1.Tujuan

- 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

2. Latar Belakang

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.

3. Percobaan

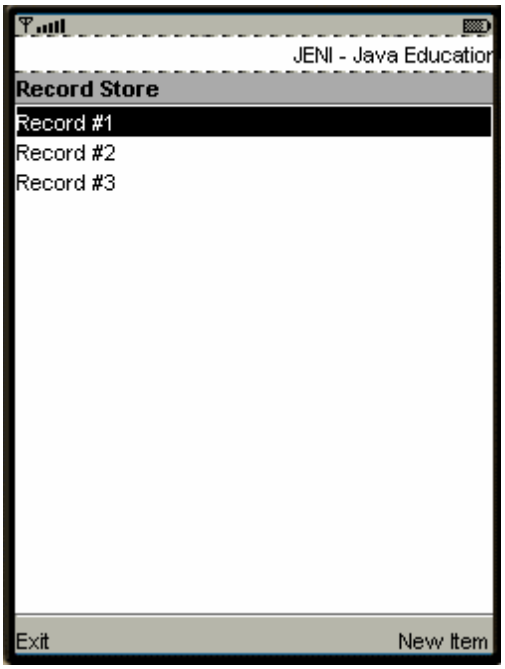
Percobaan 1: Menambah Item

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;

public class RMSExample extends MIDlet implements CommandListener {
      Display display;
      List recList;
      RecordStore recStore;
      byte[] data = null;
      Command exitCommand = new Command("Exit", Command.EXIT, 1);
      Command newCommand = new Command("New Item", Command.OK, 1);
      Ticker ticker = new Ticker(
            "JENI - Java Education Network Indonesia");

      public RMSExample(){
            recList = new List("Record Store",List.IMPLICIT);
            dispRec();
```

```
        recList.setTicker(ticker);
        recList.addCommand(exitCommand);
        recList.addCommand(newCommand);
        recList.setCommandListener(this);
}

public void startApp() {
        if (display == null){
                display = Display.getDisplay(this);
                display.setCurrent(recList);
        }
}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d){
        if (c == exitCommand){
                destroyApp(true);
                notifyDestroyed(); // Exit
        }
        if (c == newCommand){
                try{
                        // Buka dan buatlah record store dengan nama “RmsExample1”
                        recStore= RecordStore.openRecordStore("RmsExample1", true);

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

                        // Konversikan String ke byte array
                        data = newItem.getBytes();

                        // Tulislah record kedalam record store
                        recStore.addRecord(data, 0, data.length);
                        recStore.closeRecordStore();
                }catch(Exception e){
                        System.out.println(e.toString());
                }
                dispRec();
        }
}
```

```
	public void dispRec(){
		recList.deleteAll();
		String[] data = getRecordList();
		if(data.length>0){
			for(int i=0;i<data.length;i++)
				recList.append(data[i],null);
		}
	}

	public String[] getRecordList(){
		try{
			// Buka dan buatlah record store dengan nama “RmsExample1”
			recStore= RecordStore.openRecordStore("RmsExample1", true);
			// Masukkan content kedalam record store

			String[] dataList = new String[recStore.getNumRecords()];

		if (recStore.getNumRecords()>0){
				for(int recId=1; recId<=recStore.getNumRecords(); recId++){
					int size = recStore.getRecordSize( recId );
					if( data == null || data.length < size ){
						data = new byte[ size + 20 ];
					}

					// getRecord memiliki return value berupa panjang
dari record
					int recLength = recStore.getRecord(recId,data,0);
					// Mengkonversikan byte array menjadi String
					dataList[recId-1] = new String(data, 0, recLength);
				}
			}
			recStore.closeRecordStore();
			return dataList;
		}catch (Exception e){
			return null;
		}

