**LAB : 7**

**Objective :**

1. Create a MIDP application, where the user can enter player name and points. The program saves the information to the record using RMS at MIDP device. Program should also print out the top 10 player list to the end user. You can use this class in your game if you made own class for saving and reading record sets.
2. Create a slide show which has three slides, which includes pictures at PNG format. Program should change to the new slide other 5 seconds.
3. Write a sample program to show how to make a SOCKET Connection from j2me phone.
4. Create a MIDP application, which show to the user 5-10 quiz questions. All questions have 4 possible options and one right option exactly. Application counts and shows to the user how many right answers were right and shows them to user.

**Requrements :**

(a) Windows PC (Windows 7/8/10) / Mac

(b) JDK 1.5

(c) Java Wireless Toolkit 2.5.2

**Implementation :**

1. Create a MIDP application, where the user can ... record sets.

**PlayerList.java :**

import javax.microedition.midlet.\*;

import javax.microedition.lcdui.\*;

import javax.microedition.rms.\*;

import java.io.\*;

public class PlayerList extends MIDlet implements CommandListener {

public Form form1;

public Form form2;

public Form form3;

public Form form4;

public Form form5;

public Form form6;

public Form form7;

public ChoiceGroup ch1;

public ChoiceGroup ch2;

public ChoiceGroup ch3;

;

public ChoiceGroup ch4;

;

public ChoiceGroup ch5;

;

public Command nextCommand;

public Command backCommand;

public Command MenuCommand;

public Command OkCommand;

public Command ExitCommand;

public Command sCommand;

public Display display;

public StringItem st;

public TextField textfield;

public int count;

public RecordStore recordstore = null;

public RecordEnumeration re = null;

public Alert alert;

public Compare comp;

public StringItem st1;

public PlayerList() {

count = 0;

display = Display.getDisplay(this);

nextCommand = new Command("Next", Command.OK, 1);

backCommand = new Command("Back", Command.BACK, 1);

OkCommand = new Command("Save", Command.SCREEN, 1);

ExitCommand = new Command("Exit", Command.SCREEN, 1);

sCommand = new Command("TopScores", Command.SCREEN, 1);

st = new StringItem("TotalPoints", "0");

st1 = new StringItem("", "");

textfield = new TextField("EnterName", "", 20, TextField.ANY);

form1 = new Form("J2ME Stands for");

form2 = new Form("a+b=");

form3 = new Form("5\*5");

form4 = new Form("Who is AP CM");

form5 = new Form("How many Districts in AP");

form6 = new Form("Score");

form7 = new Form("Top Scoreers");

ch1 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch1.append("Java 2 Mobile Edition", null);

ch1.append("Java 2 Macro Edition", null);

ch1.append("Java 2 Micro Edition", null);

ch1.append("Java 2 Music Edition", null);

form1.append(ch1);

form1.addCommand(nextCommand);

form1.setCommandListener(this);

ch2 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch2.append("b+a", null);

ch2.append("b\*a", null);

ch2.append("b/a", null);

ch2.append("b-a", null);

form2.append(ch2);

form2.addCommand(nextCommand);

form2.addCommand(backCommand);

form2.setCommandListener(this);

ch3 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch3.append("20", null);

ch3.append("30", null);

ch3.append("10", null);

ch3.append("25", null);

form3.append(ch3);

form3.addCommand(nextCommand);

form3.addCommand(backCommand);

form3.setCommandListener(this);

ch4 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch4.append("Rose", null);

ch4.append("Jagan", null);

ch4.append("Chandani", null);

ch4.append("Kiran", null);

form4.append(ch4);

form4.addCommand(nextCommand);

form4.addCommand(backCommand);

form4.setCommandListener(this);

ch5 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch5.append("8", null);

ch5.append("4", null);

ch5.append("11", null);

ch5.append("23", null);

form5.append(ch5);

form5.addCommand(backCommand);

form5.addCommand(nextCommand);

form5.setCommandListener(this);

form6.addCommand(OkCommand);

form6.addCommand(ExitCommand);

form6.addCommand(sCommand);

form6.setCommandListener(this);

form7.addCommand(backCommand);

form7.setCommandListener(this);

try {

recordstore = RecordStore.openRecordStore("Quiz", true);

} catch (Exception ex) { }

}

public void startApp() {

display.setCurrent(form1);

