EXERCISES RELATED TO JAVA MIDLETS

Kari Laitinen
http://www.naturalprogramming.com

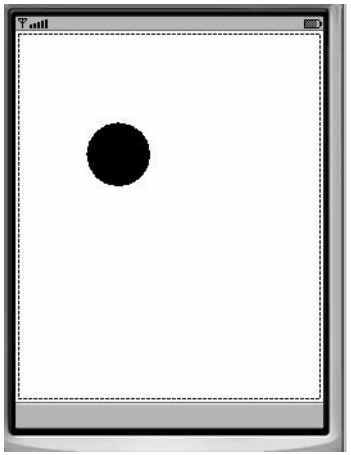
2008-06-02 File created.
2008-06-05 Last modification

- When you do midlet-related exercises, you can use JCreator as an editor. You can simply open the *MIDlet.java files to the editor and modify them. When you need to compile and execute a midlet, you must do these activities with the Sun Java Wireless Toolkit. Remember to save the file with JCreator before compilation.
- Do not create any projects when you work with JCreator.

Note that these exercises are written for the students that I teach.
Java midlets programming is not covered in "A Natural Introduction to Computer Programming with Java"

EXERCISES WITH PROGRAM KeyCodesMIDlet.java

EXERCISE 1: Modify the program so that, when it starts running, it draws a solid ball to the screen. This means that you can remove from the paint() method of class KeyCodesCanvas those method calls which draw strings. You should leave there those program lines that clear the screen and set the drawing color to black. A solid (or filled) ball can be drawn with the fillArc() method of class Graphics. The ball can be located on the screen in the following way.



EXERCISE 2: When the ball required in the previous exercise is visible on the screen, modify the program so that the ball disappears when key 0 (zero) is pressed. The ball must also reappear on the screen when the zero key is pressed again. You can achieve this by declaring the following new data field to the KeyCodesCanvas class

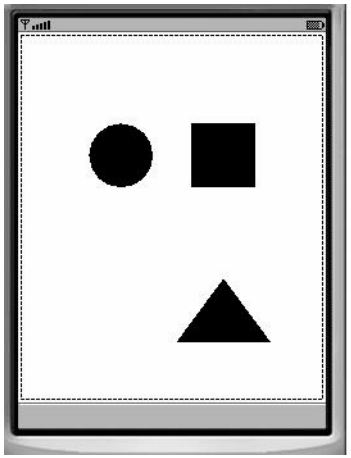
```
boolean  ball_must_be_drawn   =  true ;
```

The value of this boolean variable can then be examined in the keyPressed() method which is called automatically when a key is pressed. This variable should get value false if its current value is true, and vice versa. The keyPressed() method can begin in the following way

```
public void keyPressed( int key_code )
{
   if ( key_code  ==  '0' )
   {
      if ( ball_must_be_drawn  ==  true )
      {
         ball_must_be_drawn  =  false ;
         ...
```

You must check the value of ball_must_be_drawn inside the paint() method and decide whether or not the ball should be drawn.

EXERCISE 3: Continue developing this program so that you draw a filled square and a filled triangle to the screen. The Graphics class provides methods fillRect() and fillTriangle() to draw these graphical shapes. By studying the electronic documentation you can find out the parameters that these methods can be given. After having done this, the display should look like the following



EXERCISE 4: Improve the program so that also the square and the triangle can be made to disappear and re-appear by using a certain key of the keyboard. For instance, key 3 could be used to control the appearance of the triangle and key 4 could control the square. As in exercise 2, you can use the following kind of data fields to control the drawing of these graphical objects.

```
boolean  square_must_be_drawn    =  true ;
boolean  triangle_must_be_drawn  =  true ;
```

EXERCISES WITH PROGRAM MovingBallMIDlet.java

EXERCISE 1. Modify the appearance of the ball so that the ball is filled with the selected ball color and a black edge is drawn around the ball. To do this, you must change color in the paint() method.

