.

Penggunaan bentuk pertama dari method openRecordStore()akan menyebabkan Record Store untuk dapat diakses oleh MIDlet-MIDlet didalam suite yang sama (authmode di-set ke AUTHMODE_PRIVATE).

Untuk membuka sebuah Record Store dari MIDlet suite yang berbeda, bentuk ketiga dari method openRecordStore harus digunakan. Anda harus menspesifikasikan nama vendor (vendorName) dan nama dari Midlet suite (suiteName).

Jika sebuah Record Store terlanjur dibuka, method ini akan mengembalikan reference kepada record store tersebut. System akan tetap menghitung berapa kali Record Store telah dibuka dan setiap Record Store harus ditutup dengan jumlah yang sama pada saat ia dibuka.

Menambahkan sebuah record

int	addRecord(byte[] data, int offset, int numBytes)

Method addRecord akan membuat record yang baru didalam Record Store dan akan mengembalikan record ID.

Mengambil kembali Record

byte[]	getRecord(int recordId)
int	getRecord(int recordId, byte[] buffer, int offset)
int	getRecordSize(int recordId)

Bentuk pertama dari method getRecord akan mengembalikan copy dari data stored yang ada didalam record tertentu berdasarkan RecordID. Bentuk kedua akan meng-copy data pada paramater byte array yang telah disediakan. Pada saat menggunakan bentuk kedua, byte array tersebut haruslah dialokasikan terlebih dahulu. Jika ukuran dari record lebih besar daripada ukuran dari parameter, maka akan terjadi ArrayIndexOutOfBoundsException. Anda akan menggunakan method getRecordSize secara berurutan untuk mengetahui ukuran dari record sebelum Anda mulai untuk membacanya.

Meng-update sebuah Record

Anda tidak dapat memodifikasi hanya sebagian dari data record. Jika Anda ingin untuk memodifikasi sebuah record Anda harus:
1. Membaca tiap record dengan menggunakan getRecord
2. Meng-update record didalam memory
3. Memanggil setRecord untuk mengupdate data record

void	setRecord(int recordId, byte[] newData, int offset, int numBytes)

Menghapus Record

void	deleteRecord(int recordId)

Pada saat sebuah record dihapus, record Id akan digunakan kembali di pemanggilan berikutnya pada addRecord. Hal ini berarti, ada sebuah celah didalam Record Id. Oleh karena itu, tidak disarankan untuk menggunakan counter increment untuk

membuat list dari keseluruhan record didalam record store. Anda harus menggunakan Record Enumerator untuk mengetahui tiap record didalam sebuah list store.

Menutup sebuah Record Store

void	closeRecordStore()

Record store yang akan ditutup dengan cara pemanggilan method closeRecordStore() tidak akan benar-benar ditutup sampai closeRecordStore() dipanggil sejumlah pemanggilan dari openRecordStore() sebelumnya. Pemanggilan closeRecordStore() lebih dari jumlah pemanggilan openRecordStore() akan berakibat exception RecordStoreNotOpen.

Potongan kode dari contoh RmsExample1 adalah MIDlet sederhana yang mendemonstrasikan bagaimana untuk membuat sebuah record store, menambah record, dan memanggil kembali semua record didalam record store:

```
            // Buka dan buatlah record store dengan nama "RmsExample1"
            recStore= RecordStore.openRecordStore("RmsExample1", true);

            // Masukkan content kedalam record store
            for(int recId=1; recId<=recStore.getNumRecords(); recId++){

        // getRecord memiliki return value berupa panjang dari record
                    recLength = recStore.getRecord(recId, data, 0);

                  // Mengkonversikan byte array menjadi String
                  String item = new String(data, 0, recLength);
                  ...
          }


            ...
            // Ini adalah String yang akan kita masukkan kedalam record
            String newItem = "Record #" + recStore.getNextRecordID();

            // Konversikan String ke byte array
            byte[] bytes = newItem.getBytes();

            // Tulislah record kedalam record store
            recStore.addRecord(bytes, 0, bytes.length);
```

Tips Pemrograman:

1. *Record ID dimulai dari 1, bukan 0. Oleh karena itu, apabila menggunakan loop, ingatlah untuk menggunakan 1 sebagai index pertama dan bukan 0.*
2. *Lebih baik digunakan Record Enumerator daripada menggunakan index incrementing (seperti contoh). Record yang telah dihapus, tetapi masih tetap ingin dibaca pada contoh disini akan menyebabkan InvalidRecordIDException.*

Mendapatkan list dari Record Store didalam MIDlet Suite

static String[]	listRecordStores()

