

Lab Exercise 7

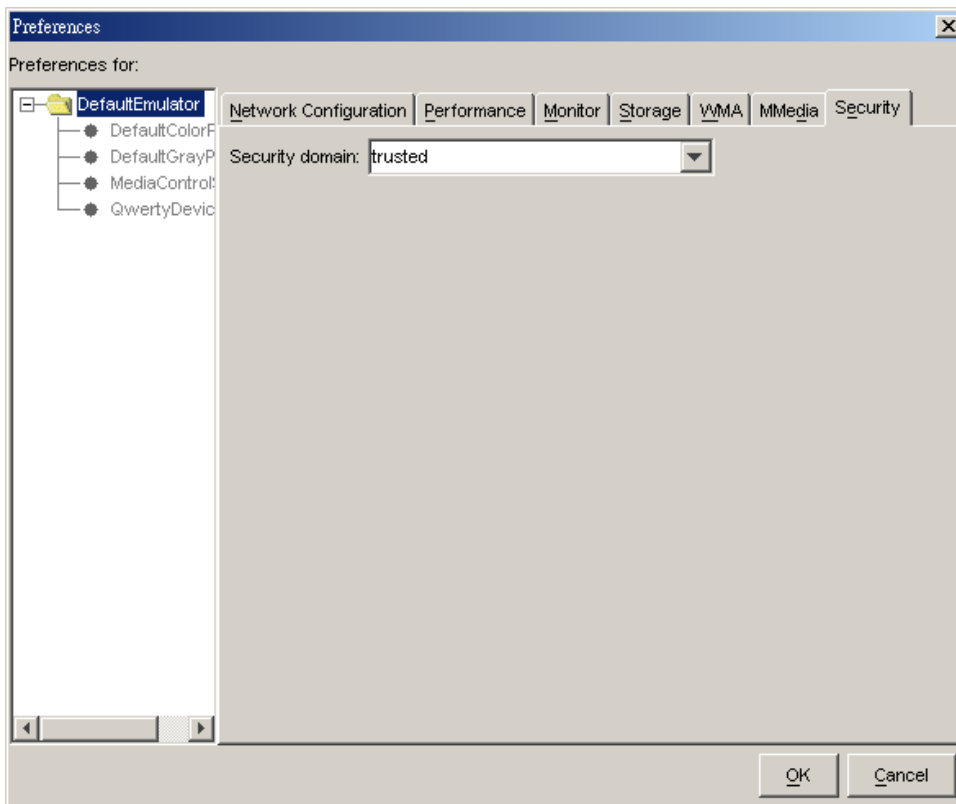
Please follow the instruction in Workshop Note 7 to complete this exercise.

1. Turn on your computer and startup **Windows XP** (*English version*), download and install **J2ME Wireless Toolkit 2.2** to your computer. Then download the laboratory resource files from <http://www.peter-lo.com/Teaching/I123-1-A/Source7.zip>.
2. Use Stream Connection and HTTP Connection to fetch an Internet file, then display it on your screen (*Page 1 – 6*)
3. Use HTTP Connection to involve a CGI script and obtain the return (*Page 7 – 8*)
4. Create a simple Image View to display PNG image from Internet (*Page 9 – 11*)
5. Develop a simple RMS Management application (*Page 12 – 14*)
6. Develop a simple local RPG. (*Page 15 – 18*)
7. Configure your IIS Web server in Windows XP, and then modify the above RPG program to an online RPG (*Page 19 – 24*)



1. Fetch a page using a `StreamConnection`

- In this example, you read the contents of a file referenced by a URL using `StreamConnection`. Here, there is no HTTP-specific behavior needed. The `Connector.open` opens a connection to the URL and returns a `StreamConnection` object. Then an `InputStream` is opened through which to read the contents of the file, character by character, until the end of the file (-1). In the event an exception is thrown, both the stream and connection are closed.
- Create a new project, name the project and the class at “`MyNetwork`” and “`MyClass`”. Then select **[Edit] → [Preference...]** from the menu bar and switch to “**Security**” tab. Change the security level to “**trusted**” and press **[OK]** to confirm.



- Create a java source files named “`MyClass.java`” in the “`src`” folder (C:\WTK21\apps\MyNetwork\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the MIDP I/O library
import javax.microedition.io.*;
// include the I/O library
import java.io.*;

/*****
/* All MIDlet applications must extend the MIDlet class.
*****/
```

```

public class MyClass extends MIDlet implements CommandListener {
    private Display midletDisplay; // Reference to Display object
    private Command cmdExit; // Command Button to exit the MIDlet
    private TextBox txtResult; // Textbox to receive result
    private StreamConnection conn; // Stream Connection
    private InputStream s; // Input Stream
    private StringBuffer b; // String Buffer to store result
    private int ch; // Define a character in ASCII code

    // Define the no-argument constructor
    public MyClass() {
        // Define the string buffer
        b = new StringBuffer();

        // Retrieve the display from the static display object
        midletDisplay = Display.getDisplay(this);
    }

    // Called by application manager to start the MIDlet.
    public void startApp() {
        try {
            // Create the connect using Stream connection
            conn = (StreamConnection)
                Connector.open("http://www.peter-lo.com/Teaching/I123-1-A/testing.txt");

            try {
                // Open input stream from the Stream connection
                s = conn.openInputStream();