	}
}
```

Hasil :



Percobaan 2: Membaca Record Store

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;
public class RmsList extends MIDlet implements CommandListener {
	Display display;
	List recList;
	RecordStore recStore;
	byte[] data = null;
	Command exitCommand = new Command("Exit", Command.EXIT, 1);
	Ticker ticker = new Ticker(
		"JENI - Java Education Network Indonesia");
	public RmsList(){
		recList = new List("Record Store List",List.IMPLICIT);
		dispList();
		recList.setTicker(ticker);
		recList.addCommand(exitCommand);
		recList.setCommandListener(this);
	}
```

```
	public void startApp() {
		if (display == null){
			display = Display.getDisplay(this);
			display.setCurrent(recList);
		}
	}

	public void pauseApp() {
	}
	public void destroyApp(boolean unconditional) {
	}
	public void commandAction(Command c, Displayable d){
		if (c == exitCommand){
			destroyApp(true);
			notifyDestroyed(); // Exit
		}
	}
	public void dispList(){
		recList.deleteAll();
		try{
			String[] data = recStore.listRecordStores();
			if(data.length>0){
				for(int i=0;i<data.length;i++)
					recList.append(data[i],null);
			}
		} catch (Exception e){
		}
	}
}
```

Hasil :

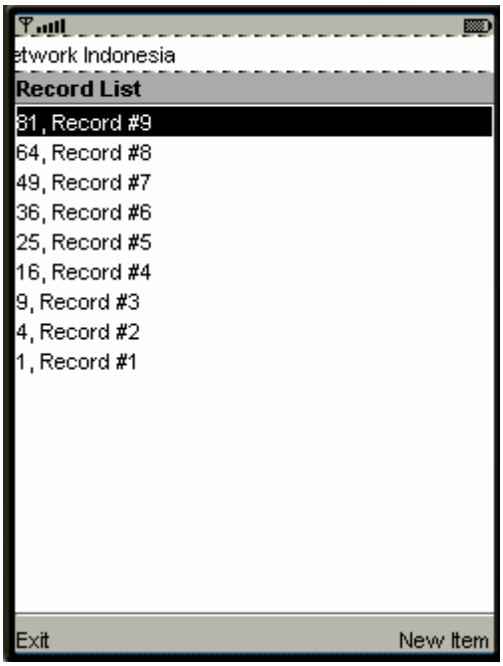
Percobaan 3: Penggunaan Enumerator

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;
import java.io.*;

public class RmsExample2 extends MIDlet implements CommandListener {
	Display display;
	List recList;
	RecordStore recStore;
	byte[] data = null;
	Command exitCommand = new Command("Exit", Command.EXIT, 1);
	Command newCommand = new Command("New Item", Command.OK, 1);
	Ticker ticker = new Ticker(
		"JENI - Java Education Network Indonesia");

	public RmsExample2(){
		recList = new List("Record List",List.IMPLICIT);
```

```
            dispRec();
            recList.setTicker(ticker);
            recList.addCommand(exitCommand);
            recList.addCommand(newCommand);
            recList.setCommandListener(this);
      }

      public void startApp() {
            if (display == null){
                  display = Display.getDisplay(this);
                  display.setCurrent(recList);
            }
      }
      public void pauseApp() {
      }
      public void destroyApp(boolean unconditional) {
      }
      public void commandAction(Command c, Displayable d){
            if (c == exitCommand){
                  destroyApp(true);
                  notifyDestroyed(); // Exit
            }
            if (c == newCommand){
                  try{
                        // Buka dan atau buatlah record store dengan nama “RmsExample2”
                        recStore= RecordStore.openRecordStore("RmsExample2", true);

                        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);
                        dOut.close();
                        out.close();
                        recStore.closeRecordStore();
                  }catch(Exception e){
                        System.out.println(e.toString());
                  }
                  dispRec();
            }
      }
      public void dispRec(){
            recList.deleteAll();
            try{
            // 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()){
                          // 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();
                          recList.append(count+", "+item,null);
                          dIn.close();
                          in.close();
                    }
                    recStore.closeRecordStore();
              }catch (Exception e){
              }
        }
}
```

Hasil :

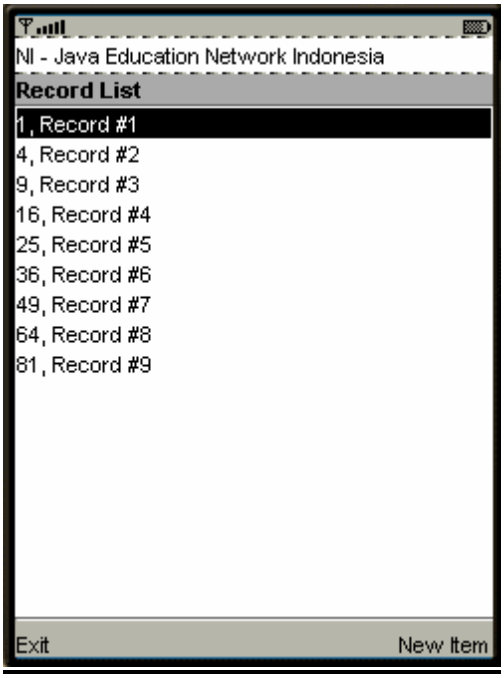


Percobaan 4: Penggunaan Record Comparator

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;
import java.io.*;
public class RmsComparator extends MIDlet implements CommandListener,RecordComparator {
	Display display;
	List recList;
	RecordStore recStore;
	byte[] data = null;
	Command exitCommand = new Command("Exit", Command.EXIT, 1);
	Command newCommand = new Command("New Item", Command.OK, 1);
	Ticker ticker = new Ticker(
		"JENI - Java Education Network Indonesia");
	public RmsComparator(){
		recList = new List("Record List",List.IMPLICIT);
		dispRec();
		recList.setTicker(ticker);
		recList.addCommand(exitCommand);
		recList.addCommand(newCommand);
		recList.setCommandListener(this);
	}
	public void startApp() {
		if (display == null){
			display = Display.getDisplay(this);
			display.setCurrent(recList);
		}
	}
	public void pauseApp() {
	}
	public void destroyApp(boolean unconditional) {
	}
	public void commandAction(Command c, Displayable d){
		if (c == exitCommand){
			destroyApp(true);
			notifyDestroyed(); // Exit
		}
		if (c == newCommand){
			try{
				// Buka dan atau buatlah record store dengan nama “RmsComparator”
				recStore= RecordStore.openRecordStore("RmsComparator", true);

				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);
                        dOut.close();
                        out.close();
                        recStore.closeRecordStore();
                  }catch(Exception e){
                        System.out.println(e.toString());
                  }
                  dispRec();
            }
      }
      public void dispRec(){
            recList.deleteAll();
            try{
            // Membuka atau membuat sebuah record store dengan nama "RmsComparator"
                  recStore = RecordStore.openRecordStore("RmsComparator", true);
                  // Mengambil isi dari record store
                  RecordEnumeration enumerator
                        = recStore.enumerateRecords(null, this, false);
                  while (enumerator.hasNextElement()){
                        // 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();
                        recList.append(count+", "+item,null);
                        dIn.close();
                        in.close();
                  }
                  recStore.closeRecordStore();
            }catch (Exception e){
            }
      }
      public int compare(byte[] rec1, byte[] rec2){
            String record1 = new String(rec1).toUpperCase();
            String record2 = new String(rec2).toUpperCase();
            //Sorting Ascending
            if (record1.compareTo(record2) < 0){
                  return(PRECEDES);
            } else {
                  if (record1.compareTo(record2) > 0){
                        return(FOLLOWS);
                  } else {
                        return(EQUIVALENT);
                  }
            }
      } }
```

Hasil :



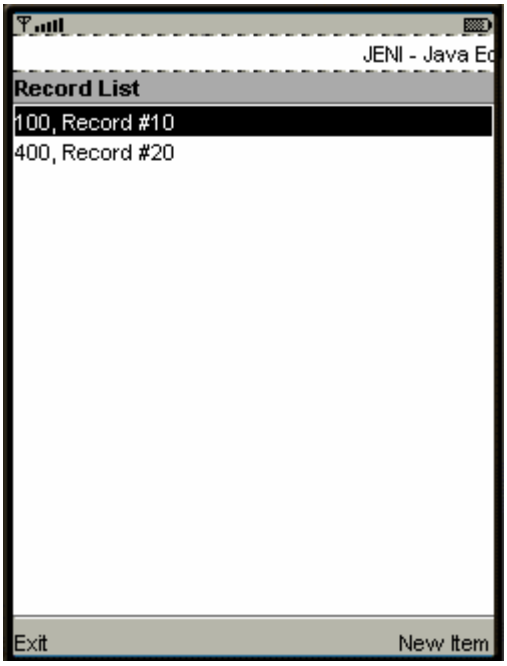
Percobaan 5: Penggunaan Record Filter

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;
import java.io.*;

public class RmsFilter extends MIDlet implements
            CommandListener,RecordComparator,RecordFilter {
       Display display;
       List recList;
       RecordStore recStore;
       byte[] data = null;
       Command exitCommand = new Command("Exit", Command.EXIT, 1);
       Command newCommand = new Command("New Item", Command.OK, 1);
       Ticker ticker = new Ticker(
              "JENI - Java Education Network Indonesia");
       public RmsFilter(){
```