Method ini akan mengembalikan array dari nama record store tersebut yang dimiliki oleh MIDlet suite. Jika MIDlet suite tidak memiliki sebuah Record Store, maka method ini akan memiliki nilai pengembalian null.

```
String[] storeNames = RecordStore.listRecordStores();
System.out.println("Record Stores for this MIDlet suite:");

for (int i=0; storeNames != null && i<storeNames.length; i++){
    System.out.println(storeNames[i]);
}
```

Contoh: RmsListStores

```
Record Stores for this MIDlet suite:
Prefs
RmsExample1
RmsExample2
```

Contoh output dari RmsListStores

Urutan penamaan yang akan dikembalikan tidak akan didefinisikan dan akan diimplementasikan secara independent. Oleh karena itu, apabila Anda ingin untuk menampilkan nama tersebut secara alphabetic, maka Anda harus melakukan sorting array terlebih dahulu.

Menyimpan Data Primitif Java

Sejauh ini, data yang telah dibuat dan dibaca dari Record Store adalah berupa String. CLDC memiliki standard classes dalam manipulasi data primitif. Class tersebut berasal dari standard library platform Java 2, yaitu Standard Edition (J2SE).

Anda dapat menulis data Java primitif dengan mengkombinasikan class ByteArrayOutputStream dan DataOutputStream. Pembacaan tipe data primitive(int, long, short, string, Boolean, dan sebagainya) dapat pula dilakukan dengan menggunakan ByteArrayInputStream dan DataInputStream.

```
ByteArrayOutputStream out = new ByteArrayOutputStream();
DataOutputStream dOut = new DataOutputStream(out);
// Menyimpan sebuah integer
dOut.writeInt(recStore.getNextRecordID() * recStore.getNextRecordID());
// Menyimpan sebuah string
dOut.writeUTF("Record #" + recStore.getNextRecordID());
byte[] bytes = out.toByteArray();
// Menuliskan Record pada Store
recStore.addRecord(bytes, 0, bytes.length);
...
```

```
// Menuju Record selanjutnya
byte[] recBytes = enumerator.nextRecord();
ByteArrayInputStream in = new ByteArrayInputStream(recBytes);
DataInputStream dIn = new DataInputStream(in);
int count = dIn.readInt();
String item = dIn.readUTF();
```

Method lain untuk Record Stores

```
long getLastModified()
 int getVersion()
```

Sistem merekam bilamana sebuah Record Store mengalami modifikasi terakhir. Method getLastModified memberikan informasi bahwa sebuah Record Store mengalami perubahan terakhir, dalam bentuk long dan sesuai format yang digunakan oleh System.currentTimeMillis().

Seluruh Record Store memiliki version information. Setiap kali sebuah record mengalami modifikasi, maka version number juga akan terupdate. Penggunaan method addRecord, setRecord dan deleteRecord menyebabkan penambahan version number dari record store tersebut.

```
static void deleteRecordStore(String recordStoreName)
                    Menghapus record store.
     String getName()
                    Mengetahui nama dari RecordStore.
        int getNextRecordID()
                    Mengetahui recordId dari record selanjutnya untuk
            disimpan pada record store.
        int getNumRecords()
                    Mendapatkan jumlah record yang terdapat pada
            Record Store.
        int getSize()
                    Mengetahui space (dalam bytes) yang dipakai oleh
            record store.
        int getSizeAvailable()
                    Mengetahui sisa space yang tersedia (dalam
            bytes).
       void setMode(int authmode,          boolean writable)
                    Mengubah access mode dari RecordStore.
```

5.3 Record Enumeration

Memeriksa sebuah record store menggunakan inceremented index adalah tidak efisien. Record stores yang telah dihapus akan terlewati jika Record Id dari record tersebut tidak digunakan kembali.

Penggunaan record enumeration dapat menyelesaikan permasalahan tersebut dengan melakukan pemeriksaaan pada record yang telah dihapus. Anda juga dapat mengurutkan enumerasi dengan menggunakan method pembanding. Dengan penggunaan method pembanding, anda dapat melewati record yang tidak diharapkan pada output.

```
RecordEnumeration enumerateRecords(RecordFilter filter,
                  RecordComparator comparator, boolean keepUpdated)
```

Method enumerateRecords dari sebuah record store akan menghasilkan enumerasi untuk memeriksa seluruh record pada sebuah record store. Ini adalah cara yang direkomendasikan untuk melewatkan seluruh record dalam record store. Filter dan Comparator akan dibahas dalam pembahasan selanjutnya.

