Programowanie na komórki

J2ME, MIDP 2.0

Adam Sawicki
http://regedit.gamedev.pl/
sawickiap@poczta.onet.pl
Grudzień 2007

W prezentacji wykorzystane są fragmenty dokumentacji MIDP 2.1 firmy Sun.

Programowanie na komórki

- W czym pisać programy na komórki?
  - To zależy od klasy telefonu.

(nic)

Java

Java, C++ i inne...

Agenda

- Wprowadzenie do J2ME
- Midlet
- GUI
- Wejście-wyjście
- Internet
- Grafika 2D
- Programowanie gier 2D
- Kilka drobiazgów
- Baza danych
- Multimedia
- Rozszerzenia

Wprowadzenie do J2ME

J2ME
Wireless Toolkit
Język Java w J2ME

Co to jest J2ME?

- Skróty, dużo skrótów...
  - J2ME – Java 2 Platform, Micro Edition
    - Platforma Java na komórki (inne to J2SE i J2EE)
  - CLDC – Connected Limited Device Configuration
  - MIDP – Mobile Information Device Profile
  - JSR – Java Specification Request
    - Specyfikacje dodatków (np. JSR-184 to M3G)
- W praktyce używamy dokumentacji MIDP, ona zawiera też dokumentację CLDC
  - MIDP 1.0 – CLCD 1.0, MIDP 2.0 – CLDC 1.1

Jakie jest J2ME?

- Ten sam język Java
  - Używamy tego samego kompilatora javac
- Zupełnie nowa biblioteka standardowa
  - Podajemy niestandardowy classpath
- Liczne ograniczenia sprzętu
  - Mało pamięci RAM
  - Wolny procesor
  - Niska rozdzielczość wyświetlacza, mało kolorów
  - Kiepska klawiatura
  - Duże zróżnicowanie klawiatur, ekranów, wydajności

Co jest potrzebne?

- JDK – Java Development Kit
  - Zawiera w sobie JRE - Java Runtime Environment
- Java Wireless Toolkit
- Edytor lub IDE
  - jEdit, Notatnik, Vim, Emacs, ...
  - NetBeans, Eclipse, ...

Wszystko dostępne za darmo (java.sun.com).

Hello World!

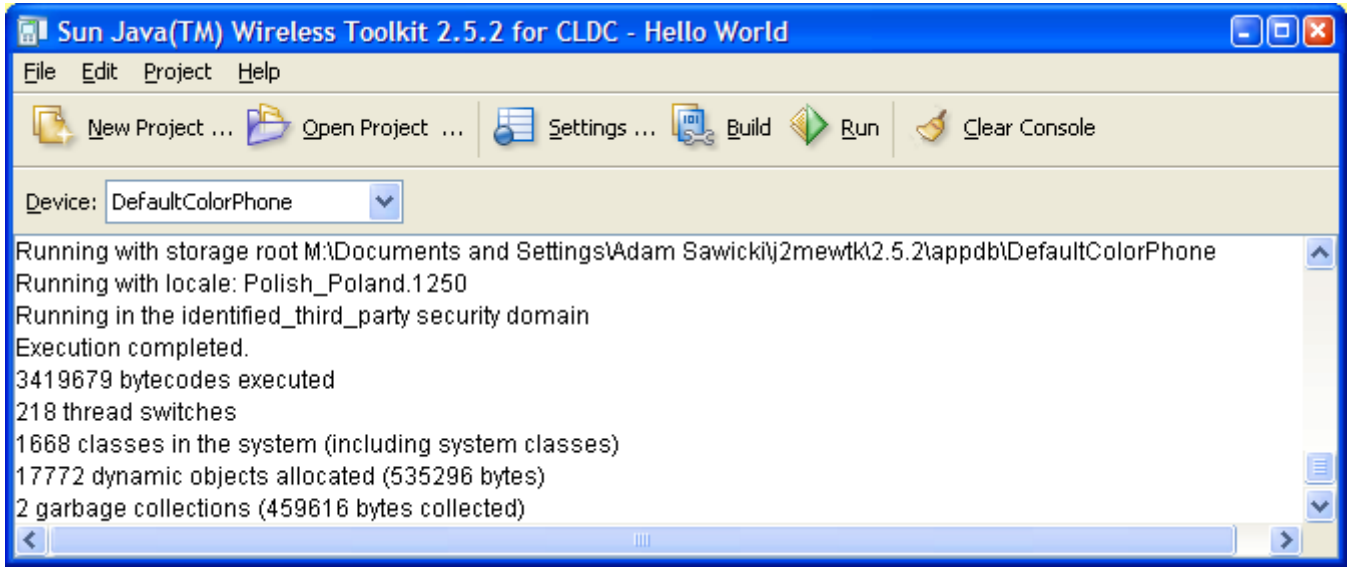
```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloWorld
  extends MIDlet
  implements CommandListener
{
  public HelloWorld()
  {
  }

  //////
  // Implementacja MIDlet

  protected void startApp()
    throws MIDletStateChangeException
  {
    Alert alert = new Alert("Hello World!");
    alert.setCommandListener(this);
    alert.addCommand(new Command("Koniec", Command.EXIT, 0));

    Display display = Display.getDisplay(this);
    display.setCurrent(alert);
  }

  protected void pauseApp()
  {
  }

  protected void destroyApp(boolean unconditional)
    throws MIDletStateChangeException
  {
  }

  //////
  // Implementacja CommandListener

  public void commandAction(Command c, Displayable d)
  {
    if (c.getCommandType() == Command.EXIT)
      notifyDestroyed();
  }
}
```

Ręczne budowanie midletu

- Midlet można zbudować ręcznie z wiersza poleceń
  - Kompilacja: javac (JDK)
  - Preweryfikacja: preverify (WTK)
  - Napisanie pliku MANIFEST.MF
  - Utworzenie paczki: jar (JDK)
  - Napisanie pliku JAD
  - Emulacja: emulator (WTK)
  - Wysłanie na komórkę
- ...ale nie trzeba, można prościej!

Wireless Toolkit

- Sun Java(TM) Wireless Toolkit for CLDC
  - Tytułowy program pakietu Wireless Toolkit
  - Proste IDE do midletów



Wireless Toolkit

- Operuje na projektach
  - Przechowuje je w Documents and Settings\LOGIN\j2mewtk\apps\NAZWA
  - Zakłada kilka katalogów
    - src – pliki JAVA
    - res – obrazki, dźwięki i inne zasoby
    - lib – dodatkowe biblioteki
    - classes, tmpclasses – pliki tymczasowe
    - bin – pliki wynikowe JAR, JAD
- Polecenia
  - Kompilacja: Build
  - Emulacja: Run
  - Utworzenie paczki: Create Package
- Nie posiada edytora!

Język Java

- Dostępny mamy normalny język Java
  - Klasy, wyjątki
  - byte, short, int, long, boolean, char itd.
  - String, StringBuffer
  - float, double – od wersji MIDP 2.0
  - NIE MA generics, enum itp. – to jest Java 1.2

Wielowątkowość

- J2ME to platforma ograniczona, ale nowoczesna.
  - Można, a nawet trzeba pisać wielowątkowo i zajmować się synchronizacją.
- JEST dostępna wielowątkowość:
  - Słowo kluczowe synchronized
  - Metody klasy bazowej Object: wait, notify, notifyAll
  - Klasa Thread, interfejs Runnable

Biblioteka standardowa

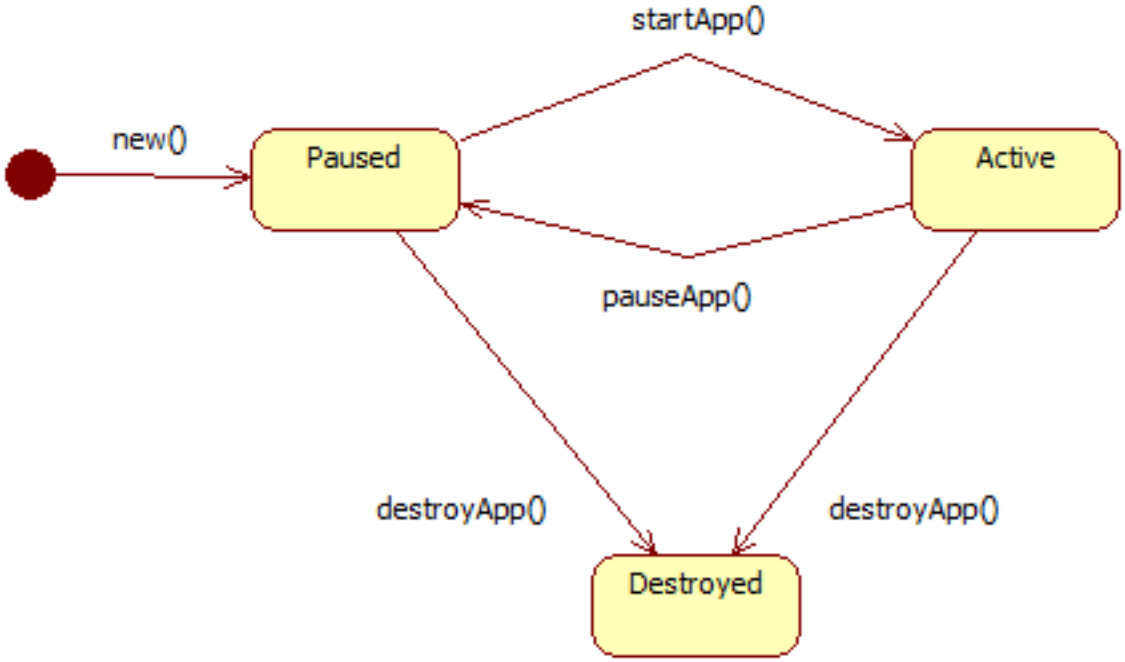
- Podstawowe typy
  - java.lang – Boolean, Character, Integer itd.
  - java.lang – String, StringBuffer
- Matematyka
  - java.lang.Math – abs, max, sin, floor, sqrt, ...
    - BRAKUJE exp, log, pow, atan2
  - java.util.Random – generator liczb pseudolosowych
- Struktury danych
  - java.util – Vector, Stack, Hashtable

Midlet

javax.microedition.midlet

Midlet

- Midlet – program na komórkę w J2ME
  - Analogicznie do: aplet, servlet
- NIE MA kolejki komunikatów ani jawnej pętli jak w Windows API
- JEST sterowany zdarzeniami, jak programy w Delphi czy C#
- Posiada **STAN**



Midlet – szkielet klasy

```
import javax.microedition.midlet.*;

public class HelloWorld extends MIDlet
{
  public HelloWorld()
  {
  }

  protected void startApp()
    throws MIDletStateChangeException
  {
  }

  protected void pauseApp()
  {
  }

  protected void destroyApp(boolean unconditional)
    throws MIDletStateChangeException
  {
  }
}
```