EXERCISE 2. Modify the program so that white color can be selected from the list of possible colors. All the RGB values are in maximum (i.e., 0xFF or 255) when white color is specified. A white ball is visible on the screen when the previous exercise is done properly.

EXERCISE 3. Add a new color named "Random color" to the list of possible selectable colors. When the random color is selected, the program generates a random RGB value in the range 0 ... 0xFFFFFF, and sets this value as the current color. You have to add an if construct to the commandAction() method to check if random color was selected. See program RandomNumbersMIDlet.java to find out how random integers can be generated.

EXERCISE 4. Modify the program so that before the ball color selection list is displayed to the user, another list with which ball size can be selected is displayed. The user must thus first select ball size and then select ball color. The list with which ball size can be created with the following statements.

```
String[] selectable_ball_sizes  =
                              { "Small ball",
                                "Medium ball",
                                "Large ball",
```

```
                                    "Extra large ball" } ;

List ball_size_selection_list  =
                new List( "Select ball size:",
                          List.IMPLICIT,
                          selectable_ball_sizes,
                          null ) ;
```

The new List object needs to have a command listener in the same way as the old List. When the program reacts to the selections that the user makes, it must check which List is being shown. The second parameter of the commandAction() method tells what is the object that is currently being displayed. This parameter can be checked in the following way:

```
else if ( given_command  ==  List.SELECT_COMMAND )
{
   if ( display_content  ==  ball_size_selection_list )
   {
      ...
```

In the original version of the program, the ball diameter is a constant (40 pixels). As this new feature allows the ball diameter to change, you need a new data field, such as

```
int  ball_diameter  =  40 ;
```

whose value is changed when a new ball size is set. You can yourself decide what are the actual possible sizes of the ball.

EXERCISE 5. Improve the program so that an alarm is given when the user moves the ball so that it almost disappears from the screen. The alarm (or alert) can be given, for instance, when the center point of the ball is outside the canvas area.

You can alert the user by using the standard Alert class. By studying the last part of program AlertGaugeDemoMIDlet.java, you can find out how an Alert object can be created, and how it is shown to the user. An Alert is such that it is possible to specify how long it will be shown to the user. The old display content is automatically re-displayed after an Alert has been shown for the specified time.

EXERCISES WITH PROGRAM ClockMIDlet.java

EXERCISE 1. Modify the program first so that the seconds hand of the clock will be drawn as a dotted line. The Graphics class provides a method named setStrokeStyle() with which a dotted drawing style can be enabled. The Graphics class contains a static constant named DOTTED that can be passed as a parameter for the setStrokeStyle() method. Only a single line of code is required to implement this feature.

EXERCISE 2. Create a Settings menu for the ClockCanvas class. You can study the RandomNumbersMIDlet.java program to find out how a Settings menu can be created. First the program should have the possibility to set either a white or black clock background. (If the clock background is black, the clock should be drawn with white color.) You must use the Form class to build the Settings menu. To the Form object you should attach a ChoiceGroup object that contains exclusive choices for white and black clock background. A ChoiceGroup can be created in the following way.

```
ChoiceGroup  clock_background_choice_group  =
    new  ChoiceGroup( "CLOCK BACKGROUND:",
                      Choice.EXCLUSIVE,
                      clock_background_choices, null );
```

The needed Form object can be created in the following way:

```
Form  settings_form  =  new  Form( "SETTINGS" ) ;
```

To activate and deactivate the Settings menu, you need a couple of Command objects in the ClockCanvas class. These objects can be created in the following way.

```
Command settings_command  =
      new  Command( "Settings", Command.SCREEN, 1 );
Command done_command      =
      new  Command( "Done", Command.SCREEN, 1 ) ;
```

The ClockCanvas class must implement the CommandListener interface, and it must contain a method named commandAction() which reacts to these commands.