```
            recList = new List("Record List",List.IMPLICIT);
            dispRec();
            recList.setTicker(ticker);
            recList.addCommand(exitCommand);
            recList.addCommand(newCommand);
            recList.setCommandListener(this);
      }
      public void startApp() {
            if (display == null){
                  display = Display.getDisplay(this);
                  display.setCurrent(recList);
            }
      }
      public void pauseApp() {
      }
      public void destroyApp(boolean unconditional) {
      }
      public void commandAction(Command c, Displayable d){
            if (c == exitCommand){
                  destroyApp(true);
                  notifyDestroyed(); // Exit
            }
            if (c == newCommand){
                  try{
                        // Buka dan atau buatlah record store dengan nama “RmsFilter”
                        recStore= RecordStore.openRecordStore("RmsFilter", true);

                        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);
                        dOut.close();
                        out.close();
                        recStore.closeRecordStore();
                  }catch(Exception e){
                        System.out.println(e.toString());
                  }
                  dispRec();
            }
      }
      public void dispRec(){
            recList.deleteAll();
            try{
                  // Membuka atau membuat sebuah record store dengan nama "RmsFilter"
```

```
                    recStore = RecordStore.openRecordStore("RmsFilter", true);
                    // Mengambil isi dari record store
                    RecordEnumeration enumerator
                          = recStore.enumerateRecords(this, this, false);
                    while (enumerator.hasNextElement()){
                          // 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();
                          recList.append(count+", "+item,null);
                          dIn.close();
                          in.close();
                    }
                    recStore.closeRecordStore();
             }catch (Exception e){
             }
      }
      public int compare(byte[] rec1, byte[] rec2){
             String record1 = new String(rec1).toUpperCase();
             String record2 = new String(rec2).toUpperCase();
             //Sorting Ascending
             if (record1.compareTo(record2) < 0){
                    return(PRECEDES);
             } else {
                    if (record1.compareTo(record2) > 0){
                          return(FOLLOWS);
                    } else {
                          return(EQUIVALENT);
                    }
             }
      }
      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){recList.append(e.toString(), null); }
             return(isaMatch);
      }
}
```

Hasil :



4. 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);
```

Jawaban:

Bkn midlet

```
/*
 * RmsPrefs.java
 *
 * (c) 2005 J.E.D.I.
 */

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;
import java.io.*;

public class RmsPrefs implements RecordFilter {
    String searchName = null;

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

            String item = dIn.readUTF();
            if (searchName != null){
                if (searchName.indexOf(item) != -1){
                    isaMatch = true;
                }
            }

            dIn.close();
            bIn.close();
        } catch (Exception e){}

        return(isaMatch);
    }

    public String readVar(RecordStore recStore, String name, String defaultValue){
        String value = defaultValue;

        searchName = name;
        try {
            // Load contents of Record Store
```

```
            RecordEnumeration enumerator = recStore.enumerateRecords(this, null,
false);

            // get only the first matching record
            if (enumerator.hasNextElement()){
                // Get the next record
                byte[] recBytes = enumerator.nextRecord();

                ByteArrayInputStream in = new ByteArrayInputStream(recBytes);
                DataInputStream dIn = new DataInputStream(in);
                String sname = dIn.readUTF();
                value = dIn.readUTF();

                dIn.close();
                in.close();
            }
        } catch (Exception e){}
        return(value);
    }

    public void writeString(RecordStore recStore, String name, String value){
        try {
            ByteArrayOutputStream out = new ByteArrayOutputStream();
            DataOutputStream dOut = new DataOutputStream(out);

            // Store the name
            dOut.writeUTF(name);

           // Store the value
           dOut.writeUTF(value);

            byte[] bytes = out.toByteArray();

            // Write the Record into the Store
            recStore.addRecord(bytes, 0, bytes.length);

            dOut.close();
            out.close();
        } catch (Exception e){}
    }
}
```