Midlet - zdarzenia

- **startApp()**
  - Program staje się aktywny
  - Dokonać inicjalizacji, wczytać zasoby, pokazać coś na ekranie
  - Jeśli nie udaje się uaktywnić, rzucić wyjątek MIDletStateChangeException
  - Jeśli nieodwracalny błąd, wywołać notifyDestroyed
- **pauseApp()**
  - Program staje się nieaktywny
  - Zwolnić wszystkie zasoby zajmujące dużo procesora lub pamięci
- **destroyApp(boolean unconditional)**
  - Program zostaje wyłączony
  - Zapisać trwałe dane
  - Jeśli odmawia wyłączenia, rzucić wyjątek MIDletStateChangeException (tylko jeśli unconditional == false)

Midlet – zmiana stanu

- **notifyDestroyed()**
  - Wywołać jeśli program chce się zakończyć
  - destroyApp nie zostanie już wywołane!
- **notifyPaused()**
  - Wywołać jeśli program chce się stać nieaktywny
- **resumeRequest()**
  - Wywołać jeśli program chce się stać aktywny

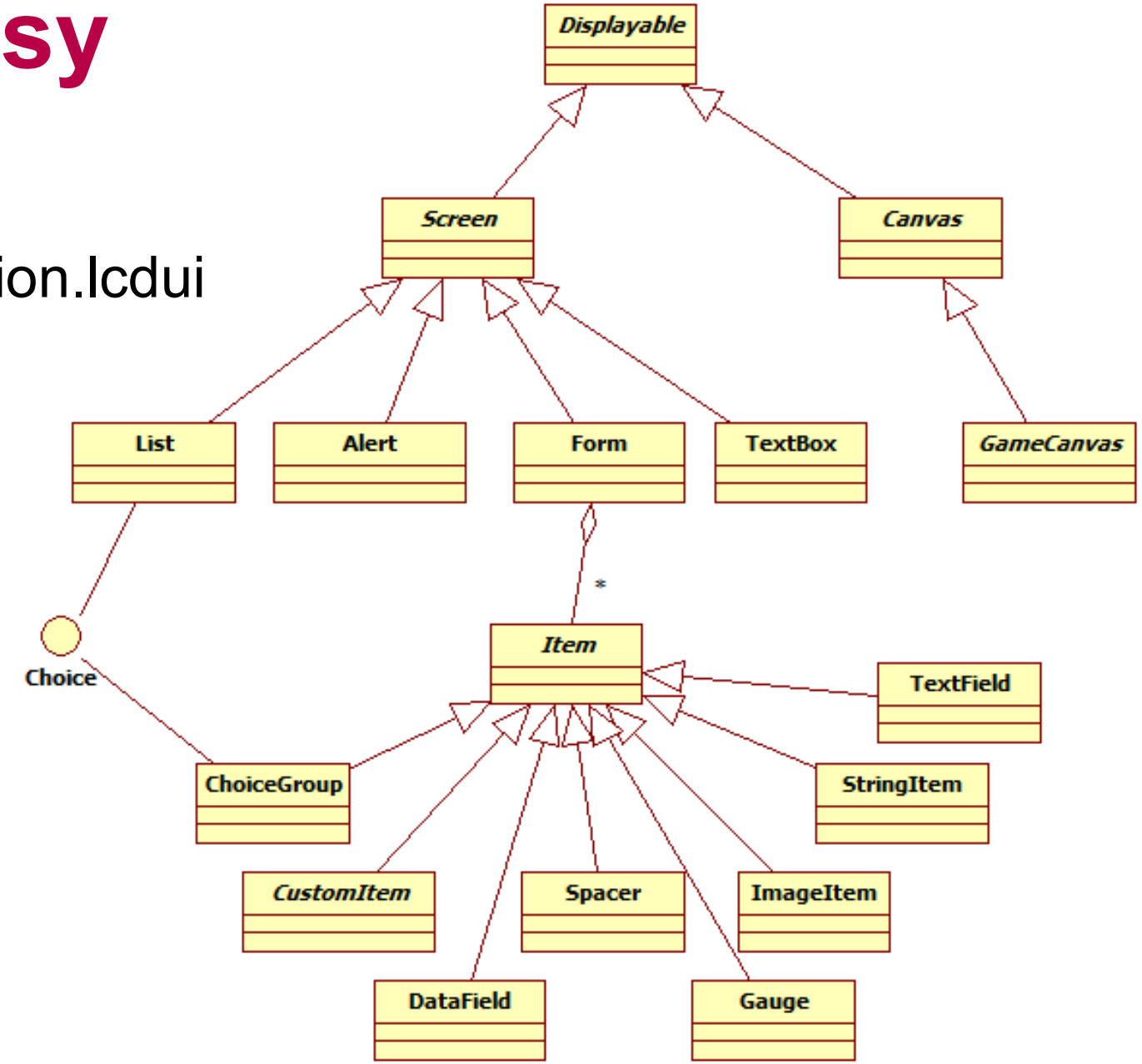
- **platformRequest(String URL)**
  - Wywołać aby system obsłużył podany URL, np. tel:NUMER

GUI

javax.microedition.lcdui

GUI - Klasy

Pakiet:
javax.microedition.lcdui



Wysoki i niski poziom

- API wysokiego poziomu
  - Zawiera: kontrolki GUI i polecenia
  - Dla: aplikacji biznesowych
  - Abstrakcyjne, niezależne od rozdzielczości
  - Rysowane wg stylu danego telefonu
- API niskiego poziomu
  - Zawiera: możliwość rysowania i reakcji na wejście z klawiatury
  - Dla: gier
  - Zależne od wyświetlacza i klawiatury danego telefonu
  - Wygląd i obsługa implementowane samodzielnie

Alert

- Alert – ekran z komunikatem tekstowym
  - Coś jak znany z Windowsa MessageBox :)

```
// Utwórz ekran komunikatu
Alert alert = new Alert(
  "Błąd",                          // title
  "Nie można otworzyć pliku.", // alertText
  null,                            // alertImage
  AlertType.ERROR);                // alertType
// Pobierz singleton wyświetlacza
Display display = Display.getDisplay(this);
// Pokaż komunikat na wyświetlaczu
display.setCurrent(alert);
```

Command i CommandListener

- Zaimplementować w jakiejś klasie L interfejs CommandListener
- Utworzyć jakiś ekran E typu pochodnego od Displayable
- Wywołać:
```
E.setCommandListener(L_obj);
```
- Wywołać dowolną liczbę razy:
```
E.addCommand(new Command(label, commandType, priority));
```
- Pokazać E na wyświetlaczu:
```
display.setCurrent(E);
```
- Czekać na zdarzenia...

Command i CommandListener

```
public class TestMidlet extends MIDlet implements CommandListener
{
  protected void startApp() throws MIDletStateChangeException
  {
    Alert alert = new Alert("Wyjście", "Czy na pewno chcesz wyjść?",
      null, null);

    alert.setCommandListener(this);
    alert.addCommand(new Command("Zakończ", Command.EXIT, 0));
    alert.addCommand(new Command("Wróć", Command.BACK, 0));

    Display display = Display.getDisplay(this);
    display.setCurrent(alert);
  }

  public void commandAction(Command c, Displayable d)
  {
    if (c.getCommandType() == Command.EXIT)
      notifyDestroyed();
    else if (c.getCommandType() == Command.BACK)
      Display.getDisplay(this).setCurrent(m_OtherScreen);
  }
```

Command

```
Command(String label, int commandType, int priority)
```

- **label** – nazwa
- **commandType** – semantyka (znaczenie)
  Stałe: Command.BACK, CANCEL, EXIT, HELP, OK, SCREEN, STOP, do pozostałych ITEM
- **priority** – określa kolejność

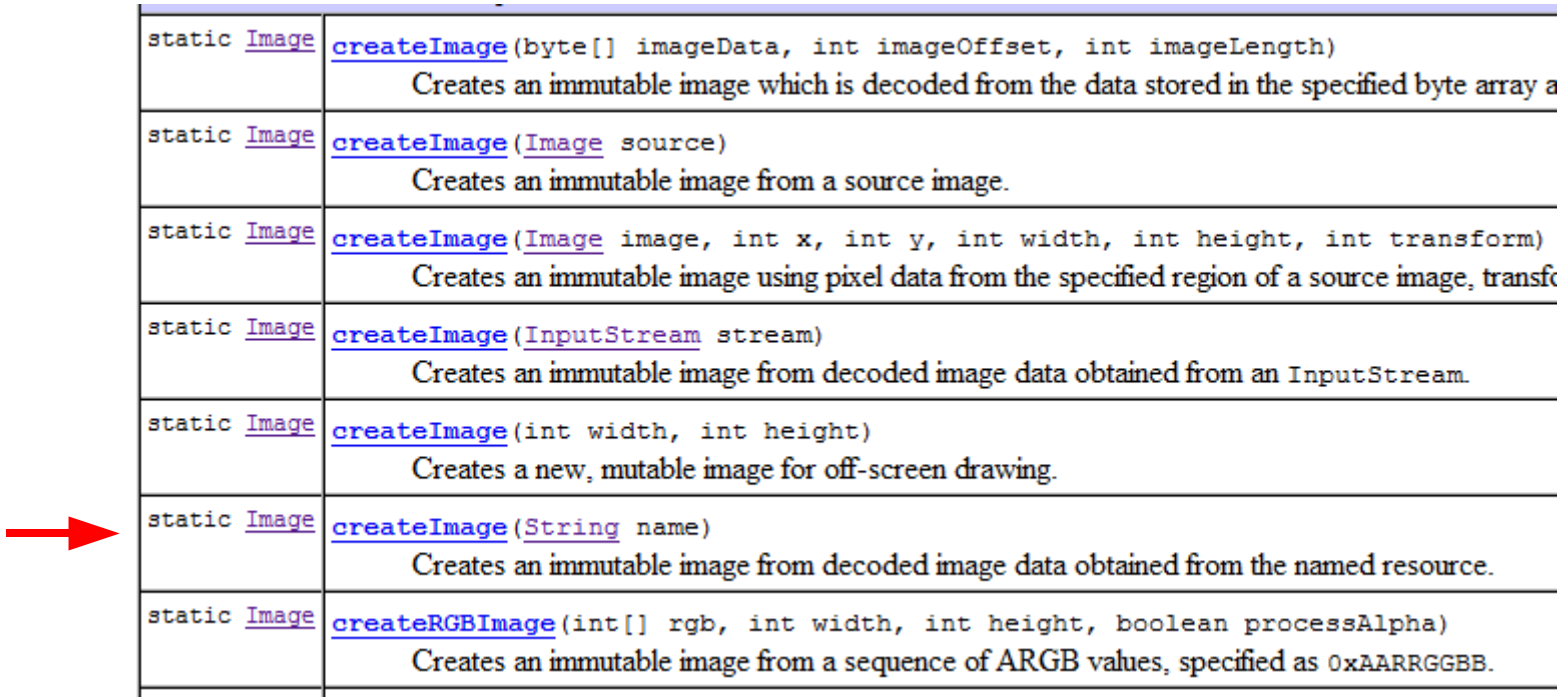
Command

- Telefon sam określa rozmieszczenie i sposób uruchamiania poleceń

- Kiedy poleceń jest dużo, zapewnia menu

Obrazek

- Obrazek w pamięci reprezentuje klasa Image
- Obrazek może być Mutable lub Immutable
- Jedyny na pewno wspierany format to PNG