}

public void pauseApp() { }

public void destroyApp(boolean unconditional) { }

public void commandAction(Command cmd, Displayable displayable) {

if (displayable == form1) {

if (cmd == nextCommand) display.setCurrent(form2);

} else if (displayable == form2) {

if (cmd == nextCommand) display.setCurrent(form3);

else if (cmd == backCommand) display.setCurrent(form1);

} else if (displayable == form3) {

if (cmd == nextCommand) display.setCurrent(form4);

else if (cmd == backCommand) display.setCurrent(form2);

} else if (displayable == form4) {

if (cmd == nextCommand)

display.setCurrent(form5);

else if (cmd == backCommand) display.setCurrent(form3);

} else if (displayable == form5) {

if (cmd == nextCommand) {

if (ch1.getSelectedIndex() == 2) count++;

if (ch2.getSelectedIndex() == 0) count++;

if (ch3.getSelectedIndex() == 3) count++;

if (ch4.getSelectedIndex() == 3) count++;

if (ch5.getSelectedIndex() == 3) count++;

st.setText(String.valueOf(count));

form6.append(st);

form6.append(textfield);

display.setCurrent(form6);

}

else if (cmd == backCommand) display.setCurrent(form4);

} else if (displayable == form6) {

if (cmd == OkCommand) {

try {

String Pname = textfield.getString();

int Points = Integer.parseInt(st.getText());

byte[] bytes;

ByteArrayOutputStream ostream = new ByteArrayOutputStream();

DataOutputStream dstream = new DataOutputStream(ostream);

dstream.writeUTF(Pname);

dstream.writeInt(Points);

dstream.flush();

bytes = ostream.toByteArray();

recordstore.addRecord(bytes, 0, bytes.length );

ostream.reset();

ostream.close();

dstream.close();

alert = new Alert("Message", "Saved", null, AlertType.INFO);

alert.setTimeout(Alert.FOREVER);

display.setCurrent(alert);

} catch (Exception ex) {

alert = new Alert("Message", ex.toString(), null, AlertType.INFO);

alert.setTimeout(Alert.FOREVER);

display.setCurrent(alert);

}

} else if (cmd == ExitCommand) {

try {

recordstore.closeRecordStore();

notifyDestroyed();

} catch (Exception ex) { }

} else if (cmd == sCommand) {

try {

byte[] bytes = new byte[300];

ByteArrayInputStream bstream = new ByteArrayInputStream(bytes);

DataInputStream dstream = new DataInputStream(bstream);

StringBuffer sb = new StringBuffer();

comp = new Compare();

re = recordstore.enumerateRecords(null, comp, false);

st1.setText("");

while (re.hasNextElement()) {

recordstore.getRecord(re.nextRecordId(), bytes, 0);

sb.append(dstream.readUTF() + "|" + dstream.readInt());

sb.append("\n");

dstream.reset();

} bstream.close();

dstream.close();

st1.setText(sb.toString());

form7.append(st1);

}

catch (Exception ex) {

alert = new Alert("Msg", ex.toString(), null, AlertType.INFO);

alert.setTimeout(Alert.FOREVER);

display.setCurrent(alert);

} display.setCurrent(form7);

}

} else if (displayable == form7) {

if (cmd == backCommand) display.setCurrent(form6);

}

}

}

class Compare implements RecordComparator {

public byte[] bytedata = new byte[300];

public ByteArrayInputStream bstream = null;

public DataInputStream dstream = null;

public int compare(byte[] r1, byte[] r2) {

int r1int, r2int;

int or = 0;

try {

int maxlen = Math.max(r1.length, r2.length);

if (maxlen > bytedata.length) {

bytedata = new byte[maxlen];

} bstream = new ByteArrayInputStream(r1);

dstream = new DataInputStream(bstream);

dstream.readUTF();

r1int = dstream.readInt();

bstream = new ByteArrayInputStream(r2);

dstream = new DataInputStream(bstream);

dstream.readUTF();

r2int = dstream.readInt();

if (r1int == r2int) {

or = RecordComparator.EQUIVALENT;

} else if (r1int > r2int) {

or = RecordComparator.PRECEDES;

} else if (r1int < r2int) {

or = RecordComparator.FOLLOWS;

} return or;

} catch (Exception ex) {

return RecordComparator.EQUIVALENT;

}

} public void compareClose() {

try {

if (bstream != null) {

bstream.close();

} if (dstream != null) {

dstream.close();

}

} catch (Exception ex) { }

}

}

**Output :**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

1. Create a slide show which has three slides, which includes pictures at PNG format. Program should change to the new slide other 5 seconds.

