Navegando entre telas em Java ME

Introdução

Este trecho de código demonstra como navegar entre três diferentes instâncias da classe `Screen`. `Screen` é a superclasse para todas as classes de interface gráfica de usuário de alto nível, como `Form` e `List`.

Na prática, a navegação é realizada definindo o `Displayable` corrente para o `Screen` especificado:

```
private void switchCurrentScreen(Displayable displayable) {
    Display.getDisplay(this).setCurrent(displayable);
}
```

Código fonte

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.StringItem;
import javax.microedition.midlet.MIDlet;
```

```
public class NavigationMIDlet extends MIDlet implements CommandListener {
    private Form firstScreen;
    private Form secondScreen;
    private Form thirdScreen;

    // Comandos de navegação
    private static final Command NEXT_COMMAND =
        new Command("Next", Command.SCREEN, 1);
    private static final Command PREV_COMMAND =
        new Command("Previous", Command.BACK, 1);
    private static final Command EXIT_COMMAND =
        new Command("Exit", Command.EXIT, 1);;

    /**
     * Construtor. Constrói o objeto e inicializa os objetos displayables.
     */
    public NavigationMIDlet() {
        // Initialize the screens
        initFirstScreen();
        initSecondScreen();
        initThirdScreen();
    }

    private void initFirstScreen() {
        firstScreen = new Form("First example screen");
        StringItem item = new StringItem("Description",
            "This is the first example screen.");
        firstScreen.append(item);
        firstScreen.addCommand(NEXT_COMMAND);
        firstScreen.addCommand(EXIT_COMMAND);
```


(C) Copyright Nokia 2014. All rights reserved.

```
        firstScreen.setCommandListener(this);
    }

    private void initSecondScreen() {
        secondScreen = new Form("Second example screen");
        StringItem item = new StringItem("Description",
            "This is the second example screen.");
        secondScreen.append(item);
        secondScreen.addCommand(NEXT_COMMAND);
        secondScreen.addCommand(PREV_COMMAND);
        secondScreen.setCommandListener(this);
    }

    private void initThirdScreen() {
        thirdScreen = new Form("Third example screen");
        StringItem item = new StringItem("Description",
            "This is the third example screen.");
        thirdScreen.append(item);
        thirdScreen.addCommand(PREV_COMMAND);
        thirdScreen.setCommandListener(this);
    }

    private void switchCurrentScreen(Displayable displayable) {
        Display.getDisplay(this).setCurrent(displayable);
    }

    /**
     * Método pertencente a MIDlet.
     * Invocado quando a MIDlet é inicializada
     */
    public void startApp() {
        // A tela inicial é o primeiro componente de tela Form
        switchCurrentScreen(firstScreen);
    }

    /**
     * Método pertencente a MIDlet.
     * Invocado pelo MIDlet para entrar no estado de Pause
     */
    public void pauseApp() {
        // Nenhuma implementação necessária
    }

    /**
     * Método pertencente a MIDlet.
     * Invocado quando a MIDlet é finalizada
     * @param unconditional se a MIDlet pode ser finalizada
     */
    public void destroyApp(boolean unconditional) {
        // Nenhuma implementação necessária
    }

    /**
     * Método da interface CommandListener.
     * Invocado pelo sistema para indicar o disparo de um evento de um comando
pertencente a um displayable em particular
```


(C) Copyright Nokia 2014. All rights reserved.

```
     * @param command referência a um objeto Command que foi invocado
     * @param displayable referência ao displayable ao qual o objeto Command pertence
     */
    public void commandAction(Command command, Displayable displayable) {
        if (command == EXIT_COMMAND) {
            // Sai da MIDlet
            notifyDestroyed();
        } else if (command == NEXT_COMMAND) {
            if (displayable == firstScreen) {
                switchCurrentScreen(secondScreen);
            } else if (displayable == secondScreen) {
                switchCurrentScreen(thirdScreen);
            }
        } else if (command == PREV_COMMAND) {
            if (displayable == secondScreen) {
                switchCurrentScreen(firstScreen);
            } else if (displayable == thirdScreen) {
                switchCurrentScreen(secondScreen);
            }
        }
    }
}
```

É possível também exebir alertas (instâncias da classe `Alert`) antes de efetuar a troca das telas. Adicionaalmente, a troca pode ser feita apenas após o usuário ter descartado o alerta. A classe `AlertType` define o tipo do alerta e se a interação do usuário é necessária antes da tela ser alterada. (`AlertType.CONFIRMATION`). Para implementar a exibição de um alerta informativo (`AlertType.INFO`; não é necessária qualquer interação do usuário) antes de ser realizada a troca das telas. Adicione o trecho de código a seguir ao exemplo anterior:

```
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
```

```
private void switchCurrentScreen(Alert alert, Displayable displayable) {
    Display.getDisplay(this).setCurrent(alert, displayable);
}
```

```
Alert alert = new Alert("Screen changing",
    "Switching current screen in a moment...", null, AlertType.INFO);
switchCurrentScreen(alert, firstScreen);
```

Pós-condições

A MIDlet demonstra a navegação entre diferentes telas.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.