static Image	createImage(byte[] imageData, int imageOffset, int imageLength) Creates an immutable image which is decoded from the data stored in the specified byte array a
static Image	createImage(Image source) Creates an immutable image from a source image.
static Image	createImage(Image image, int x, int y, int width, int height, int transform) Creates an immutable image using pixel data from the specified region of a source image, transfo
static Image	createImage(InputStream stream) Creates an immutable image from decoded image data obtained from an InputStream.
static Image	createImage(int width, int height) Creates a new, mutable image for off-screen drawing.
static Image	createImage(String name) Creates an immutable image from decoded image data obtained from the named resource.
static Image	createRGBImage(int[] rgb, int width, int height, boolean processAlpha) Creates an immutable image from a sequence of ARGB values, specified as 0xAARRGGBB.

Obrazek - wykorzystanie

```
try
{
  Image image = Image.createImage("/Ikonka.png");
  Alert alert = new Alert(
    "Wyjście",
    "Czy na pewno chcesz wyjść?",
    image,
    null);

  alert.setCommandListener(this);
  alert.addCommand(new Command("Zakończ", Command.EXIT, 0));
  alert.addCommand(new Command("Wróć", Command.BACK, 0));

  Display display = Display.getDisplay(this);
  display.setCurrent(alert);
}
catch (java.io.IOException e)
{
  e.printStackTrace();
}
```

Plik w podkatalogu res/

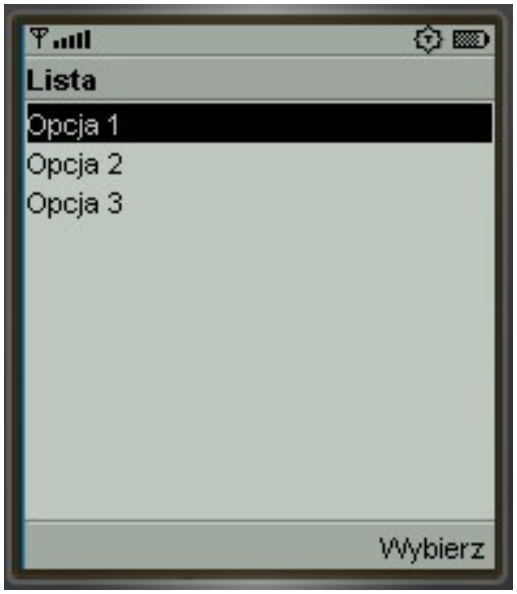
Lista

```
List m_List;

protected void startApp() throws MIDletStateChangeException
{
  m_List = new List("Lista", Choice.EXCLUSIVE);
  m_List.append("Opcja 1", null);
  m_List.append("Opcja 2", null);
  m_List.append("Opcja 3", null);

  m_List.setCommandListener(this);
  m_List.addCommand(new Command("Zakończ", Command.EXIT, 0));
  m_List.addCommand(new Command("OK", Command.OK, 0));

  Display display = Display.getDisplay(this);
  display.setCurrent(m_List);
}

public void commandAction(Command c, Displayable d)
{
  if (c.getCommandType() == Command.EXIT)
    notifyDestroyed();
  else if (c.getCommandType() == Command.OK)
    GoFurther(m_List.getSelectedIndex());
}
```

Lista wielokrotnego wyboru

```
List m_List;

protected void startApp() throws MIDletStateChangeException
{
  m_List = new List("Lista", Choice.MULTIPLE);
  m_List.append("Opcja 1", null);
  m_List.append("Opcja 2", null);
  m_List.append("Opcja 3", null);

  m_List.setCommandListener(this);
  m_List.addCommand(new Command("Zakończ", Command.EXIT, 0));
  m_List.addCommand(new Command("OK", Command.OK, 0));

  Display display = Display.getDisplay(this);
  display.setCurrent(m_List);
}

public void commandAction(Command c, Displayable d)
{
  if (c.getCommandType() == Command.EXIT)
    notifyDestroyed();
  else if (c.getCommandType() == Command.OK)
    GoFurther(m_List.isSelected(0), m_List.isSelected(1), m_List.isSelected(2));
}
```

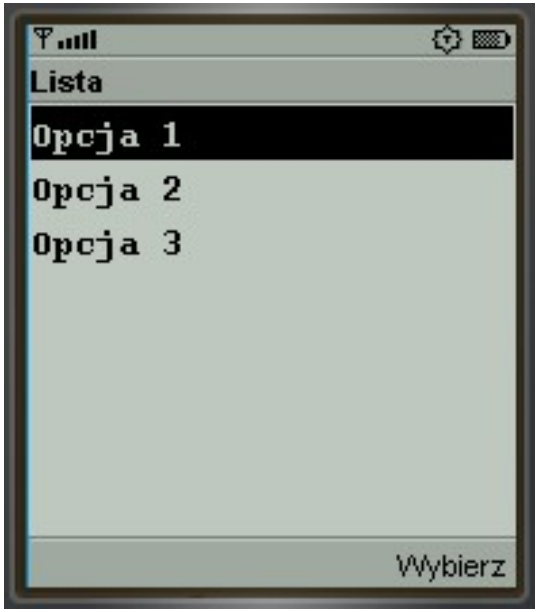
Lista jako menu




```
List m_List;

protected void startApp() throws MIDletStateChangeException
{
  m_List = new List("Lista", Choice.IMPLICIT);
  m_List.append("Opcja 1", null);
  m_List.append("Opcja 2", null);
  m_List.append("Opcja 3", null);

  m_List.setCommandListener(this);
  m_List.setSelectCommand(new Command("Wybierz", Command.ITEM, 0));

  Display display = Display.getDisplay(this);
  display.setCurrent(m_List);
}

public void commandAction(Command c, Displayable d)
{
  if (c.getCommandType() == Command.ITEM)
    GoFurther(m_List.getSelectedIndex());
}
```

Czcionka

```
Font font = Font.getFont(
   Font.FACE_MONOSPACE,
   Font.STYLE_BOLD,
   Font.SIZE_LARGE);

m_List = new List("Lista", Choice.IMPLICIT);
m_List.append("Opcja 1", null);
m_List.append("Opcja 2", null);
m_List.append("Opcja 3", null);
m_List.setFont(0, font);
m_List.setFont(1, font);
m_List.setFont(2, font);
```



Czcionka

```
static Font | getFont(int face, int style, int size)
```

- **face**:
  FACE_MONOSPACE
  FACE_PROPORTIONAL
  FACE_SYSTEM
- **style**:
  STYLE_BOLD
  STYLE_ITALIC
  STYLE_PLAIN
  STYLE_UNDERLINED
- **size**:
  SIZE_LARGE
  SIZE_MEDIUM
  SIZE_SMALL
- Nie zawsze dostajemy pożądane, różniące się czcionki!

TextBox

```
TextBox m_TextBox;

protected void startApp() throws MIDletStateChangeException
{
  m_TextBox = new TextBox("Login", "", 32, 0);

  m_TextBox.setCommandListener(this);
  m_TextBox.addCommand(new Command("OK", Command.OK, 0));

  Display display = Display.getDisplay(this);
  display.setCurrent(m_TextBox);
}

public void commandAction(Command c, Displayable d)
{
  if (c.getCommandType() == Command.OK)
    GoFurther(m_TextBox.getString());
}
```

TextBox - Constraints

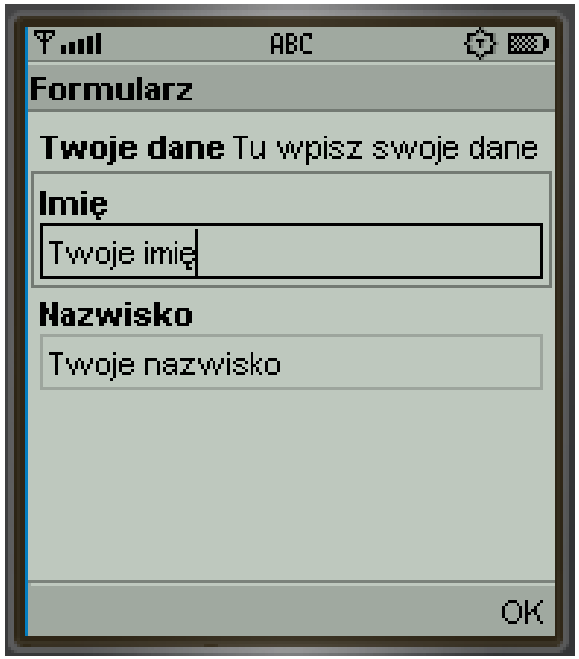
```
TextBox(String title, String text, int maxSize, int constraints)
```

- **Ograniczenia:**
  TextField.ANY
  TextField.EMAILADDR
  TextField.NUMERIC
  TextField.PHONENUMBER
  TextField.URL
  TextField.DECIMAL

- **Modyfikatory:**
  TextField.PASSWORD
  TextField.UNEDITABLE
  TextField.SENSITIVE
  TextField.NON_PREDICTIVE
  TextField.INITIAL_CAPS_WORD
  TextField.INITIAL_CAPS_SENTENCE

Formularz

```
m_Form = new Form("Formularz");

Item item;
item = new StringItem("Twoje dane", "Tu wpisz swoje dane", 0);
item.setLayout(Item.LAYOUT_2 | Item.LAYOUT_NEWLINE_AFTER);
m_Form.append(item);
item = new TextField("Imię", "Twoje imię", 32, 0);
item.setLayout(Item.LAYOUT_2 | Item.LAYOUT_NEWLINE_AFTER);
m_Form.append(item);
item = new TextField("Nazwisko", "Twoje nazwisko", 32, 0);
item.setLayout(Item.LAYOUT_2 | Item.LAYOUT_NEWLINE_AFTER);
m_Form.append(item);

m_Form.setCommandListener(this);
m_Form.addCommand(new Command("OK", Command.OK, 0));

Display display = Display.getDisplay(this);
display.setCurrent(m_Form);
```



- Kilka różnych kontrolek na jednym ekranie.

