WEEK 4 - DEMO CODE

Basic RMS, RecordFilter and RecordComparator

** @author George Nguyen <George.Nguyen@rmit.edu.vn>

```
public class RMSDemo extends MIDlet implements RecordListener, RecordFilter {

    private RecordStore rms;
    private byte bytes[];
    private String names[] = {"George Nguyen", "Kevin Jackson", "Barend Scholtus", "Thanh Nguyen", "Quang Tran", "Stanley
Luong"};
    private String name, filterString;

    protected void startApp() throws MIDletStateChangeException {
        try {
            rms = RecordStore.openRecordStore("My Record", true);

            //Register listener for a RecordStore
            rms.addRecordListener(this);

            for (int i = 0; i < names.length; i++) {
                bytes = names[i].getBytes();
                rms.addRecord(bytes, 0, bytes.length);
            }

            System.out.println("-----------------------------------------------");
            System.out.println("Normal RecordStore without Filter & Comparator");
            //Enumerate Record Store
            RecordEnumeration token = rms.enumerateRecords(null, null, true);
            while (token.hasNextElement()) {
                name = new String(token.nextRecord());
                System.out.println("Name: " + name);
            }

            //Set filter criteria & filter RecordStore
            System.out.println("-----------------------------------------------");
            System.out.println("RecordStore with Filter");
            filterString = "Nguyen";

            token = rms.enumerateRecords(this, null, true);

            while (token.hasNextElement()) {
                name = new String(token.nextRecord());
                System.out.println("Name: " + name);
            }
```

```
        //Create new Comparator instance& filter RecordStore
        System.out.println("-----------------------------------------------");
        System.out.println("RecordStore with Comparator");
        CompareLastName lnSort=new CompareLastName();

        token = rms.enumerateRecords(null, lnSort, true);

        while (token.hasNextElement()) {
            name = new String(token.nextRecord());
            System.out.println("Name: " + name);
        }


    } catch (RecordStoreException ex) {
        System.out.println("Cannot create/open record");
    } finally {
        try {
            rms.closeRecordStore();
        } catch (RecordStoreException ex) {
            System.out.println("Cannot close record");
        }
    }


}

protected void pauseApp() {
}

protected void destroyApp(boolean unconditional) throws MIDletStateChangeException {
}

public void recordAdded(RecordStore recordStore, int recordId) {
    try {
        System.out.println(new String(rms.getRecord(recordId)) + " is added");
    } catch (RecordStoreException ex) {
        ex.printStackTrace();
    }
}

public void recordChanged(RecordStore recordStore, int recordId) {
    System.out.println("A record has been updated in My Record");
}

public void recordDeleted(RecordStore recordStore, int recordId) {
    System.out.println("A record has been deleted from My Record");
}

public boolean matches(byte[] bytes) {
```

```
        String candidate = new String(bytes);
        if (candidate.indexOf(filterString) > 0) {
            return true;
        }
        return false;
    }
}

class CompareLastName implements RecordComparator {

    public int compare(byte[] bytes, byte[] bytes1) {
        String str1 = new String(bytes);
        String str2 = new String(bytes1);

        //extract lastname
        str1 = str1.substring(str1.indexOf(" ") + 1);
        str2 = str2.substring(str2.indexOf(" ") + 1);
        int result = str1.compareTo(str2);
        if (result == 0) {
            return RecordComparator.EQUIVALENT;
        } else if (result < 0) {
            return RecordComparator.PRECEDES;
        } else {
            return RecordComparator.FOLLOWS;
        }
    }
}
```