Cara paling sederhana dalam menggunakan method ini adalah memberikan nilai null untuk filter dan comparator. Hal ini akan menghasilkan enumerasi dari seluruh record pada sebuah store dalam urutan acak.

```
// Membuka atau membuat sebuah record store dengan nama "RmsExample2"
recStore = RecordStore.openRecordStore("RmsExample2", true);

// Mengambil isi dari record store
RecordEnumeration enumerator
       = recStore.enumerateRecords(null, null, false);
while (enumerator.hasNextElement()){
       // Mendapatkan record selanjutnya dan konversi byte array menjadi string
       String item = new String(enumerator.nextRecord());
       // Area kode manipulasi record
       ...
}
```

5.4 Record Comparator

Pengurutan sebuah enumerasi dapat didefinisikan menggunakan sebuah Record Comparator. Record Comparator digunakan pada method enumerateRecords. Jika anda ingin mengurutkan output dari enumerasi, anda harus membuat comparator dan mengimplementasikannya sebagai parameter kedua pada enumerateRecords.

```
int compare(byte[] rec1, byte[] rec2)
```

Untuk membuat sebuah record comparator, anda harus mengimplementasikan interface RecordComparator. Interface tersebut mendefinisikan method tunggal, compare(byte[] rec1, byte[] rec2). Method ini harus menghasilkan return value, RecordComparator.FOLLOWS atau RecordComparator.PRECEDES.

RecordComparator.EQUIVALENT harus dihasilkan jika rec1 adalah ekuivalen terhadap rec2 dalam pengurutan.

```
// Membuat enumerasi, diurutkan menurut alfabet
RecordEnumeration enumerator
       = recStore.enumerateRecords(null, new AlphaOrder(), false);
       ...

// Pengurutan menurut alfabet
class AlphaOrder implements RecordComparator {
       public int compare(byte[] rec1, byte[] rec2){
              String record1 = new String(rec1).toUpperCase();
              String record2 = new String(rec2).toUpperCase();
              if (record1.compareTo(record2) < 0){
                     return(PRECEDES);
              } else {
                     if (record1.compareTo(record2) > 0){
                            return(FOLLOWS);
                     } else {
                            return(EQUIVALENT);
                     }
              }
       }
}
```

5.5 Record Filter

Contoh – contoh selama ini membaca seluruh record dari sebuah store. Kita dapat menggunakan sebuah filter untuk mendapatkan hanya record yang kita inginkan.

Program Anda harus mengimpelementasikan method match() untuk menyeleksi record. Method tersebut akan menghasilkan nilai true jika record sesuai dengan criteria.

```
 boolean matches(byte[] candidate)
```

```
public boolean matches(byte[] candidate){
       boolean isaMatch = false;
       try {
              ByteArrayInputStream bin = new ByteArrayInputStream(candidate);
              DataInputStream dIn = new DataInputStream(bin);
```

```
                int count = dIn.readInt();
                String item = dIn.readUTF();
                // mendapatkan record dengan akhiran "0"
                if (item.endsWith("0")){
                        isaMatch = true;
                } else {
                        isaMatch = false;
                }
        } catch (Exception e){items.append(e.toString(), null); }
        return(isaMatch);
}
```

5.6 Record Listener

Sebuah Record Store dapat menggunakan lebih dari satu record listener. Record listener adalah object yang dipanggil pada saat sebuah record ditambahkan, diubah atau dihapus dari record store. Record listeners harus mengimplementasikan interface RecordListener.

Record Listener diregristrasikan pada record store menggunakan method addRecordListener(). Pada saat sebuah record store ditutup, seluruh record listener yang terkait juga akan dihapus.

Penggunaan method deleteRecordStore() tidak akan menghasilkan pemanggilan recordDeleted() pada record listener manapun yang terkait.

void	**recordAdded**(RecordStore recordStore, int recordId) Dipanggil saat sebuah record ditambahkan pada record store.
void	**recordChanged**(RecordStore recordStore, int recordId) Dipanggil setelah sebuah record pada record store diubah.
void	**recordDeleted**(RecordStore recordStore, int recordId) Dipanggil setelah sebuah record dihapus dari record store.

5.7 Latihan

5.7.1 Penyimpanan Pilihan

Buat sebuah class yang dapat melangsungkan pemilihan pada program. Class tersebut akan menyimpan pilihan pada sebuah Record Store. Setiap record akan memiliki variabel name dan value. Setiap pasangan variabel disimpan pada sebuah record. Name dan value disimpan pada database sebagai string.

Class Anda harus mengimplementasikan method sebagai berikut :

```
public String readVar(RecordStore recStore, String name, String defaultValue){
public void writeString(RecordStore recStore, String name, String value);
```