                // Read character one by one from stream and append it to the string buffer
                while((ch = s.read()) != -1) {
                    b.append((char) ch);
                }

                // Put the string to the Textbox
                txtResult = new TextBox("Result", b.toString(), 256, 0);
            } finally {
                // Block of code that is always executed when the try block is exited,
                // no matter how the try block is exited.
                // Close the Input Stream when finish
                if(s != null) {
                    s.close();
                }
                // Close the Stream Connection when finish
                if (conn != null) {
                    conn.close();
                }
            }
        }

        // Create the Exit button
        cmdExit = new Command("Exit", Command.EXIT, 0);

        // Create Textbox and Commands, listen for events
        txtResult.addCommand(cmdExit);
        txtResult.setCommandListener(this);

        // Set the current display of the midlet to the Form
        midletDisplay.setCurrent(txtResult);
    } catch (IOException e) {
        // Handle Exceptions and print out on console.
        System.out.println("IOException " + e);
        e.printStackTrace();
    }
}

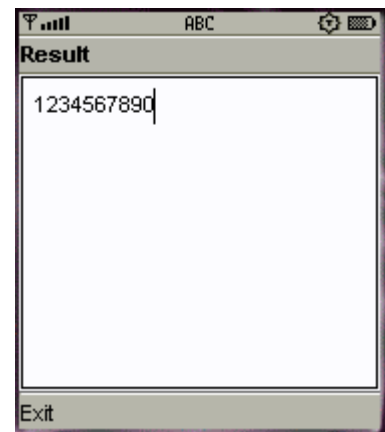
// PauseApp is used to suspend background activities and release resources
// on the device when the midlet is not active.
public void pauseApp() {
}

```

```
// DestroyApp is used to stop background activities and release
// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
}

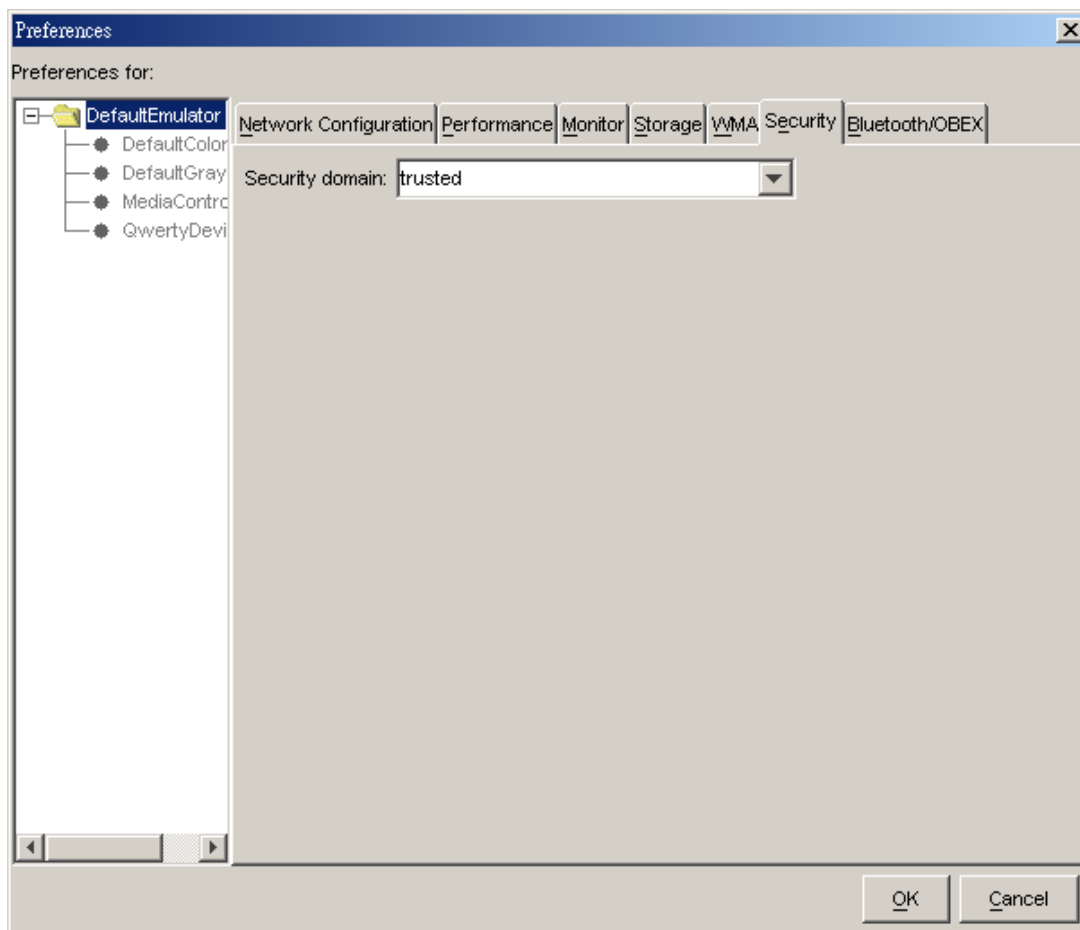
// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
        destroyApp(true);
        notifyDestroyed();
    }
}
}
```

4. Compile and execute the program. If you cannot execute the program or exception error throw, please check your Internet connection.



2. Fetch a Page Using an HttpURLConnection

1. When you open the input stream, the connection is opened and the HTTP headers read. The *getLength* gets the content length, and if you want to get the content type, use the *getType* method. The *Connector.open* opens a connection to the URL and returns a *StreamConnection* object. Then an *InputStream* is opened through which to read the contents of the file, character by character, until the end of the file (-1). In the event an exception is thrown, both the stream and connection are closed.
2. Create a new project, name the project and the class at “**MyNetwork**” and “**MyClass**”. Then select [Edit] → [Preference...] from the menu bar and switch to “**Security**” tab. Change the security level to “**trusted**” and press [OK] to confirm.



3. Create a java source files named “**MyClass.java**” in the “**src**” folder (C:\WTK22\apps\MyNetwork\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the MIDP I/O library
```

```

import javax.microedition.io.*;
// include the I/O library
import java.io.*;

/*****
/* All MIDlet applications must extend the MIDlet class.
*****/

public class MyClass extends MIDlet implements CommandListener {
    private Display midletDisplay; // Reference to Display object
    private Command cmdExit; // Command Button to exit the MIDlet
    private TextBox txtResult; // Textbox to receive result
    private HttpConnection conn; // HTTP Connection
    private InputStream s; // Input Stream
    private StringBuffer b; // String Buffer to store result
    private int ch; // Define a character in ASCII code

    // Define the no-argument constructor
    public MyClass() {
        // Define the string buffer
        b = new StringBuffer();

        // Retrieve the display from the static display object
        midletDisplay = Display.getDisplay(this);
    }

    // Called by application manager to start the MIDlet.
    public void startApp() {
        try {
            // Create the connect using HTTP connection
            conn = (HttpConnection)
                Connector.open("http://www.peter-lo.com/Teaching/I123-1-A/testing.txt");

            try {
                // Open input stream from the HTTP connection
                s = conn.openInputStream();

                // Read character one by one from stream and append it to the string buffer
                while((ch = s.read()) != -1) {
                    b.append((char) ch);
                }

                // Put the string to the Textbox
                txtResult = new TextBox("Result", b.toString(), 256, 0);
            } finally {
                // Close the Input Stream when finish
                if(s != null) {
                    s.close();
                }
                // Close the HTTP Connection when finish
                if (conn != null) {
                    conn.close();
                }
            }

            // Create the Exit button
            cmdExit = new Command("Exit", Command.EXIT, 0);

            // Create Textbox and Commands, listen for events
            txtResult.addCommand(cmdExit);
            txtResult.setCommandListener(this);

            // Set the current display of the midlet to the Form
            midletDisplay.setCurrent(txtResult);
        } catch (IOException e) {
            // Handle Exceptions and print out on console.
            System.out.println("IOException " + e);
            e.printStackTrace();
        }
    }
}

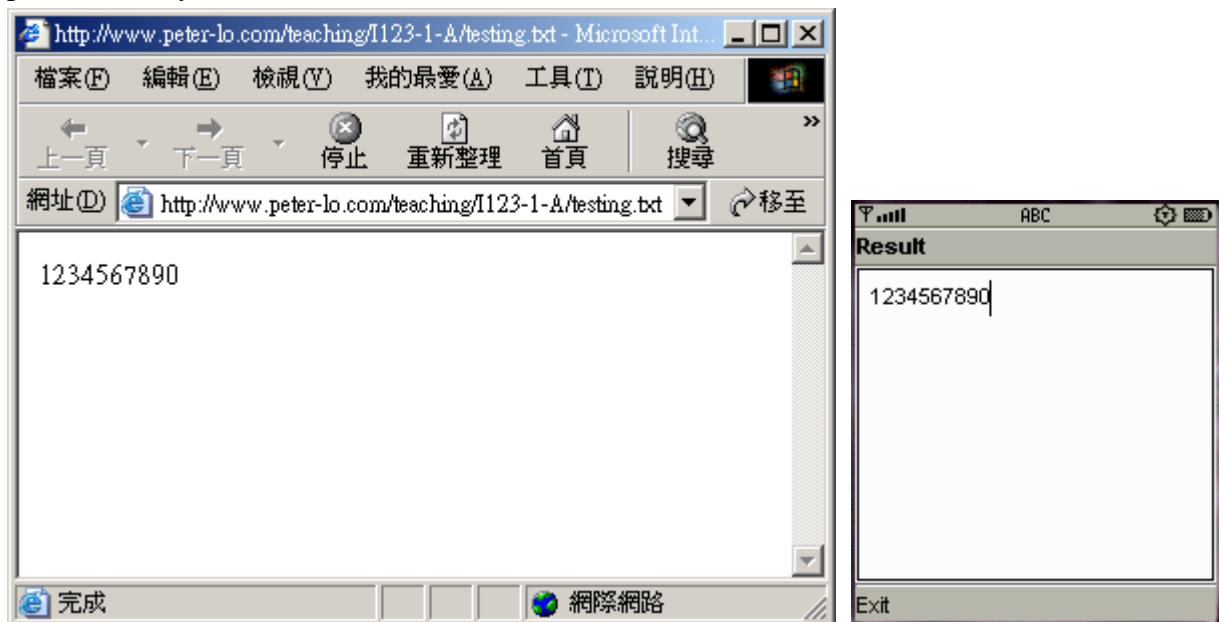
```

```
// PauseApp is used to suspend background activities and release resources
// on the device when the midlet is not active.
public void pauseApp() {
}

// DestroyApp is used to stop background activities and release
// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
}

// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
        destroyApp(true);
        notifyDestroyed();
    }
}
}
```

4. Compile and execute the program. If you cannot execute the program or exception error throw, please check your Internet connection.



3. Invoking a CGI Script

1. A CGI script is invoked from the HTTP server side, and the results are captured and displayed on the hand-held device screen. This example uses an *HttpConnection* object returned by the call to the *Connector.open* method. The call to *setRequestMethod* is set to **GET** and some request headers are also set. In this example, a script is invoked that accepts a name from a form. The script searches a file and returns the current date with the input name.
2. Open the previous project “*MyNetwork*”, and then modify the java source file “*MyClass.java*” (C:\WTK22\apps\MyNetwork\src) as follow:

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the MIDP I/O library
import javax.microedition.io.*;
// include the I/O library
import java.io.*;

/*****
/* All MIDlet applications must extend the MIDlet class.
*****/
public class MyClass extends MIDlet implements CommandListener {
    private Display midletDisplay; // Reference to Display object
    private Command cmdExit; // Command Button to exit the MIDlet
    private TextBox txtResult; // Textbox to receive result
    private HttpConnection conn; // HTTP Connection
    private InputStream s; // Input Stream
    private StringBuffer b; // String Buffer to store result
    private int ch; // Define a character in ASCII code

    // Define the no-argument constructor
    public MyClass() {
        // Define the string buffer
        b = new StringBuffer();

        // Retrieve the display from the static display object
        midletDisplay = Display.getDisplay(this);
    }

    // Called by application manager to start the MIDlet.
    public void startApp() {
        try {
            // Create the connect using HTTP connection
            conn = (HttpConnection) Connector.open
                ("http://www.peter-lo.com/Teaching/I123-1-A/testing.php?Name=Leo");

            // Set the Request method to GET
            conn.setRequestMethod(HttpConnection.GET);

            try {
                // Open input stream from the HTTP connection
                s = conn.openInputStream();

                // Read character one by one from stream and append it to the string buffer
                while((ch = s.read()) != -1) {
                    b.append((char) ch);
                }
            }
        }
    }
}
```

```

// Put the string to the Textbox
txtResult = new TextBox("Result", b.toString(), 256, 0);
} finally {
// Close the Input Stream when finish
if(s != null) {
s.close();
}
// Close the HTTP Connection when finish
if (conn != null) {
conn.close();
}
}

// Create the Exit button
cmdExit = new Command("Exit", Command.EXIT, 0);

// Create Textbox and Commands, listen for events
txtResult.addCommand(cmdExit);
txtResult.setCommandListener(this);

// Set the current display of the midlet to the Form
midletDisplay.setCurrent(txtResult);
} catch (IOException e) {
// Handle Exceptions and print out on console.
System.out.println("IOException " + e);
e.printStackTrace();
}
}

// PauseApp is used to suspend background activities and release resources
// on the device when the midlet is not active.
public void pauseApp() {
}

// DestroyApp is used to stop background activities and release
// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
}

// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
if (c == cmdExit) {
destroyApp(true);
notifyDestroyed();
}
}
}
}

```

3. Compile and execute the program. If you cannot execute the program or exception error throw, please check your Internet connection.



4. Simple Image Viewer

1. Create a new project and name the class as “**PngViewer**”. Then create a java files called “**PngViewer.java**” in the “src” folder.

PngViewer.java

```
// include the MIDP library
import javax.microedition.midlet.*;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the MIDP I/O library
import javax.microedition.io.*;
// include the I/O library
import java.io.*;

/*-----*/
/* Java class: PngViewer.java */
/*-----*/
/* PNG Viewer to view Internet Photo */
/*-----*/

public class PngViewer extends MIDlet implements CommandListener {
    private Display display;
    private TextBox tbMain;
    private Form fmViewPng;
    private Command cmExit;
    private Command cmView;
    private Command cmdBack;

    // Constructor for the class.
    public PngViewer() {
        display = Display.getDisplay(this);

        // Create the Main textbox with a maximum of 100 characters
        tbMain = new TextBox("Enter PNG URL",
"http://www.peter-lo.com/Teaching/I123-1-A/sample.png", 100, 0);

        // Create commands and add to textbox
        cmView = new Command("View", Command.SCREEN, 1);
        cmExit = new Command("Exit", Command.EXIT, 0);
        tbMain.addCommand(cmView );
        tbMain.addCommand(cmExit);

        // Set up a listener to "capture" button presses
        tbMain.setCommandListener(this);

        // Create the form that will hold the png image
        fmViewPng = new Form("PNG Image");

        // Create commands and add to form
        cmdBack = new Command("Back", Command.BACK, 1);
        fmViewPng.addCommand(cmdBack);

        // Set up a listener to "capture" button presses
        fmViewPng.setCommandListener(this);
    }

    // Called by the Application Manager on the device to start the MIDlet.
    public void startApp() {
        // Display the Main textbox
        display.setCurrent(tbMain);
    }
}
```

```

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

// Process events
public void commandAction(Command c, Displayable s) {
    // If the Command button pressed was "Exit"
    if (c == cmExit) {
        destroyApp(false);
        notifyDestroyed();
    }
    else if (c == cmView) { // If the Command button pressed was "View"
        // Remove anything that may be on the form
        if (fmViewPng.size() > 0)
            for (int i = 0; i < fmViewPng.size(); i++)
                fmViewPng.delete(i);

        try {
            // Get the image from the web and append to the form
            Image im;
            if ((im = getImage(tbMain.getString())) != null)
                fmViewPng.append(im);