**SlideShow.java :**

import java.io.\*;

import javax.microedition.lcdui.\*;

import javax.microedition.midlet.\*;

public class SlideShow extends MIDlet implements CommandListener {

public Form slide1;

public Form slide2;

public Form slide3;

public Image image1;

public Image image2;

public Image image3;

public ImageItem imageitem1;

public ImageItem imageitem2;

public ImageItem imageitem3;

Display display = null;

Command exit = new Command("Exit", Command.EXIT, 1);

public SlideShow() {

display = Display.getDisplay(this);

try {

image1 = Image.createImage("/SlideS1.png");

image2 = Image.createImage("/SlideS2.jpg");

image3 = Image.createImage("/SlideS3.png");

imageitem1 = new ImageItem(null, image1, ImageItem.LAYOUT\_CENTER, "image1");

imageitem2 = new ImageItem(null, image2, ImageItem.LAYOUT\_CENTER, "image2");

imageitem3 = new ImageItem(null, image3, ImageItem.LAYOUT\_CENTER, "image3");

} catch (Exception e) {}

slide1 = new Form("Slide1");

slide2 = new Form("Slide2");

slide3 = new Form("Slide3");

slide1.append(imageitem1);

slide1.addCommand(exit);

slide2.append(imageitem2);

slide2.addCommand(exit);

slide3.append(imageitem3);

slide3.addCommand(exit);

slide1.setCommandListener(this);

slide2.setCommandListener(this);

slide3.setCommandListener(this);

}

public void startApp() {

Thread runner = new Thread(new ThreadRunner(display, slide1, slide2, slide3)); runner.start();

}

public void pauseApp() {

}

public void destroyApp(boolean unconditional) {

}

public void commandAction(Command command, Displayable displayable) {

if (displayable == slide1) {

if (command == exit)

notifyDestroyed();

} else if (displayable == slide2) {

if (command == exit)

notifyDestroyed();

} else if (displayable == slide3) {

if (command == exit)

notifyDestroyed();

}

}

}

class ThreadRunner implements Runnable {

Display display;

public int c = 0;

public Form slide1;

public Form slide2;

public Form slide3;

public ThreadRunner(Display display, Form slide1, Form slide2, Form slide3) {

this.display = display;

this.slide1 = slide1;

this.slide2 = slide2;

this.slide3 = slide3;

}

public void run() {

while (true) {

c++;

if (c == 1)

display.setCurrent(slide1); else if (c == 2)

display.setCurrent(slide2); else if (c == 3)

display.setCurrent(slide3); else if (c == 4)

c = 0;

try {

Thread.sleep(1500);

} catch (Exception ex) {}

}

}

}

**Output :**

|  |  |
| --- | --- |
|  |  |
|  |  |

1. Write a sample program to show how to make a SOCKET Connection from j2me phone.

**SocketConnection.java :**

import javax.microedition.midlet.\*;

import javax.microedition.io.\*;

import javax.microedition.lcdui.\*;

import java.io.\*;

public class SocketConnection extends MIDlet {

//

private StreamConnection streamConnection = null;

private OutputStream outputStream = null;

private DataOutputStream dataOutputStream = null;

private InputStream inputStream = null;

private DataInputStream dataInputStream = null;

private String connectString = "socket://www.java-samples.com:80";

private StringBuffer results;

private Display myDisplay = null;

private Form resultScreen;

private StringItem resultField;

public SocketConnection() {

sults = new StringBuffer();

myDisplay = Display.getDisplay(this);

resultScreen = new Form("Page Content:");

}

public void startApp() {

try {

reamConnection =

(StreamConnection) Connector.open(connectString);

tputStream = streamConnection.openOutputStream();

dataOutputStream = new DataOutputStream(outputStream);

taOutputStream.writeChars("GET /index.htm HTTP/1.0 \n");

dataOutputStream.flush();

putStream = streamConnection.openInputStream();

dataInputStream = new DataInputStream(inputStream);

t inputChar;

while ( (inputChar = dataInputStream.read()) != -1) {

results.append((char) inputChar);

}

sultField = new StringItem(null, results.toString());

resultScreen.append(resultField);

myDisplay.setCurrent(resultScreen);

} catch (IOException e) {

System.err.println("Exception caught:" + e);