Kontrolki statyczne


```
m_Form = new Form("Formularz");

Image image;
try
{
  image = Image.createImage("/Ikonka2.png");
}
catch (java.io.IOException e)
{
  e.printStackTrace();
  image = null;
}

m_Form.append(new StringItem(null, "Napis...", Item.PLAIN));
m_Form.append(new Spacer(32, 32));
m_Form.append(new ImageItem(null, image, ImageItem.LAYOUT_DEFAULT, null));

m_Form.setCommandListener(this);
m_Form.addCommand(new Command("OK", Command.OK, 0));

Display display = Display.getDisplay(this);
display.setCurrent(m_Form);
```

TextField

- Pole do wprowadzania tekstu
- Flagi takie same jak dla TextBox

```
m_Form.append(new TextField(
  "URL",
  "http://www.google.pl/",
  256,
  TextField.URL));
```

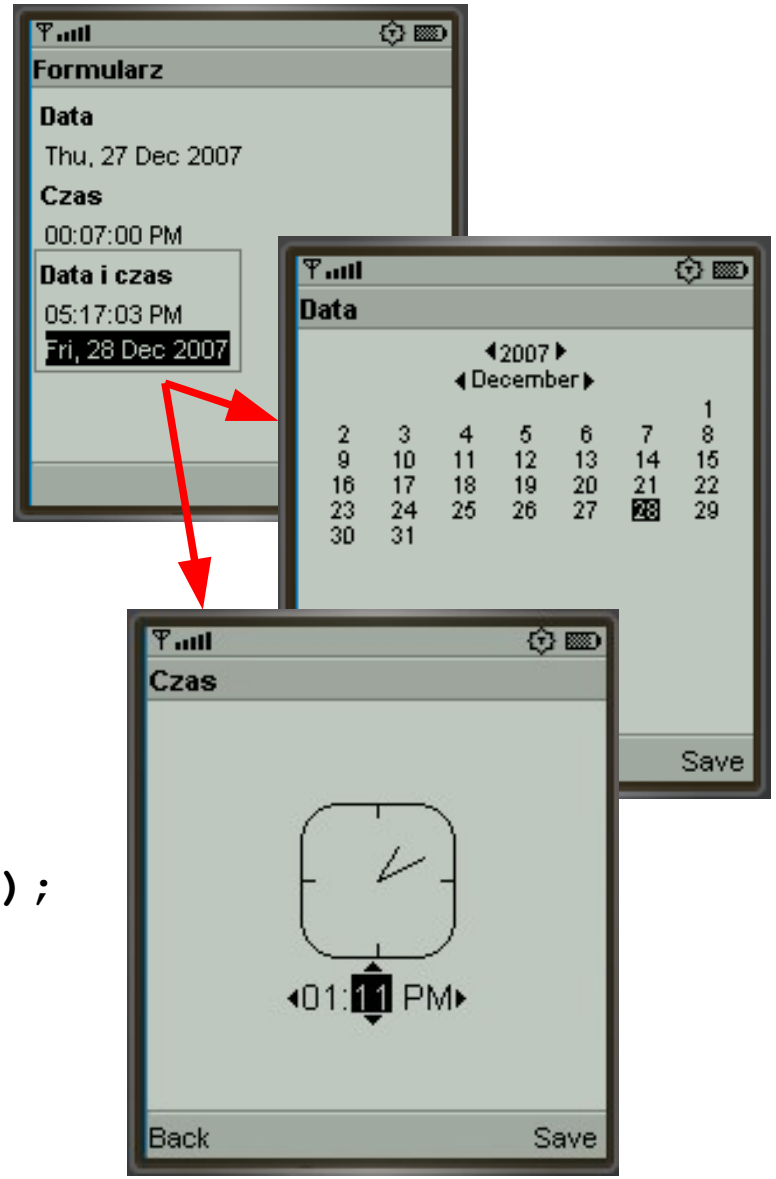
Gauge

- Suwak do wybierania wartości liczbowej

```
m_Form.append(new Gauge(
  "Głośność", // label
  true,       // interactive
  9,          // maxValue
  5));        // initialValue
```

DateTime

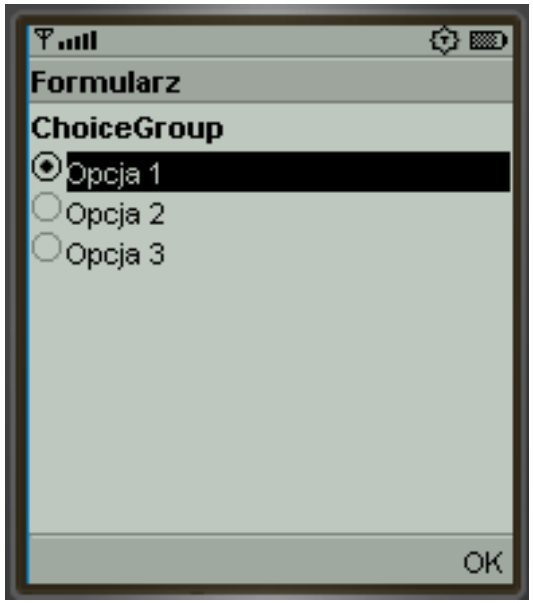
- Wybór daty i/lub czasu




```
m_Form.append(new DateField(
  "Data", DateField.DATE));
m_Form.append(new DateField(
  "Czas", DateField.TIME));
m_Form.append(new DateField(
  "Data i czas", DateField.DATE_TIME));
```

ChoiceGroup

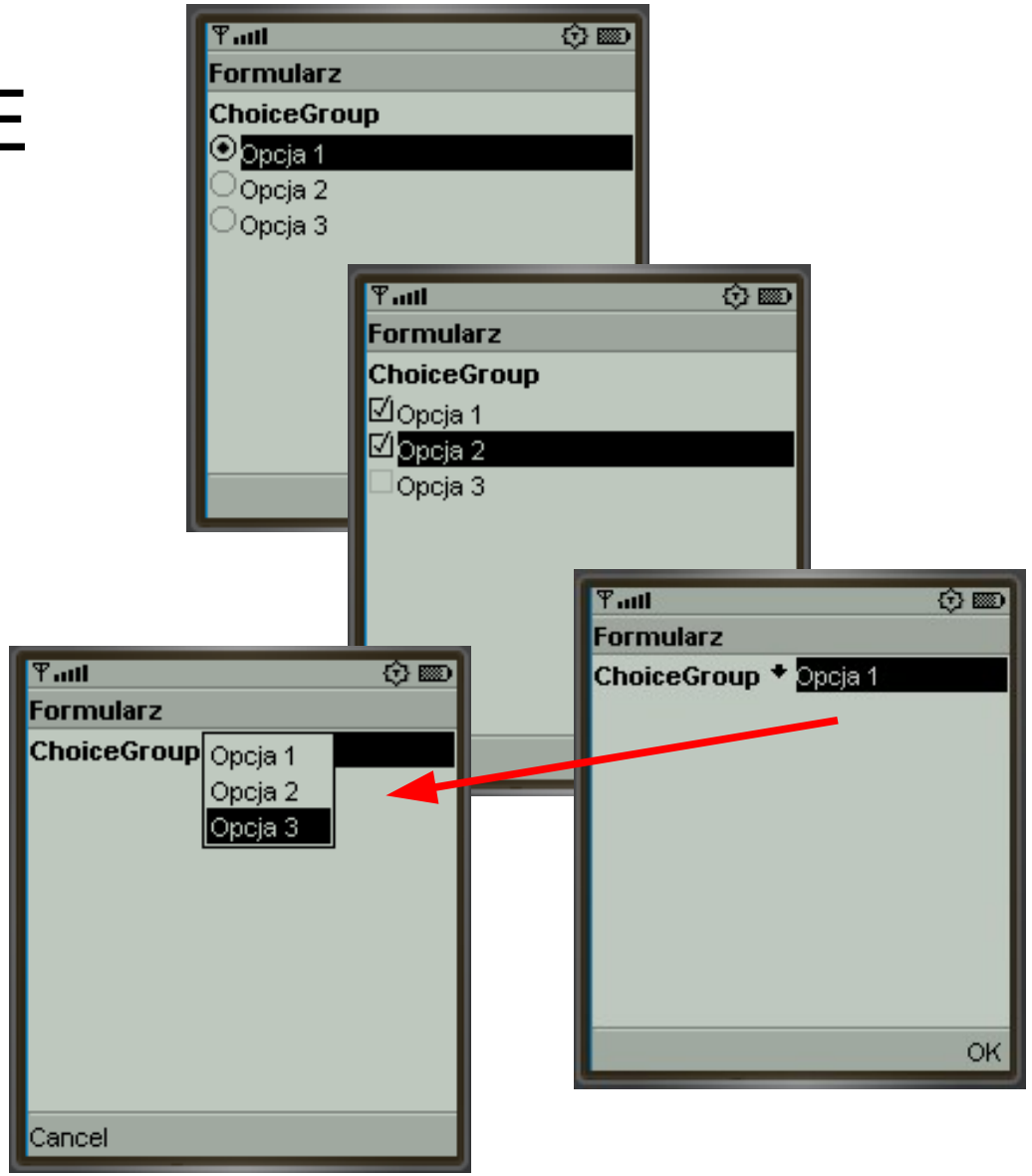
- Lista, podobnie jak ekran List




```
String[] elements = {
  "Opcja 1", "Opcja 2", "Opcja 3"
};
m_Form.append(new ChoiceGroup(
  "ChoiceGroup",     // label
  Choice.EXCLUSIVE, // choiceType
  elements,          // elements
  null));            // imageElements
```

ChoiceGroup - rodzaje

- Choice.EXCLUSIVE
- Choice.MULTIPLE
- Choice.POPUP



Alert raz jeszcze - Indicator

- Inne zastosowanie kontrolki Gauge
  - Pokazywanie postępu wewnątrz ekranu Alert

```
Alert alert = new Alert(
  "Loading",
  "Trwa ładowanie gry, proszę czekać...",
  null,
  AlertType.INFO);
alert.setIndicator(new Gauge(
  null,  // label
  false, // interactive
  9,     // maxValue
  0));   // initialValue
alert.setTimeout(Alert.FOREVER);
```

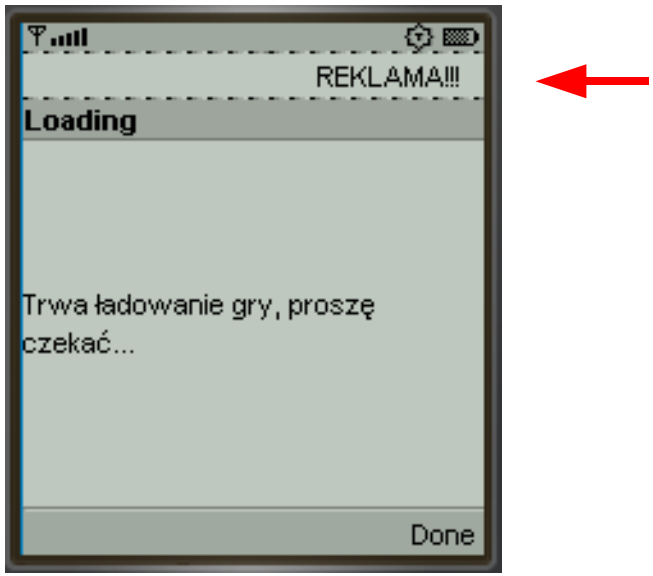
Musi tak być

Flagi: INDEFINITE, CONTINUOUS_IDLE, CONTINUOUS_RUNNING, INCREMENTAL_IDLE, INCREMENTAL_UPDATING

Ticker

- Tekst przelatujący u góry ekranu
  - Dostępny na dowolnym ekranie – metoda setTicker klasy Displayable

```
Alert alert = new Alert(
   "Loading",
   "Trwa ładowanie gry, proszę czekać...",
   null,
   AlertType.INFO);
alert.setTimeout(Alert.FOREVER);

Ticker ticker = new Ticker("REKLAMA!!!");
alert.setTicker(ticker);
```



Timer, TimerTask

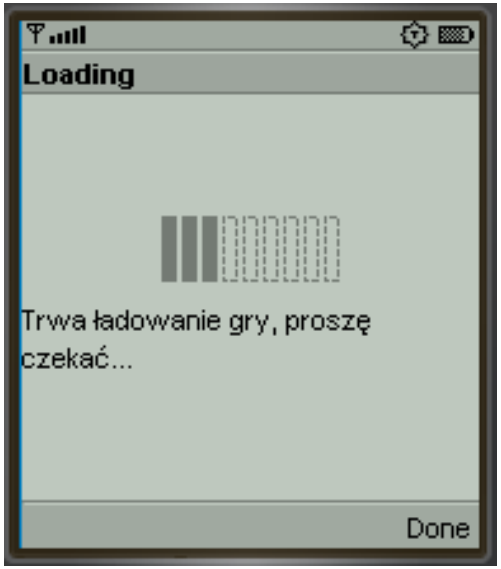
- **java.util.Timer** – pozwala zakolejkować zadania do wykonania za/co określony czas
- **java.util.TimerTask** – klasa bazowa dla zadań

- Każdy Timer to osobny wątek
  - Treść zadania jest wykonywana w osobnym wątku.
  - Czy nie warto czasem napisać własny wątek zamiast używać Timera?

Timer - przykład

```
Gauge m_Gauge;
java.util.Timer m_Timer;

protected void startApp()
  throws MIDletStateChangeException
{
  Alert alert = new Alert(
    "Loading",
    "Trwa ładowanie gry, proszę czekać...",
    null,
    AlertType.INFO);
  alert.setTimeout(Alert.FOREVER);

  m_Gauge = new Gauge(null, false, 9, 0);
  alert.setIndicator(m_Gauge);

  Display display = Display.getDisplay(this);
  display.setCurrent(alert);

  m_Timer = new java.util.Timer();
  m_Timer.schedule(new MyTask(), 500, 500);
}
```

```
private class MyTask
  extends java.util.TimerTask
{
  public void run()
  {
    m_Gauge.setValue(
      m_Gauge.getValue() + 1);
  }
}
```

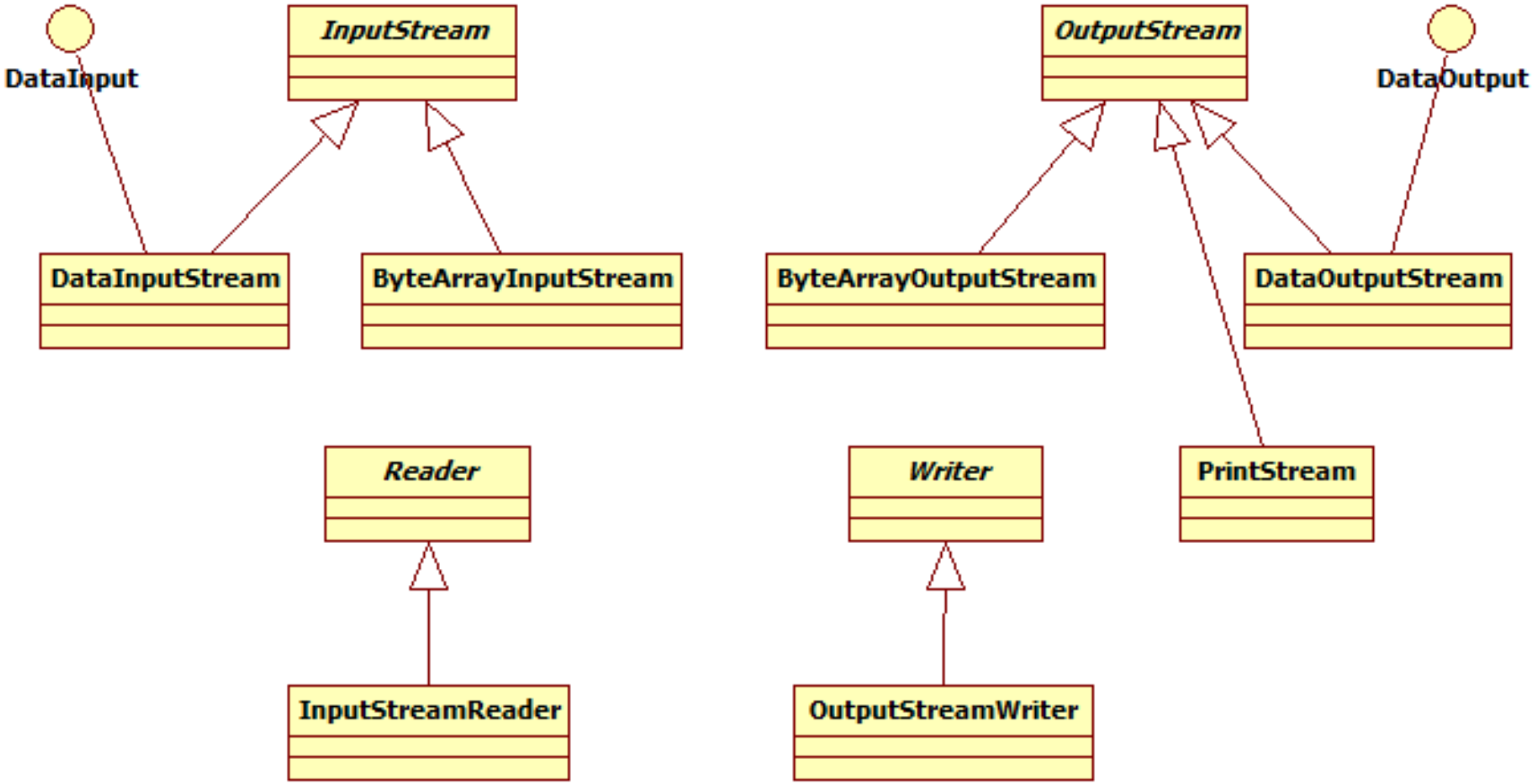


Wejście-wyjście

java.io

Wejście-wyjście

- Pakiet java.io - strumienie



Wejście-wyjście

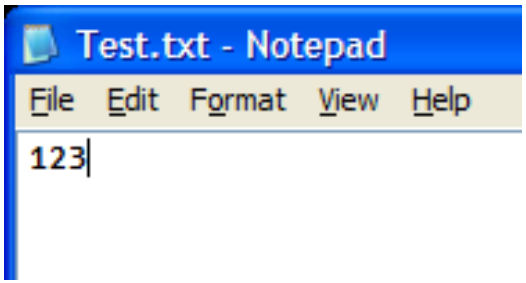
- **InputStream** - abstrakcyjna klasa bazowa do odczytywania danych binarych jako bajty
- **OutputStream** - abstrakcyjna klasa bazowa do zapisywania bajtów jako dane binarne

- **DataInput** - interfejs do odczytywania danych binarnych jako różne typy
  - **DataInputStream** - nakładka na InputStream do odczytywania danych binarnych jako różne typy
- **DataOutput** - interfejs do zapisywania różnych typów jako dane binarne
  - **DataOutputStream** - nakładka na OutputStream do zapisywania różnych typów jako dane binarne

- **Writer** – abstrakcyjna klasa bazowa do zapisywania znaków
  - **OutputStreamWriter** - nakładka na strumień zapisująca do niego znaki
- **Reader** - abstrakcyjna klasa bazowa do odczytywania znaków
  - **InputStreamReader** - nakładka na strumień odczytująca z niego znaki

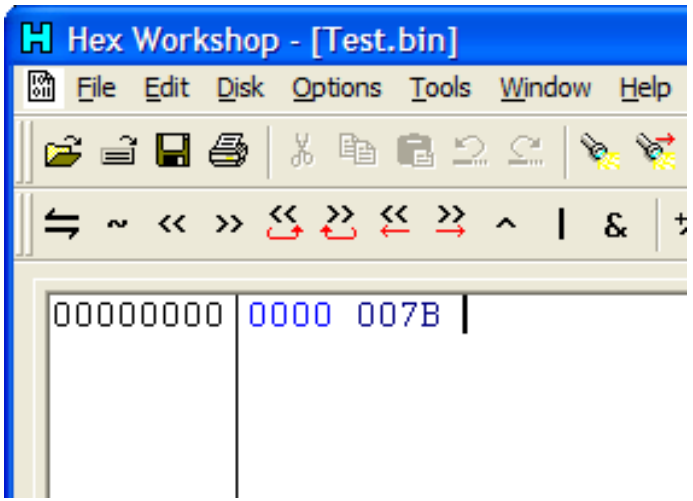
Wczytanie pliku tekstowego

```
String s;
try
{
  java.io.InputStream is =
    this.getClass().getResourceAsStream("/Test.txt");
  StringBuffer sb = new StringBuffer();
  int i;
  while ( (i = is.read()) != -1 )
    sb.append( (char)i );
  s = sb.toString();
}
catch (java.io.IOException e)
{
  e.printStackTrace();
  s = "";
}

// Plik res\Test.txt
```



Wczytanie pliku binarnego

```
String s;
try
{
  java.io.InputStream is =
    this.getClass().getResourceAsStream("/Test.bin");
  java.io.DataInputStream dis =
    new java.io.DataInputStream(is);
  int i = dis.readInt();
  s = Integer.toString(i);
}
catch (java.io.IOException e)
{
  e.printStackTrace();
  s = "";
}


// Plik res\Test.bin (Big Endian!)
```



Internet

javax.microedition.io

Internet

- **HTTP**
  javax.microedition.io.HttpConnection
- **HTTPS, SSL/TLS**
  javax.microedition.io.HttpsConnection
  javax.microedition.io.SecureConnection
- **TCP, UDP**
  javax.microedition.io.SocketConnection
  javax.microedition.io.ServerSocketConnection
  javax.microedition.io.DatagramConnection
  javax.microedition.io.UDPDatagramConnection
- **Push**
  javax.microedition.io.PushRegistry
- **Port szeregowy**
  javax.microedition.io.CommConnection

HTTP - przykład

```
import java.io.*;
import javax.microedition.io.*;

try {
  HttpConnection c =
    (HttpConnection)Connector.open("http://www.google.pl");
  int ResponseCode = c.getResponseCode();
  if (ResponseCode != HttpConnection.HTTP_OK)
    // Błąd!
  String ContentType = c.getType();
  int Length = (int)c.getLength();
  InputStream is = c.openInputStream();
  ...
}
catch (java.io.IOException e)
{
  ...
}
```

Grafika 2D

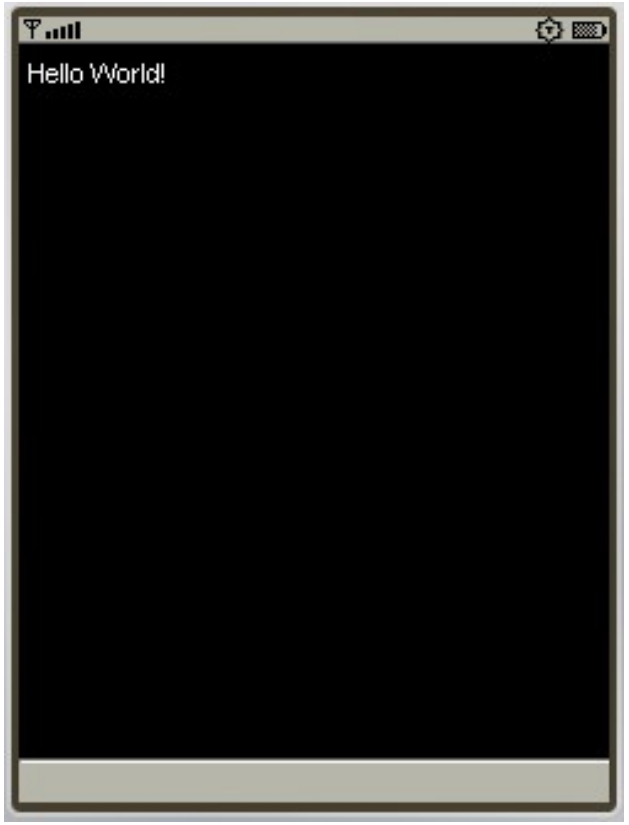
Canvas
Graphics

Canvas

- Klasa **javax.microedition.lcdui.Canvas**
  - Dziedziczy z Displayable
  - Ekran całkowicie do dyspozycji dla programisty
  - Ręczne rysowanie i obsługa wejścia
- Sposób użycia:
  - Zdefiniować klasę dziedziczącą z **Canvas**
  - Zaimplementować metody:
    Obowiązkowo: **paint**
    Opcjonalnie: **sizeChanged, showNotify, hideNotify, keyPressed, keyReleased, keyRepeated, pointerDragged, pointerPressed, pointerReleased**

Canvas - przykład

```
private class MyCanvas extends Canvas
{
  protected void paint(Graphics g)
  {
    g.setColor(0xFF000000);
    g.fillRect(0, 0, getWidth(), getHeight());
    g.setColor(0xFFFFFFFF);
    g.drawString("Hello World!", 4, 4,
      Graphics.TOP | Graphics.LEFT);
  }
}

protected void startApp()
  throws MIDletStateChangeException
{
  MyCanvas canvas = new MyCanvas();

  Display display = Display.getDisplay(this);
  display.setCurrent(canvas);
}
```



Sprawdzanie możliwości

- **Pamięć**

```
Runtime rt = Runtime.getRuntime();
rt.totalMemory(), rt.freeMemory()
```

- **Wyświetlacz**

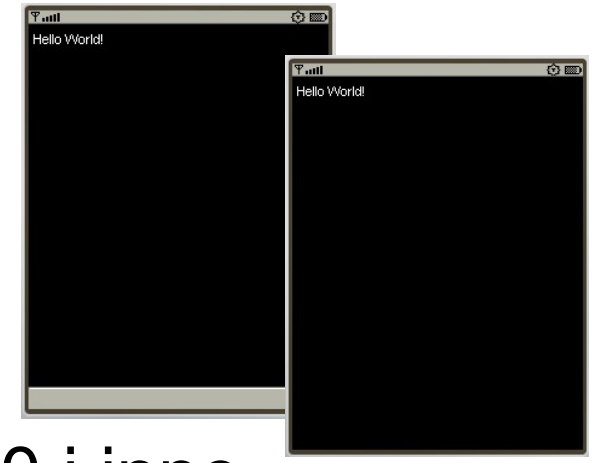
```
Display d = Display.getDisplay(this);
d.isColor(), d.numColors(), d.numAlphaLevels()
```

- **Canvas**

```
c.getWidth(), c.getHeight(),
c.isDoubleBuffered(), c.hasPointerEvents(),
c.hasPointerMotionEvents(), c.hasRepeatEvents()
```

Różnorodność możliwości

- **Klawiatura**
  - Zestaw wymaganych klawiszy + niestandardowe
  - Problemy z klawiszami wciskanymi jednocześnie
- **Urządzenie wskazujące**
  - Całkowicie opcjonalnie
- **Ekran**
  - Bardzo różnorodne rozdzielczości
    128x128, 128x160 176x220, 240x320 i inne...
  - Niektóre ekrany są wyższe, inne szersze!
    640x200, 320x240 i inne...
  - FullScreen – nawet wtedy mogą pozostać paski!
    `canvas.setFullScreenMode(true);`



Graphics – ustawienia

- Prostokąt przycinania

```
void | setClip(int x, int y, int width, int height)
void | clipRect(int x, int y, int width, int height)
```

- Kolor

```
void | setColor(int RGB)
void | setColor(int red, int green, int blue)
void | setGrayScale(int value)
```

- Czcionka

```
void | setFont(Font font)
```

- Styl linii – SOLID, DOTTED

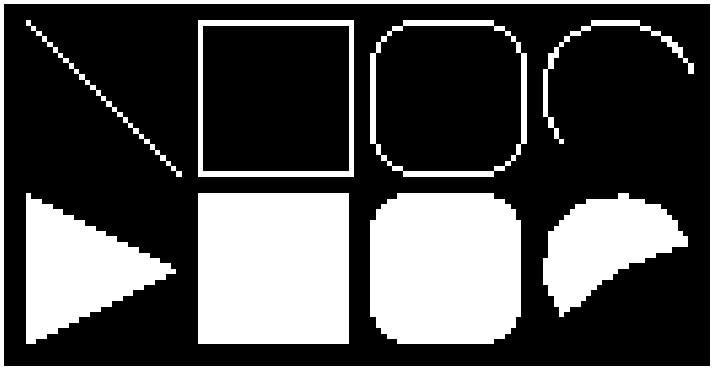
```
void | setStrokeStyle(int style)
```

- Początek układu współrzędnych

```
void | translate(int x, int y)
```

Graphics – rysowanie

```
g.drawLine(4, 4, 32, 32);
g.drawRect(36, 4, 28, 28);
g.drawRoundRect(68, 4, 28, 28, 16, 16);
g.drawArc(100, 4, 28, 28, 20, 200);

g.fillTriangle(4, 36, 32, 50, 4, 64);
g.fillRect(36, 36, 28, 28);
g.fillRoundRect(68, 36, 28, 28, 16, 16);
g.fillArc(100, 36, 28, 28, 20, 200);
```



Graphics – tekst, obrazek

```
g.drawString("Hello World!", getWidth()/2, 4,
  Graphics.TOP | Graphics.HCENTER);

g.drawImage(m_Image, getWidth()/2, getHeight()/2,
  Graphics.HCENTER | Graphics.VCENTER);
```

Anchor:
Graphics.BOTTOM, TOP, VCENTER, BASELINE, LEFT, VCENTER, RIGHT

Canvas - klawiatura

```
private class MyCanvas extends Canvas
{
  protected void keyPressed(int keyCode)
  {
    // ...
  }

  protected void keyReleased(int keyCode)
  {
    // ...
  }

  protected void keyRepeated(int keyCode)
  {
    // ...
  }

  // ...
```

Canvas.KEY_NUM0 – KEY_NUM9
KEY_STAR (*)
KEY_POUND (#)
if (keyCode > 0)
    char ch = (char)keyCode;
Niestandardowe...

Działa tylko jeśli hasRepeatEvents()

Klawisze do gier

- Mapowanie klawiszy zwykłych na akcje gry: getGameAction
- Stałe akcji gry: Canvas.UP, DOWN, LEFT, RIGHT, FIRE, GAME_A, GAME_B, GAME_C, GAME_D

```
protected void keyPressed(int keyCode)
{
  int game_action = getGameAction(keyCode);
  if (game_action == Canvas.FIRE)
    Fire();
  else if (game_action == Canvas.UP)
    Jump();
  // ...
}
```

Programowanie gier 2D

GameCanvas
Layer
LayerManager

GameCanvas

- Pakiet **javax.microedition.lcdui.game**
  - Klasy przeznaczone do pisania gier 2D
- Klasa **GameCanvas**
  - Dziedziczy z Canvas
  - Zapewnie podwójne buforowanie obrazu
  - Umożliwia odpytywanie o wciśnięte klawisze
- Pętlę gry trzeba zrealizować samemu
  - Za pomocą Timera lub
  - Za pomocą własnego wątku

Pętla gry – propozycja

```
private class MyCanvas
  extends GameCanvas
  implements Runnable
{
  public MyCanvas()
  {
    super(false);
    setFullScreenMode(true);
  }

  public void run()
  {
    Graphics g = getGraphics();
    while (true) {
      int keyState = getKeyStates();
      if ((keyState & LEFT_PRESSED) != 0)
        // ...
      // Obliczenia...

      // Rysowanie...
      g.setColor(0xFF000000);
      g.fillRect(0, 0,
        getWidth(), getHeight());

      flushGraphics(); // Koniec rysowania
    }
  }
}
```

```
MyCanvas m_MyCanvas;
Thread m_Thread;

protected void startApp()
throws MIDletStateChangeException
{
  m_MyCanvas = new MyCanvas();

  Display display =
   Display.getDisplay(this);
  display.setCurrent(m_MyCanvas);

  m_Thread = new Thread(m_MyCanvas);
  m_Thread.start();
}
```

Wejście z klawiatury

- **getKeyStates()**
  Zwraca flagi bitowe z wciśniętymi w danej chwili klawiszami
- **super(false);**
  Zdarzenia keyPressed, keyReleased, keyRepeated nie będą generowane - wydajność!

Stałe GameCanvas.
LEFT_PRESSED
RIGHT_PRESSED
UP_PRESSED
DOWN_PRESSED
FIRE_PRESSED
GAME_A_PRESSED
GAME_B_PRESSED
GAME_C_PRESSED
GAME_D_PRESSED

```
protected GameCanvas(boolean suppressKeyEvents)
```

Pomiar czasu i FPS

```
long StartTime = System.currentTimeMillis();
long LastTime = 0;
long t = 0, dt = 0;
long LastFpsTime = 0;
int FpsCounter = 0, Fps = 0;

while (true) {
  int KeyStates = getKeyStates();
  CalcFrame(t, dt, KeyStates);
  DrawFrame(g);
  DrawFPS(Fps);
  flushGraphics();

  LastTime = t;
  t = System.currentTimeMillis() - StartTime;
  dt = t - LastTime;
  FpsCounter++;
  if (LastFpsTime + 1000 <= t) {
    Fps = FpsCounter;
    FpsCounter = 0;
    LastFpsTime += 1000;
  }
}
```

- **System.currentTimeMillis()** - zwraca 64-bitową liczbę milisekund od 1 stycznia 1970.
- Warto stosować **Thread.sleep()** - oszczędność energii!

Layer

- Klasa **javax.microedition.lcdui.game.Layer**
  - Reprezentuje pojedynczy obiekt graficzny 2D (wbrew nazwie!)
  - Dwie klasy pochodne: **Sprite**, **TiledLayer**
- Zawiera:
  - Pozycję
  - Stan widoczności
  - Rozmiar (tylko do odczytu)
  - Umiejętność narysowania się na Graphics

int	`getHeight()` Gets the current height of this la
int	`getWidth()` Gets the current width of this la
int	`getX()` Gets the horizontal position of t
int	`getY()` Gets the vertical position of this
boolean	`isVisible()` Gets the visibility of this Layer.
void	`move(int dx, int dy)` Moves this Layer by the specifi
abstract void	`paint(Graphics g)` Paints this Layer if it is visible.
void	`setPosition(int x, int y)` Sets this Layer's position such t
void	`setVisible(boolean visible)` Sets the visibility of this Layer.

Sprite

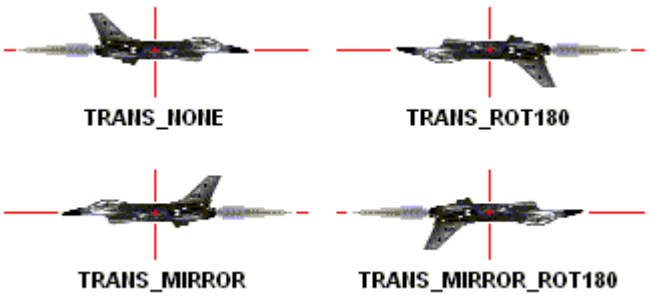
```
Sprite m_Sprite;

public MyCanvas()
{
  // ...
  m_Sprite = new Sprite(image);
  m_Sprite.setPosition(4, 4);
}

public void run()
{
  // ...
  int keyState = getKeyStates();
  if ((keyState & LEFT_PRESSED ) != 0) m_Sprite.move(-1 * (int)dt, 0);
  if ((keyState & RIGHT_PRESSED) != 0) m_Sprite.move( 1 * (int)dt, 0);
  if ((keyState & UP_PRESSED   ) != 0) m_Sprite.move(0, -1 * (int)dt);
  if ((keyState & DOWN_PRESSED ) != 0) m_Sprite.move(0,  1 * (int)dt);

  // ...
  m_Sprite.paint(g);

  // ...
}
```

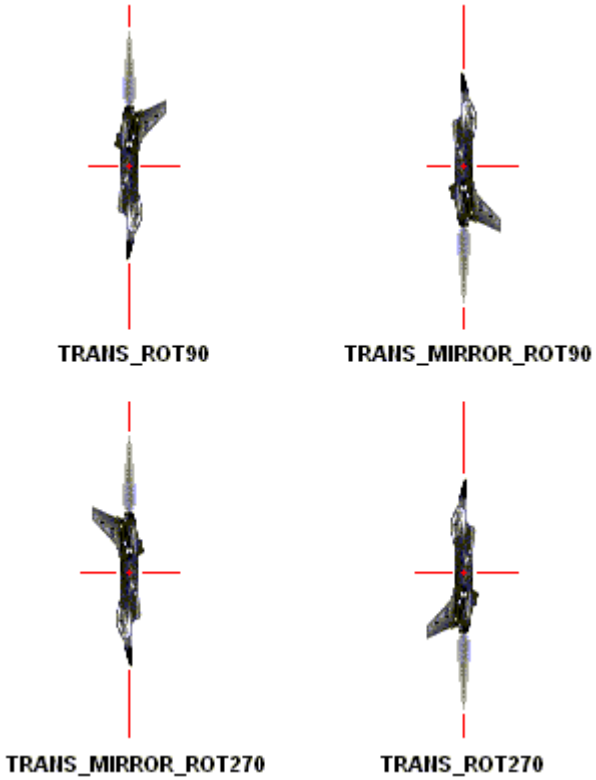
Pojedynczy obrazek

Sprite - transformacje

- Obrazek może być obrócony lub odbity
  - Niedostępne w zwykłym Image!
  - Skalowanie i obracanie o dowolny kąt nadal niedostępne :(

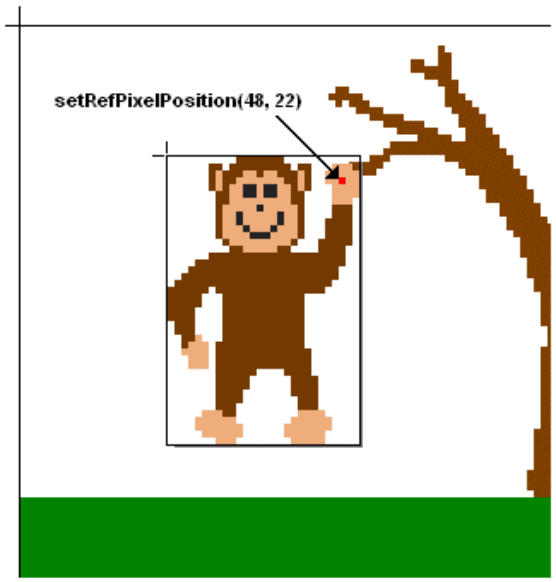
```
void setTransform(int transform)
```

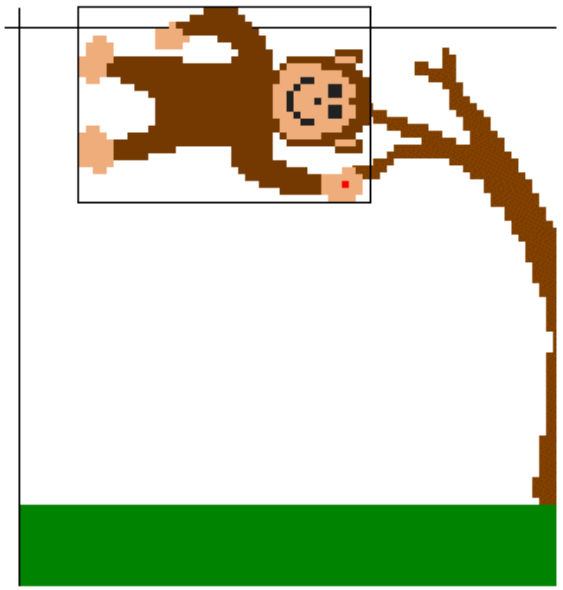




Sprite – piksel odniesienia

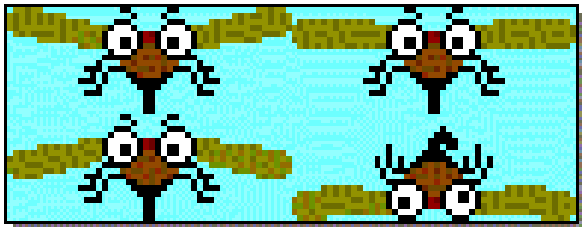
- **defineReferencePixel()** - ustawia piksel odniesienia
- **setRefPixelPosition()** - ustawia obrazek tak żeby piksel odniesienia był w danym miejscu
  - Działa także z obrazkami obróconymi