Because the ClockMIDlet class, the master class of ClockCanvas, already contains a CommandAction() method that reacts to commands, it will not be easy to make the same method work in the ClockCanvas class. One possibility to solve this problem is to disable the commandAction() method of the ClockMIDlet class. This can be achieved by not setting a CommandListener in the ClockMIDlet class. You can thus comment out one statement of the startApp() method of the ClockMIDlet ClockMIDlet class:

```
//      clock_canvas.setCommandListener( this ) ;
```

After this operation the Exit command will not work, but this does not matter for the time being.

You can proceed with this exercise so that you first make the Settings menu appear and disappear when the commands are given. After this you can make the clock behave correctly according to the specified settings.

In order to make a white or black clock backgrund according to what has been selected in the corresponding ChoiceGroup, your program need to read the state of the setting in the paint() method of the ClockCanvas class. The state of the setting can be inspected with method isSelected() in the following way.

```
if ( clock_background_choice_group.isSelected( 0 ) )
{
   //  ChoiceGroup first selection has been made
```

EXERCISE 3. In this exercise your task is to improve the program so that the clock becomes an alarm clock. To implement an alarm clock, you need to be able to set the time when alarm will be given (the wake-up time), and also you need to activate and deactivate the alarm. To set the alarm time, you can use a standard class named DateField. A DateField object can be attached to a Form, so that it will be available in the Settings menu that was constructed in the previous exercise. A suitable DateField object can be created in the following way.

```
DateField  wakeup_time_field  =
           new DateField( "WAKE-UP TIME:",
                          DateField.DATE_TIME ) ;
```

This statement will create a DateField so that both time and date can be set. To make the DateField usable, you have to intitialize and append it to the Form with the following statements in the ClockCanvas constructor:

```
Calendar default_wakeup_time  =
                    Calendar.getInstance() ;
wakeup_time_field.setDate(
                    default_wakeup_time.getTime() ) ;
settings_form.append( wakeup_time_field ) ;
```

Before proceeding, ensure that the DateField is available in the Settings Form.

To activate the alarm, you can add a new ChoiceGroup to the Settings Form. This can be done with the following statements.

```
String[]  alarm_activated_choice  =
                    { "Alarm activated" } ;

ChoiceGroup  alarm_activation_choice_group  =
       new  ChoiceGroup( "ALARM ACTIVATION:",
                          Choice.MULTIPLE,
                         alarm_activated_choice, null ) ;
```

In this ChoiceGroup there is only one choice that can be enabled or disabled. The intention is that the user must activate the alarm with this ChoiceGroup after he or she has set the alarm time with the DateField.

You can control the alarming process with a new if construct at the end of the paint() method. You should first check whether or not the alarm is activated. If the alarm is activated, you should check whether it is time to give the alarm. The required if construct can begin in the following way.

```
if ( alarm_activation_choice_group.isSelected( 0 ) )
{
   Calendar wakeup_time  =  Calendar.getInstance() ;

   wakeup_time.setTime( wakeup_time_field.getDate() ) ;

   int wakeup_hour   = wakeup_time.get( Calendar.HOUR_OF_DAY ) ;
   int wakeup_minute = wakeup_time.get( Calendar.MINUTE ) ;

   if ( ...
```

The above program lines show how you can read the requested wake-up time from the DateField. It is somewhat complicated because the method getDate() of class DateField returns a Date object which we convert to a Calendar object in order to access individual pieces of time information.

The program should print "Alarm activated" to the display when the alarm is activated but the wake-up time has not yet been reached. It should print something like "WOULD YOU, PLEASE, WAKE UP." when the wake-up time has been reached. The alarming process will be accurate enough if you compare current hours and minutes to the set wake-up hour and minute.

Note that this is a visual alarm clock. The sleeper needs to constantly watch the clock to know when it is time to wake up.

EXERCISE 4. Improve the program so that the alarm can be deactivated by pressing any numerical key.

EXERCISE 5. See program MovingBallMIDlet.java to find out how you can make the Exit command work again.

EXERCISE 6. Find out how you could make the text "WOULD YOU, PLEASE, WAKE UP." blink or how you could make the phone shake while it is alarming.