} finally {

y {

if (dataInputStream != null)

dataInputStream.close();

} catch (Exception ignored) {}

try {

if (dataOutputStream != null)

dataOutputStream.close();

} catch (Exception ignored) {}

try {

if (outputStream != null)

outputStream.close();

} catch (Exception ignored) {}

try {

if (inputStream != null)

inputStream.close();

} catch (Exception ignored) {}

try {

if (streamConnection != null)

streamConnection.close();

} catch (Exception ignored) {}

}

}

public void pauseApp() {

}

public void destroyApp(boolean unconditional) {

}

}

**Output :**

|  |  |
| --- | --- |
|  |  |

1. Create a MIDP application, which show to the user 5-10 quiz questions. ... shows them to user.

**Quiz.java :**

import javax.microedition.midlet.\*;

import javax.microedition.lcdui.\*;

public class Quiz extends MIDlet implements

CommandListener {

private Display display;

private Form form0, form1, form2, form3, form4, form5;

private ChoiceGroup ch0, ch1, ch2, ch3, ch4;

private Command next;

private Command back;

private Command ok;

private Command exit;

private StringItem st;

int count = 0;

public Quiz() {

display = Display.getDisplay(this);

next = new Command("Next", Command.OK, 1);

back = new Command("Back", Command.BACK, 1);

st = new StringItem("Total correct answers", "0");

form0 = new Form("1. 3\*5 is?");

ch0 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch0.append("9", null);

ch0.append("12", null);

ch0.append("15", null);

ch0.append("18", null);

form0.append(ch0);

form0.addCommand(next);

form0.setCommandListener(this);

form1 = new Form("2.Android is based on?");

ch1 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch1.append("Linux", null);

ch1.append("Unix", null);

ch1.append("Kai", null);

ch1.append("None of Above", null);

form1.append(ch1);

form1.addCommand(next);

form1.setCommandListener(this);

form2 = new Form("3. Is JAVA purely object oriented Language?");

ch2 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch2.append("True", null);

ch2.append("False", null);

form2.append(ch2);

form2.addCommand(next);

form2.addCommand(back);

form2.setCommandListener(this);

form3 = new Form("4.Root of 625?");

ch3 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch3.append("15", null);

ch3.append("35", null);

ch3.append("45", null);

ch3.append("25", null);

form3.append(ch3);

form3.addCommand(next);

form3.addCommand(back);

form3.setCommandListener(this);

form4 = new Form("5.Root of 144?");

ch4 = new ChoiceGroup("", Choice.EXCLUSIVE);

ch4.append("14", null);

ch4.append("12 ", null);

ch4.append("11", null);

ch4.append("It's not an Integer", null);

form4.append(ch4);

form4.addCommand(next);

form4.addCommand(back);

form4.setCommandListener(this);

form5 = new Form("Score");

exit = new Command("Exit", Command.SCREEN, 1);

ok = new Command("Submit", Command.OK, 2);

form5.addCommand(ok);

form5.addCommand(exit);

form5.setCommandListener(this);

} public void startApp() {

display.setCurrent(form0);

}

public void pauseApp() {}

public void destroyApp(boolean unconditional) {}

public void commandAction(Command cmd, Displayable displayable) {

if (displayable == form0) {

if (cmd == next)

display.setCurrent(form1);

} else if (displayable == form1) {

if (cmd == next)

display.setCurrent(form2);

else if (cmd == back)

display.setCurrent(form0);

} else if (displayable == form2) {

if (cmd == next)

display.setCurrent(form3);

else if (cmd == back)

display.setCurrent(form1);

} else if (displayable == form3) {

if (cmd == next)

display.setCurrent(form4);

else if (cmd == back)

display.setCurrent(form2);

} else if (displayable == form4) {

if (cmd == next) {

if (ch0.getSelectedIndex() == 2)

count++;

if (ch1.getSelectedIndex() == 0)

count++;

if (ch2.getSelectedIndex() == 1)

count++;

if (ch3.getSelectedIndex() == 3)

count++;

if (ch4.getSelectedIndex() == 1)

count++;

st.setText(String.valueOf(count));

form5.append(st);

display.setCurrent(form5);

}

} else if (displayable == form5) {

if (cmd == ok)

{display.setCurrent(form5);}

else if (cmd == exit) {

destroyApp(false);

notifyDestroyed();

}

}

}

}

**Output :**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |