# Java ME for Education

## Exercise book

## v 1.0

# Module 0: Recap of OO and Threading

**Aim:** Recap of object orientated programming with Java.

**Timing estimate**: 120 minutes

## 1. Composition and Inheritance

**Class**

Design a class Point that represents x and y values. Add setPoint(int x, int y) method that initializes attributes. Make a proper String toString() and boolean equals(Object a) - methods. Make two constructors: default and Point(int x, int y). Default constructor initializes x and y values to origo.  
  
**Composition**

Make a Line-class. Line consists of two points (begin and end). Implement constructor that initializes begin and end points to origo. Implement also setLine-method where one can initialize line's begin and end points. Create proper String toString()-method (returns end and begin x y values).

**Inheritance**

Implement a new class Pixel that inherites Point. Pixel represents a pixel in a monitor. Pixel should have x, y and color (java.awt.Color)- values. Create constructor Pixel(int x, int y, Color color). Implement also setPixel(int x, int y, Color color) method. Use functionality coded into Point class as much as you can.

**Running**

Create class AppRunner that has only one method: main. Test in main method classes Point, Line and Pixel.

# Javadoc

# Comment your code using Javadoc and create html-documentation from this exercise.

## 2. Threading

# Make a class Coin. Coin has one private attribute coinSide which type is boolean. Classes default constructor calls throwCoin() method. The public void throwCoin() - method initializes random value (either true or false) to the attribute coinSide (See documentation of class Math). Create accessor method to the attribute.

# Create a main method inside class Coin. Throw the coin ten times in a separate thread. Wait one second between each coin toss. Print to the console texts like "Coin was thrown: heads". When the thread finishes, print how many heads and how many tails were thrown.

# Module 1: Java ME Development Overview

**Aim:** Learn how to create, compile and deploy a MIDP – application

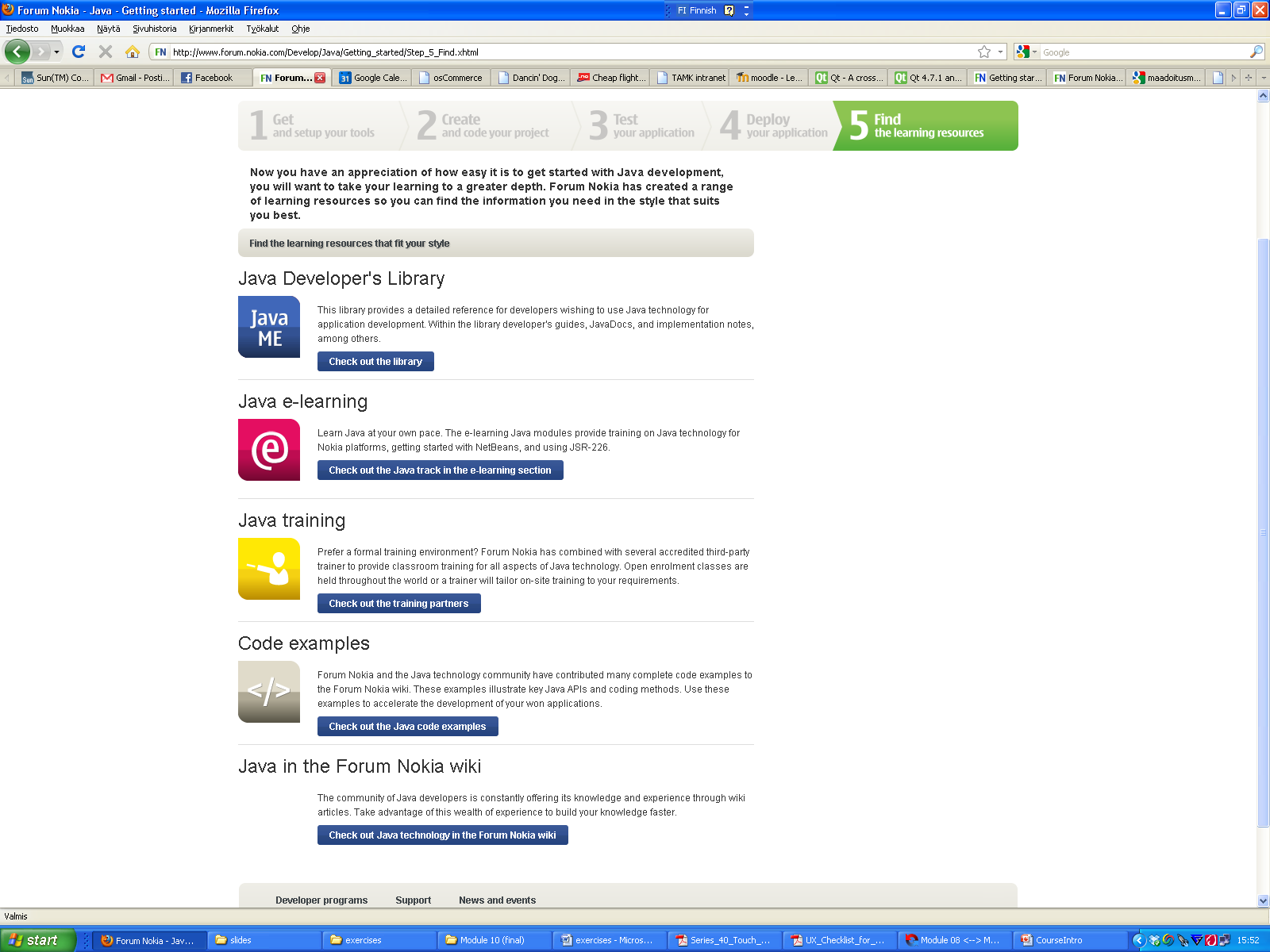
**Timing estimate**: 90-120 minutes depending on if you have the SDKs and tools installed.

## 1. Developing your first Java ME application

Follow the instructions in Java - Getting Started website of Forum Nokia:

http://www.forum.nokia.com/Develop/Java/Getting\_started/

In the exercise you will compile and deploy your first MIDP application wit NetBeans by following the step-by-step instructions given. Get also familiar with Java learning resources of Forum Nokia (step 5 of the Getting Started workshop):



## 2. Answer to the questions

# What is the trade-off between memory and speed?

# How can you trace call coverage in your MIDlet?

# How can you use the JDK to debug your MIDlet?

# How can you debug and profile your MIDlet in a target device?

# What are some of the ways you can deploy a MIDlet to a physical device?

# What is OTA and why is it important?

# How can you test your MIDlet so that it supports OTA?

# How can you test the memory usage of your MIDlet?

# How can you test the runtime speed of your MIDlet?

# Why is the emulator useful?

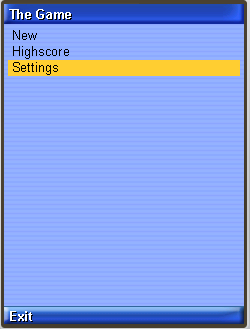
# Why must you always test your MIDlet on a physical device before shipment?

# Module 2: High Level UI API

**Aim:** Getting familiar with High Level UI API and Event Handling.

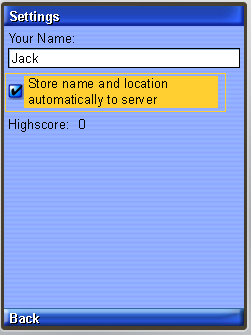
**Timing estimate**: 90 minutes

## 1. Main Screen for a Game

In modules 3 to 10 you will be creating a game with collision detecting, highscore, sounds, network connection and more. In this exercise you will be creating the main screen for you game application.

1. Start a new Java ME project. Create new MIDlet "MainScreen"
2. When the MIDlet launches, display a implicit List with three choices: "New Game", "Highscore" and "Settings"
3. When user selects one of given options, an Alert is displayed showing the text user selected
4. Implement also an exit command. When selected, the MIDlet exists

## 2. Settings Screen for the Game

Now implement a settings screen for the game. In the settings, user is able to give his/her name and enable automatic connection to server. Also user can view his/her personal high score.

1. Create new class "SettingsScreen" which inherites Form. Display the SettingsScreen when user selects Settings from the main screen.
2. Add to the Form a new TextField, where user is able to write his/her name
3. Add to the Form a checkbox (ChoiceGroup) where user can enable automatic connection to server and location. When user achieves a new personal highscore, the score will be sent to server with user's location automatically. This functionality will be implemented later.
4. Add StringItem to the form. StringItem displays the personal highscore, at this time 0.
5. Implement also "Back" command which brings the user to the main screen.

## 3. Answer the questions

# True or False: Use the high-level API whenever possible because it is portable

# True or False: If you need special screen layouts (e.g. games) then use the low-level API

# What are tickers used for?

# What are abstract commands?

# How to you capture user events?

# How does the look-and-feel of the High Level API vary over platforms?

# How can you create custom widgets?

# What is a Screen?

# What are the different types of Screens?

# How many screens can be attached to a display at the same time?

# How do you override the drawing and customize a Screen or Form Item?

# Module 3: Low Level UI API

**Aim:** Getting familiar with Low Level UI API and Threading

**Timing estimate**: 90 minutes

## 1. Splash Screen

Implement Splash Screen to your Game. Splash Screen is shown three seconds when the app is launching. User is able to bypass the splash screen by pressing some key or giving touch input.

1. Create new class "SplashScreen" which inherites Canvas
2. Implement the paint(Graphics g) - method and draw a beautiful splash screen to your game. (Implementation free)
3. When the Splash Screen is initalized, start a new Thread. Thread sleeps for three seconds and after that the main screen is shown
4. Implement keyReleased and pointerReleased methods where user can bypass the splash screen and move straight to the main screen
5. In the main screen, set the splash screen as current when the MIDlet starts.

## 2. Answer the questions

# True or False: Update the Canvas intelligently or your MIDlet will be slow

# True or False:The KVM is fast at number crunching but slow at accessing the native OS calls used for drawing

# In addition to the paint event, why would the developer wish to override the key event even if there was no interaction with the MIDlet via key events?

# Is the low-level graphics API threadsafe? What concurrency issues must the developer always be watchful of?

# How would you implement double buffering if your device doesn’t automatically support it?

# Why would it make sense to implement a MIDlet only with the low-level API even though multiple display “cards” or available with each MIDlet deck?

# What is the main reason why the Nokia UI is implemented the way it is?

# Module 4: Game API and Graphics

**Aim:** Getting familiar with game development using the Game API.

Farmer Smith's home has been attached by dangerous singing birds (that don't sing in this exercise). Because these birds are extremely dangerous, farmer Smith is determined to kill all the birds. Smith's only weapons are stones; whenever Smith sees a dangerous singing bird, he tries to throw a stone towards the bird.

Since the birds are very aggressive, Smith will die if he has a contact with the birds. It's very important that Smith is able to kill the bird with the stones before the bird is in contact with Smith.

The idea is to implement a game, where a dangerous singing bird is flying horizontally with certain speed. When the bird hits the end of the screen, bird turns around and lowers it's vertical direction with couple of pixels. Smith is standing on the ground and if he gets hit by the bird, game is over. Smith is able to throw stones towards the bird and if the bird is hit by the stone, bird is killed. Whenever a bird is killed, new bird appears to the screen (upper-corned) and this bird will fly faster than the previous one.

**Timing estimate**: 180 minutes

1. **Background:** Create new GameCanvas - class. Use grass.png as a background - image. Create TiledLayer from the image and repeat the image. This TiledLayer will be the grass where Smith will walk. Create also a text on the upper-corner of the screen where you tell to the player how many birds have been killed.
2. **Mr Smith:** Put Mr Smith on the ground, on top of the grass (middle). Create functionality where you are able to move Smith from left to right. Transform the Sprite, when the direction is changed. Animate the sprite when Smith is walking. Use mrsmith.png.
3. **Bird:** Implement functionality where the dangerous singing bird is able to fly across the screen. Bird appears from the left upper-corner of the screen and when it collides with the right edge of the screen, bird will turn and lower it's position with couple of pixels. Animate the bird when it is flying. Use bird.png
4. **Collisiong with Mr Smith:** Implement functionality where the game ends when the bird hits Smith.
5. **Throwing stones:** Implement the functionality needed when Smith is throwing stones towards the bird. When the stone collides with the bird, the bird will explode. When the bird has exploded, new bird appears in the upper corner and this time the bird will have more speed. Use rock.png and explosion.png

# Module 5: Persistent Storage

**Aim:** Getting familiar with RMS. Storing and Retrieving records from Record Store

**Timing estimate**: 90 minutes

Implement functionality where game settings including user name, enable automatic connections and high score are stored persistently.

# Module 6: Networking

**Aim:** Getting familiar with http connection. Using http as a interface.

This exercise requires web server with php capabilities. You can either upload the given server side scripts to some server or you can use the server provided you by the instructor.

**Timing estimate**: 180 minutes

## 1. Server

The idea is to implement a simple common highscore for all the players. All the scores are stored in web-server. Ask the specific URL from the instructor.

The script is used with following interface http://www.webserver.com/save.php?name=*yourname*&score=*yourscore*&longitude=*logitudevalue*&latitude=*latitudevalue*

Example usage:

http://www.webserver.com/save.php?name=Jack&score=50&longitude=2.0&latitude=2.0

The longitude and latitude parameters are optional:

http://www.webserver.com/save.php?name=Jack&score=50

The showall.php scripts displays all the scores in a simple text file.

http://www.webserver.com/showall.php

Upload the scripts into a web server and test that you can store a score using the http interface. The given values are stored in xml-file:

http://www.webserver.com/highscore.xml

## 2. Client: Storing Name and Score

Now implement functionality when a new highscore is achieved, save the user name (from settings) and score to the web server using the given http interface. The connection is made in a different thread. The name and score is saved automatically if user chose this option from the settings.

* Implement class HttpConnection that inherites Thread
* run() - methods makes a connection to given url
* The url can be passed to the class in it's constructor
* When the connection is over the class can make a callback to it's host informing what was the result.

## 3. Client: Retrieving Highscores

Implement function where the user can retrieve all the highscores in a form. In the main screen, when user selects "High Scores", a form is set as current and the form contains all the highscores located in the server. The form contains a item for each score. Create your own custom item for the scores.

# Module 7: Mobile Media API

**Aim:** Learn how to implement sounds in MIDP using the Mobile Media API.

**Timing estimate**: 60 minutes

Add sound effects to your game. When the game starts, a background music (mid) is played. When rock is thrown and bird is hit, different sound effects are played. Also when Mr. Smith dies, play a sound effect. Use free sounds from the Internet (wav or midi) or use the sounds given you by the instructor.

Module 8: Location API

**Aim:** Learn how to get location in MIDP

**Timing estimate**: 60 minutes

When user hits a new high score, the score is sent to the web server (Exercise 6.2). Now implement functionality where the location of the user is sent to the server together with the high score info. Use Location API to retrieve the longitude and latitude of the user in a separate thread and send this information to the web server.

Module 9: Security and Publishing

**Aim:** Learn how to create a key pair and signing the MIDlet. Getting familiar with publishing applications in Ovi Store ([www.ovi.com](http://www.ovi.com)).

**Timing estimate**: 60 minutes

With NetBeans, create a trusted MIDlet of your application. Follow the instructions of the ide to go through this. Find out, which procedures you need to go through in order to publish your application in Nokia Ovi Store.

Extra Module: Touch and Type UI

**Aim:** Learn how to utilize Gesture API and Frame Animator API in applications ran on Touch and Type UI devices.

**Note:** Touch and Type device such as Nokia X3-02 is needed for conducting this exercise. The SDK version must also support touch framework e.g. Series 40 6th Edition SDK.

**Timing estimate**: 60 minutes

Implement a feature where the user can type the phone number and then send the top 3 of the high score list with name and score to the phone number specified.

## 1. Expanding the FrameAnimator Demo application

Download and open the FrameAnimator Demo in NetBeans. The application code can be found found from <http://www.forum.nokia.com/Develop/Java/Code_examples/>

1. Add an implementation, which changes the rectangle color to blue, when it is tapped and to red, when it is long tapped. The corresponding events are:

### GESTURE\_TAP

### GESTURE\_LONG\_PRESS

1. The application features dragging and kinetic scrolling movement in any given direction (FRAME\_ANIMATOR\_FREE\_ANGLE). Try out different friction options:

### FRAME\_ANIMATOR\_FRICTION\_LOW

### FRAME\_ANIMATOR\_FRICTION\_MEDIUM

### FRAME\_ANIMATOR\_FRICTION\_HIGH

## 2. Using Touch UI in the Game Application

1. Add a feature where you can drag Mr.Smith in X-direction on the ground
2. Use AnimatorFramework to animate Mr.Smith to the new location. Try out different animaton options to move him left/right.

Extra Module: Wireless Messaging

**Aim:** Learn how to utilize SMS messaging from your application. This exercise can be performed if you have a SIM card available (billable events).

**Timing estimate**: 90 minutes

Implement a feature where the user can type the phone number and then send the top 3 of the high score list with name and score to the phone number specified.