            // Display the form with the image
            display.setCurrent(fmViewPng);
        } catch (Exception e) {
            System.err.println("Msg: " + e.toString());
        }
    } else if (c == cmdBack) // If the Command button pressed was "Back"
        display.setCurrent(tbMain);
}

// Open an http connection and download a png file into a byte array.
private Image getImage(String url) throws IOException {
    ContentConnection connection = (ContentConnection) Connector.open(url);
    DataInputStream iStrm = connection.openDataInputStream();

    Image im = null;

    try {
        // ContentConnection includes a length method
        byte imageData[];
        int length = (int) connection.getLength();
        if (length != -1) {
            imageData = new byte[length];

            // Read the png into an array
            iStrm.readFully(imageData);
        } else { // Length not available...
            ByteArrayOutputStream bStrm = new ByteArrayOutputStream();

            int ch;
            while ((ch = iStrm.read()) != -1)
                bStrm.write(ch);

            imageData = bStrm.toByteArray();
            bStrm.close();
        }

        // Create the image from the byte array
        im = Image.createImage(imageData, 0, imageData.length);
    } finally {
        // Clean up
        if (iStrm != null)
            iStrm.close();
        if (connection != null)
            connection.close();
    }
}

```

```
return (im == null ? null : im);  
}  
}
```

2. Compile and execute the J2ME program. Can you download the photo after input the corresponding image path?



5. Simple RMS Management

1. Create a new project, name the project and the class as “MyRMS” and “MyClass”. Then create a java source files named “MyClass.java” in the “src” folder (C:\WTK22\apps\MyRMS\src).

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the RMS library
import javax.microedition.rms.*;
// include the I/O library
import javax.microedition.io.*;

/*****
/* All MIDlet applications must extend the MIDlet class.
*****/
public class MyClass extends MIDlet implements CommandListener {
    private Display midletDisplay;           // Reference to Display object
    private TextBox txtResult;              // Textbox to receive result
    private Command cmdExit;                // Command to exit the MIDlet
    private Command cmdSave;                // Command to Save the Record
    private Command cmdLoad;                // Command to Load the Record
    private Command cmdDelete;              // Command to Delete the Record
    private RecordStore rs;                  // Record Store
    private int dataLen;                     // Length of Record Data
    private byte[] RecData = new byte[50]; // Define a byte array to store record data

    // Define the constant for the name of the RMS
    final String RECORD_STORE = "MyStore";

    // Define the no-argument constructor
    public MyClass() {
        // Put the string to the Textbox
        txtResult = new TextBox("Result", "", 50, 0);

        // Create the Exit, Save, Load and Delete button
        cmdExit = new Command("Exit", Command.EXIT, 0);
        cmdSave = new Command("Save", Command.SCREEN, 1);
        cmdLoad = new Command("Load", Command.SCREEN, 1);
        cmdDelete = new Command("Delete", Command.SCREEN, 1);

        // Retrieve the display from the static display object
        midletDisplay = Display.getDisplay(this);
    }

    // Called by application manager to start the MIDlet.
    public void startApp() {
        // Create Textbox and Commands, listen for events
        txtResult.addCommand(cmdExit);
        txtResult.addCommand(cmdSave);
        txtResult.addCommand(cmdLoad);
        txtResult.addCommand(cmdDelete);
        txtResult.setCommandListener(this);

        // Set the current display of the midlet to the Form
        midletDisplay.setCurrent(txtResult);
    }

    // PauseApp is used to suspend background activities and release resources

```

```
// on the device when the midlet is not active.
public void pauseApp() {
}

// DestroyApp is used to stop background activities and release
// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
}

// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
        destroyApp(true);
        notifyDestroyed();
    } else if (c == cmdSave) {
        // Open the record store
        OpenRMS();

        // Write the Textbox content to the record store
        SaveRMS(txtResult.getString());

        // Close record store
        CloseRMS();
    } else if (c == cmdLoad) {
        // Open the record store
        OpenRMS();

        // Read from the record store
        ReadRMS();

        // Close record store
        CloseRMS();

        // Set the current display of the midlet to the Form
        midletDisplay.setCurrent(txtResult);
    } else if (c == cmdDelete) {
        // Delete record store
        DeleteRMS();
    }
}

public void OpenRMS() {
    try {
        // Create the record store if it does not exist
        rs = RecordStore.openRecordStore(RECORD_STORE, true);
    } catch (Exception e) {
        System.out.println(e);
    }
}

public void ReadRMS() {
    try {
        for (int i = 1; i <= rs.getNumRecords(); i++) {
            // get the data and return the string length
            dataLen = rs.getRecord(i, RecData, 0);
        }

        // Append the last string to TextBox
        txtResult.setString(new String(RecData, 0, dataLen));
    } catch (Exception e) {
        System.out.println(e);
    }
}

public void SaveRMS(String str) {
    // Use a byte array to store the string
    RecData = str.getBytes();

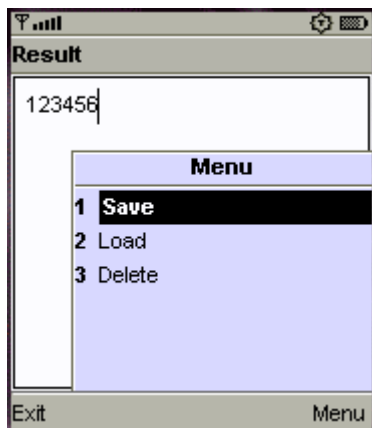
    try {
        // Add the record to the record store
    }
}
```

```
        rs.addRecord(RecData, 0, RecData.length);
    } catch (Exception e) {
        System.out.println(e);
    }
}

public void CloseRMS() {
    try {
        // Close the record store
        rs.closeRecordStore();
    } catch (Exception e) {
        System.out.println(e);
    }
}

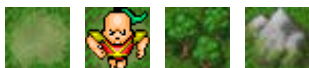
public void DeleteRMS() {
    if (RecordStore.listRecordStores() != null) {
        try {
            // Delete the record store if it exist
            RecordStore.deleteRecordStore(RECORD_STORE);
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
}
```

2. Compile and Execute the program, you can use “SAVE”, “LOAD” and “DELETE” to manage the record.



6. Simple RPG Demo

1. Create a new project, name the project and the class as “LocalRPG” and “MyClass”.
2. Design your own RPG map, and select useful map component image, character images and the background music into the “res” folder (C:\WTK22\apps\LocalRPG\res).



3. Create two java files “MyClass.java” and “ImageCanvas.java” in the “src” folder (C:\WTK22\apps\LocalRPG\src). You can modify the variable “BigMap” in order to create a beautiful map, and modify the variable “UserID” to change other character.

MyClass.java

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the media library
import javax.microedition.media.*;
// include the I/O library
import java.io.*;

/*****
/* Sample 2D RPG Game.
*****/

public class MyClass extends MIDlet implements CommandListener {
    // Define the GUI components
    private ImageCanvas MyCanvas;           // Canvas
    private Command cmdExit;                // Exit Button
    private Thread MyThread;                // Thread
    private Player p;                        // Player
    InputStream infile;                      // Input Buffer

    // Define the no-argument constructor
    public MyClass() {
        // Define the Exit Button
        cmdExit = new Command("Exit", Command.EXIT, 0);

        // Define the canvas
        MyCanvas = new ImageCanvas();

        try {
            // Read midi file, packaged in the jar file
            infile = getClass().getResourceAsStream("/rpg.mid");

            // Set the player to play midi
            p = Manager.createPlayer(infile, "audio/midi");

            // Constructs Player without acquiring the scarce and exclusive resources
            p.realize();

            // Define the play back
            p.setLoopCount(99);
        }
    }
}
```

```

    // Acquires the scarce and exclusive resources and processes
    // as much data as necessary to reduce the start latency
    p.prefetch();

    // Start playing multimedia file
    p.start();
} catch (Exception e) {
    System.out.print ("Error! Unable to play Midi file!");
}
}

// Called by application manager to start the MIDlet
public void startApp() {
    // Add the Command button and the Command Listener
    MyCanvas.addCommand(cmdExit);
    MyCanvas.setCommandListener(this);

    // Set the current display of the midlet to the Canvas
    Display.getDisplay(this).setCurrent(MyCanvas);

    // Start the thread only one time
    if (MyThread == null) {
        // Define the thread
        MyThread = new Thread(MyCanvas);

        // Start the thread
        MyThread.start();
    }
}

// PauseApp is used to suspend background activities and release resources
// on the device when the midlet is not active.
public void pauseApp() {
}

// DestroyApp is used to stop background activities and release
// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
    // Release the resource
    p.deallocate();

    // Close the Player
    p.close();
}

// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
        destroyApp(true);
        notifyDestroyed();
    }
}
}
}

```

ImageCanvas.java

```

// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the I/O library
import java.io.*;
// include the MIDP I/O library
import javax.microedition.io.*;

/*-----*/
/* Canvas for graphic display */
/*-----*/

```

```

public class ImageCanvas extends Canvas implements Runnable {
    private Image img[] = new Image[15]; // Image for characters and items
    private int i, j; // Loop variables
    private int UserID = 5; // User ID: (1 - 7)
    private int x = 2; // Character x-coordinate
    private int y = 2; // Character y-coordinate
    private boolean ExitFlag = false; // The Flag to Exit the Thread

    // Whole RPG Map
    private int BigMap[][] = {{9,9,9,9,9,9,9,9,9,9,9,9,9,9,9},
                              {9,9,9,9,9,9,9,9,9,9,9,9,9,9,9},
                              {9,9,0,0,0,0,0,0,8,8,8,0,0,0,0,9,9},
                              {9,9,0,0,0,0,0,0,0,8,0,0,0,0,0,0,9,9},
                              {9,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,9},
                              {9,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,9},
                              {9,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,9},
                              {9,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,9},
                              {9,9,0,8,8,8,8,8,0,0,0,8,8,0,8,0,9,9},
                              {9,9,8,8,0,0,0,0,0,0,0,0,0,0,0,0,9,9},
                              {9,9,8,0,0,0,0,8,0,0,0,0,0,8,0,8,0,9,9},
                              {9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9},
                              {9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9}};

    // Initial the User Display Map
    private int Map[][] = {{9,9,9,9,9},
                          {9,9,9,9,9},
                          {9,9,UserID,0,0},
                          {9,9,0,0,0},
                          {9,9,0,0,0}};

    public ImageCanvas() {
        // Load the image from resource folder
        try {
            for (i=0 ; i<10; i++) {
                img[i] = Image.createImage("/" + i + ".png");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    // Execute the Thread
    public void run() {
        try {
            // Continuous to repaint the screen until the ExitFlag = True
            while (!ExitFlag) {
                repaint(); // Call paint() to repaint the screen
                Thread.sleep(100); // Sleep for 0.1 second
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    public void keyPressed(int keyCode) {
        // Clear the character on the big map
        BigMap[x][y] = 0;

        if (getGameAction(keyCode) == LEFT) {
            // Allow to move Left only if no blocking
            if (BigMap[x][y-1] == 0) {
                y = y - 1;
            }
        } else if (getGameAction(keyCode) == RIGHT) {
            // Allow to move Right only if no blocking
            if (BigMap[x][y+1] == 0) {
                y = y + 1;
            }
        } else if (getGameAction(keyCode) == UP) {
            // Allow to move Up only if no blocking

```

```

    if (BigMap[x-1][y] == 0) {
        x = x - 1;
    }
} else if (getGameAction(keyCode) == DOWN) {
    // Allow to move Down only if no blocking
    if (BigMap[x+1][y] == 0) {
        x = x + 1;
    }
}

// Locate the character on the big map
BigMap[x][y] = UserID;

// Calculate the new map for user
for (i=x-2; i<=x+2; i++) {
    for (j=y-2; j<=y+2; j++) {
        Map[i-x+2][j-y+2] = BigMap[i][j];
    }
}

public void paint(Graphics g) {
    // Clear the screen
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, getWidth(), getHeight());

    // Draw the map on screen
    for (i=0; i<5; i++) {
        for (j=0; j<5; j++) {
            g.drawImage(img[Map[j][i]], i*32, j*32, g.LEFT|g.TOP);
        }
    }
}
}
}

```

4. Compile and execute the program, you can use the arrow key to move your character to move around the map.



7. Simple Online RPG Demo

1. Create a new project, name the project and the class as “**OnlineRPG**” and “**MyClass**”.
2. Put all the character image, map components and background music into the “**res**” folder (C:\WTK22\apps\OnlineRPG\res).



3. Create two java files “**MyClass.java**” and “**ImageCanvas.java**” in the “**src**” folder (C:\WTK22\apps\OnlineRPG\src). You can modify the variable “**UserID**” for using other character.

MyClass.java

```
// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the media library
import javax.microedition.media.*;
// include the I/O library
import java.io.*;

/*****
/* Sample Online RPG Game.          */
*****/
public class MyClass extends MIDlet implements CommandListener {
    // Define the GUI components
    private ImageCanvas MyCanvas;      // Canvas
    private Command cmdExit;           // Exit Button
    private Thread MyThread;           // Thread
    private Player p;                  // Player
    InputStream infile;                // Input Buffer

    // Define the no-argument constructor
    public MyClass() {
        // Define the Exit Button
        cmdExit = new Command("Exit", Command.EXIT, 0);

        // Define the canvas
        MyCanvas = new ImageCanvas();

        try {
            // Read midi file, packaged in the jar file
            infile = getClass().getResourceAsStream("/rpg.mid");

            // Set the player to play midi
            p = Manager.createPlayer(infile, "audio/midi");

            // Constructs Player without acquiring the scarce and exclusive resources
            p.realize();

            // Define the play back
            p.setLoopCount(99);
        }
    }
}
```

```

    // Acquires the scarce and exclusive resources and processes
    // as much data as necessary to reduce the start latency
    p.prefetch();

    // Start playing multimedia file
    p.start();
} catch (Exception e) {
    System.out.print ("Error! Unable to play Midi file!");
}
}

// Called by application manager to start the MIDlet
public void startApp() {
    // Add the Command button and the Command Listener
    MyCanvas.addCommand(cmdExit);
    MyCanvas.setCommandListener(this);

    // Set the current display of the midlet to the Canvas
    Display.getDisplay(this).setCurrent(MyCanvas);

    // Start the thread only one time
    if (MyThread == null) {
        // Define the thread
        MyThread = new Thread(MyCanvas);

        // Start the thread
        MyThread.start();
    }
}

// PauseApp is used to suspend background activities and release resources
// on the device when the midlet is not active.
public void pauseApp() {
}

// DestroyApp is used to stop background activities and release
// resources on the device when the midlet is at the end of its life cycle.
public void destroyApp(boolean unconditional) {
    // Release the resource
    p.deallocate();

    // Close the Player
    p.close();
}

// Implement the event handling method defined in the CommandListener interface.
public void commandAction(Command c, Displayable s) {
    if (c == cmdExit) {
        destroyApp(true);
        notifyDestroyed();
    }
}
}
}

```

ImageCanvas.java

```

// include the MIDlet supper class
import javax.microedition.midlet.MIDlet;
// include the GUI libraries of MIDP
import javax.microedition.lcdui.*;
// include the I/O library
import java.io.*;
// include the MIDP I/O library
import javax.microedition.io.*;

/*-----*/
/* Canvas for graphic display */
/*-----*/

```

```

public class ImageCanvas extends Canvas implements Runnable {
    private Image img[] = new Image[15]; // Image for characters and items
    private int i, j; // Loop variables
    private int UserID = 3; // User ID: (1 - 7)
    private int x = 3; // Character x-coordinate
    private int y = 3; // Character y-coordinate
    private int motion = 0; // User Movement Direction
    private boolean ExitFlag = false; // The Flag to Exit the Thread
    private String ServerURL; // URL Address
    private HttpURLConnection conn; // Stream Connection
    private InputStream s; // Input Stream
    private int ch; // define a character in ASCII code

    // Initial the User Display Map
    private int Map[][] = {{9,9,9,9,9},
                           {9,9,9,9,9},
                           {9,9,UserID,0,0},
                           {9,9,0,0,0},
                           {9,9,0,0,0}};

    public ImageCanvas() {
        // Load the image from resource folder
        try {
            for (i=0 ; i<10; i++) {
                img[i] = Image.createImage("/") + i + ".png");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    // Execute the Thread
    public void run() {
        try {
            // Continuous to repaint the screen until the ExitFlag = True
            while (!ExitFlag) {
                repaint(); // Call paint() to repaint the screen
                Thread.sleep(100); // Sleep for 0.1 second
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    public void keyPressed(int keyCode) {
        motion = 0;
        if (getGameAction(keyCode) == LEFT && x > 3) {
            motion = 1;
        } else if (getGameAction(keyCode) == RIGHT && x < 99) {
            motion = 2;
        } else if (getGameAction(keyCode) == UP && y > 3) {
            motion = 3;
        } else if (getGameAction(keyCode) == DOWN && y < 99) {
            motion = 4;
        }

        if (motion > 0) {
            // Generate the CGI Address for passing parameters
            ServerURL = "http://localhost/I123-1-A/RPG.asp?u=" + UserID + "&x=" + x + "&y=" +
y + "&m=" + motion;

            // Create a new thread to handle network connection
            Thread t = new Thread() {
                public void run() {
                    try {
                        // Create the connect using Stream connection
                        conn = (HttpURLConnection)Connector.open(ServerURL);

                        // Set the Request method to GET
                        conn.setRequestMethod(HttpURLConnection.GET);
                    }
                }
            };
            t.start();
        }
    }
}

```

```

try {
    // Open input stream from the Stream connection and prepare to read the content
    s = conn.openInputStream();

    // Read the new x coordinate
    x = (int)s.read();

    // Read the new y coordinate
    y = (int)s.read();

    // Read the Map
    for (i=0; i<5; i++) {
        for (j=0; j<5; j++) {
            ch = s.read();
            Map[i][j] = ch - 48;
        }
    }
} finally {
    // Close the Input Stream when finish
    if (s != null) {
        s.close();
    }

    // Close the Stream Connection when finish
    if (conn != null) {
        conn.close();
    }
}
} catch (IOException e) {
    System.out.println("IOException " + e);
}
};
t.start();
}
}

public void paint(Graphics g) {
    // Clear the screen
    g.setColor(255, 255, 255);
    g.fillRect(0, 0, getWidth(), getHeight());

    // Draw the map on screen
    for (i=0; i<5; i++) {
        for (j=0; j<5; j++) {
            g.drawImage(img[Map[j][i]], i*32, j*32, g.LEFT|g.TOP);
        }
    }
}
}
}

```

4. Create a map file called “**Map.txt**” (use 0 for Ground, 8 for forest, and 9 for Hill) in your IIS Web server (C:\Inetpub\wwwroot\map.txt).



5. Add the following code in the “**Global.asa**” (C:\Inetpub\wwwroot\Global.asa).

```

<Script Language=VBScript RunAt=Server>
Sub Application_OnStart
    dim LocalArray(100, 100)

    ` Read the file
    Set fs = Server.CreateObject("Scripting.FileSystemObject")
    Map_file = Server.MapPath("/map.txt")
    Set txt = fs.OpenTextFile( Map_file )

    ` Read the Map
    For i = 1 to 25
        tmp = txt.ReadLine
        for j = 1 to 100
            LocalArray(i, j) = mid(tmp, j, 1)
        next
    next
    Application("MapArray") = LocalArray

    txt.Close
End Sub
</Script>

```

6. Create an ASP file in the web server, and named it as “**rpg.asp**”.

```

<%
` Obtain the User parameters
UserID = Request("u")
x = Request("x")
y = Request("y")
Motion = Request("m")

` Lock the Application
Application.Lock

` Process and calculate the new location
LocalArray = Application("MapArray")
If Motion = 1 Then ` Left
    If LocalArray(x, y-1) = "0" Then
        LocalArray(x, y) = "0"
        y = y - 1
        LocalArray(x, y) = UserID
    End If
ElseIf Motion = 2 Then ` Right
    If LocalArray(x, y+1) = 0 Then
        LocalArray(x, y) = 0
        y = y + 1
        LocalArray(x, y) = UserID
    End If
ElseIf Motion = 3 Then ` Up
    If LocalArray(x-1, y) = "0" Then
        LocalArray(x, y) = "0"
        x = x - 1
        LocalArray(x, y) = UserID
    End If
ElseIf Motion = 4 Then ` Down
    If LocalArray(x+1, y) = "0" Then
        LocalArray(x, y) = "0"
        x = x + 1
        LocalArray(x, y) = UserID
    End If
End If
Application("MapArray") = LocalArray

` Unlock the Application
Application.Unlock

` Write the result to user
response.write chr(x)

```

```
response.write chr(y)
for i = x-2 to x+2
  for j = y-2 to y+2
    response.write LocalArray(i,j)
  next
next
next
%>
```

7. Compile and execute the J2ME program. Let's explore the map with your friends