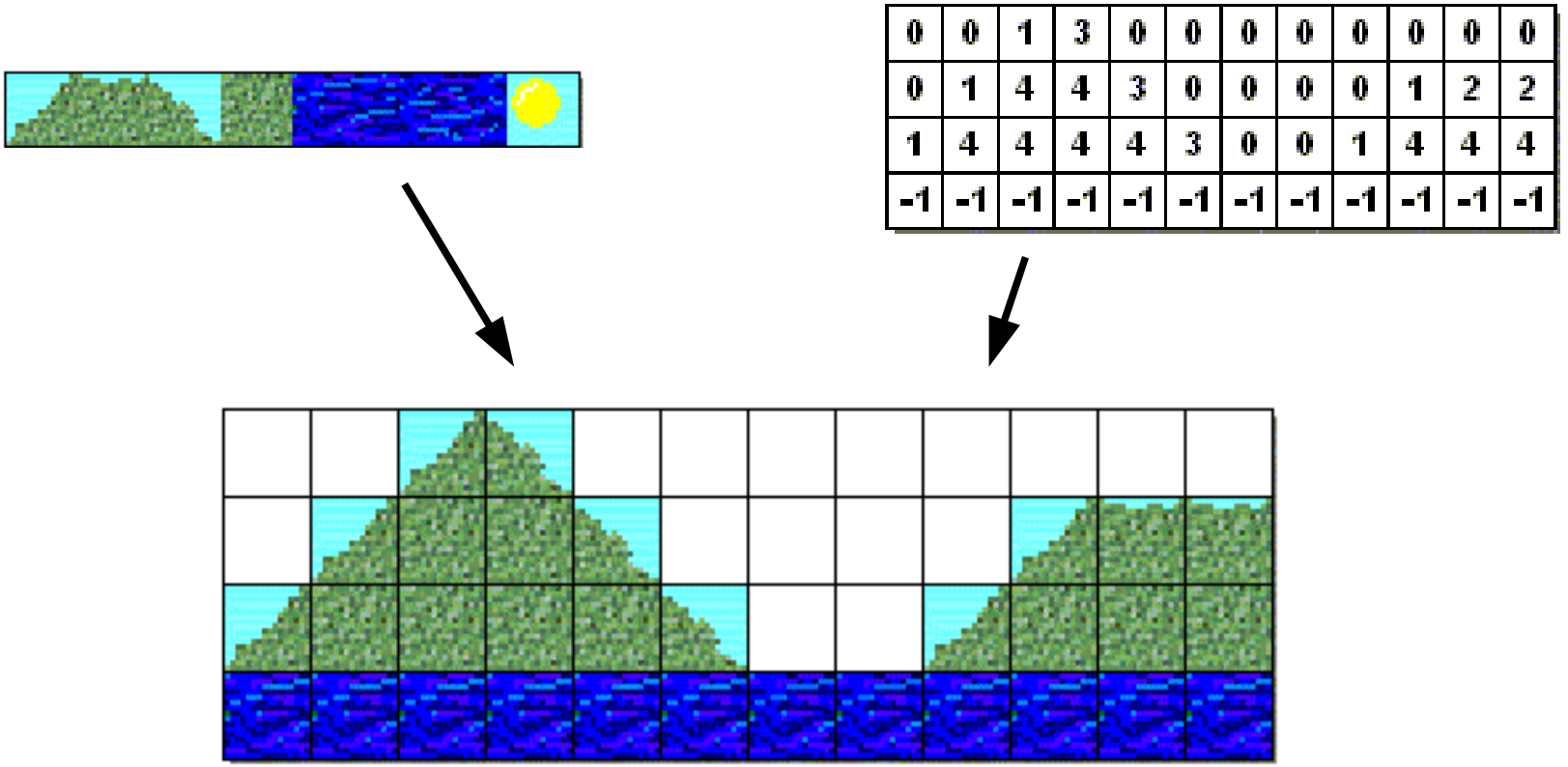
Sprite - animacja

- Obrazek może się składać z wielu klatek animacji
- Domyślnie animację tworzą klatki pokazywane kolejno
- Można ustawić własną sekwencję klatek **setFrameSequence()**
- Animacją trzeba sterować ręcznie **setFrame(), prevFrame(), nextFrame()**



TiledLayer

- Obiekt pokazujący tablicę kafelków kopiowanych z obrazka źródłowego



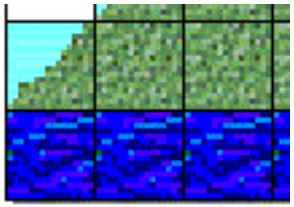
TiledLayer – rodzaje kafelków

```
void setCell(int col, int row, int tileIndex)
void fillCells(int col, int row, int numCols, int numRows, int tileIndex)
```

- = 0 – pusta komórka
- > 0 – fragment obrazka źródłowego
- < 0 – komórka animowana

```
int createAnimatedTile(int staticTileIndex)
void setAnimatedTile(int animatedTileIndex, int staticTileIndex)
```

- Zastosowanie – np. animowana woda



Sprite - kolizje

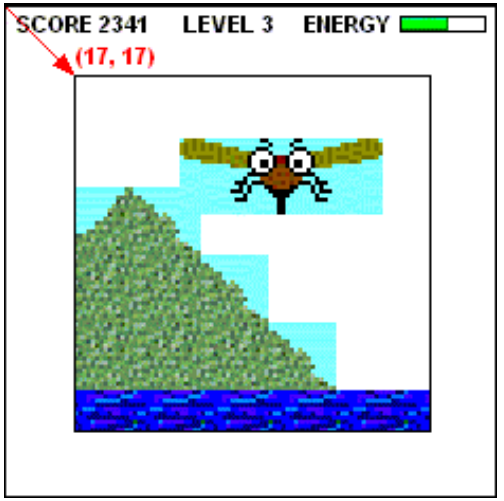
boolean	collidesWith(Image image, int x, int y, boolean pixelLevel) Checks for a collision between this Sprite and the specified Image with its u
boolean	collidesWith(Sprite s, boolean pixelLevel) Checks for a collision between this Sprite and the specified Sprite.
boolean	collidesWith(TiledLayer t, boolean pixelLevel) Checks for a collision between this Sprite and the specified TiledLayer.
void	defineCollisionRectangle(int x, int y, int width, int height) Defines the Sprite's bounding rectangle that is used for collision detection pu

- Wsparcie dla kolizji między obiektami
  - Także z kafelkami TiledLayer
  - Także na poziomie przezroczystości pojedynczych pikseli !!!

LayerManager

- Klasa upraszczająca zarządzanie obiektami 2D
  - Jej wykorzystanie jest opcjonalne.
- Przechowuje listę obiektów klasy Layer
  - Pamięta kolejność
  - Pozwala je narysować jednym wywołaniem
- Oddzielenie prostokąta ekranu od prostokąta rysowanej sceny
  - Zaleta #1: Rysowanie widoku tylko w ograniczonym prostokącie
  - Zaleta #2: Przewijanie mapy

```
void setViewWindow(int x, int y, int width, int height)
void paint(Graphics g, int x, int y)
```



Kilka drobiazgów

Wibracja
Właściwości systemu
Obsługa błędów

Wibracja

- Dodatkowy efekt zmysłowy!
  - Niedostępny na PC
  - Chyba, że gracz ma joypad z Force Feedback i używasz DirectInput...

```
Display display = Display.getDisplay(this);
display.vibrate(200);
```

- Wywołanie jest asynchroniczne – nie blokuje
- Podawany czas jest w milisekundach
  - 0 oznacza wyłączenie wibracji

Właściwości systemu

- **System.getProperty**

```
static String getProperty(String key)
```

- Właściwości standardowe
  microedition.profiles
  microedition.configuration
  microedition.locale
  itd...
- Właściwości dodatkowe
  microedition.m3g.version
  Bluetooth.api.version
  video.encodings
  itd...

Obsługa błędów

- Trzeba zawsze łapać wyjątki lub deklarować ich zgłaszanie
  - Jak zawsze w Javie.
  - Są to np.:
    - **java.io.IOException**
      podczas wczytywania plików
    - **java.lang.InterruptedException**
      przy oczekiwaniu w Thread, np. join(), sleep()
- W emulatorze działa **System.out.println()**
  - A więc także **Throwable.printStackTrace()**

```
Image image;
try {
  image = Image.createImage("/Obrazek.png”);
}
catch (java.io.IOException e) {
  e.printStackTrace();
  image = null;
}
```

Baza danych

javax.microedition.rms.RecordStore

Baza danych???

- MIDP nie daje dostępu do systemu plików
  - Chyba, że przez rozszerzenie – JSR-75
  - Nie ma innej możliwości zachowania trwałych danych niż RMS
- RMS – Record Management System
  - Pakiet **javax.microedition.rms**, klasa **RecordStore**
  - Aplikacja może tworzyć i używać baz danych. Każda ma nazwę.
  - Baza danych składa się z rekordów. Każdy ma identyfikator oraz treść – surowe dane binarne.

Dodawanie rekordu

```
import java.io.*;
import javax.microedition.rms.*;

try
{
  RecordStore rs = RecordStore.openRecordStore("Baza1", true);
  ByteArrayOutputStream baos = new ByteArrayOutputStream();
  DataOutputStream dos = new DataOutputStream(baos);
  dos.writeUTF("Mieczysław");
  dos.writeInt(1000);
  byte[] data = baos.toByteArray();
  int rec_id = rs.addRecord(data, 0, data.length);
}
catch (RecordStoreException e) { }
catch (IOException e) { }
```

Enumeracja rekordów

```
try
{
  RecordStore rs = RecordStore.openRecordStore("Baza1", true);
  RecordEnumeration re = rs.enumerateRecords(null, null, false);
  while (re.hasNextElement())
  {
    int id = re.nextRecordId();
    ByteArrayInputStream bais =
      new ByteArrayInputStream(rs.getRecord(id));
    DataInputStream dis = new DataInputStream(bais);
    String name = dis.readUTF();
    int score = dis.readInt();
    System.out.println(name + " - " + score);
  }
}
catch (RecordStoreException e) { }
catch (IOException e) { }
```

```
RecordEnumeration enumerateRecords(RecordFilter filter, RecordComparator comparator, boolean keepUpdated)
```

Enumeracja - filtrowanie

```
private class MyFilter implements RecordFilter
{
  public boolean matches(byte[] candidate)
  {
    try {
      ByteArrayInputStream bais = new ByteArrayInputStream(candidate);
      DataInputStream dis = new DataInputStream(bais);
      String name = dis.readUTF();
      int score = dis.readInt();
      return (score > 1000);
    }
    catch (IOException e) { e.printStackTrace(); return false; }
  }
}

...

RecordEnumeration re = rs.enumerateRecords(new MyFilter(), null, false);
```

Enumeracja - sortowanie

```
private class MyComparator implements RecordComparator
{
  public int compare(byte[] rec1, byte[] rec2)
  {
    try {
      ByteArrayInputStream bais1 = new ByteArrayInputStream(rec1);
      ByteArrayInputStream bais2 = new ByteArrayInputStream(rec2);
      DataInputStream dis1 = new DataInputStream(bais1);
      DataInputStream dis2 = new DataInputStream(bais2);
      int score1 = dis1.readInt();
      int score2 = dis2.readInt();
      String name1 = dis1.readUTF();
      String name2 = dis2.readUTF();
      int cmp = name1.compareTo(name2);
      if (cmp < 0)       return RecordComparator.PRECEDES;
      else if (cmp == 0) return RecordComparator.EQUIVALENT;
      else               return RecordComparator.FOLLOWS;
    }
    catch (IOException e) {
      e.printStackTrace();
      return RecordComparator.PRECEDES;
    }
  }
}
...
RecordEnumeration re = rs.enumerateRecords(null, new MyComparator(), false);
```

Inne możliwości

- Operacje na rekordach
  addRecord, getRecord, getRecordSize, deleteRecord
- Nasłuchiwanie zmian
  addRecordListener, removeRecordListener,
  interfejs RecordListener
- Informacje o bazie danych
  getNumRecords, getSize, getSizeAvailable,
  getLastModified, getVersion
- Operacje na bazach danych
  listRecordStores, openRecordStore,
  deleteRecordStore
- Współdzielenie baz danych między midletami...

MIDP 2.0 Media API

javax.microedition.media

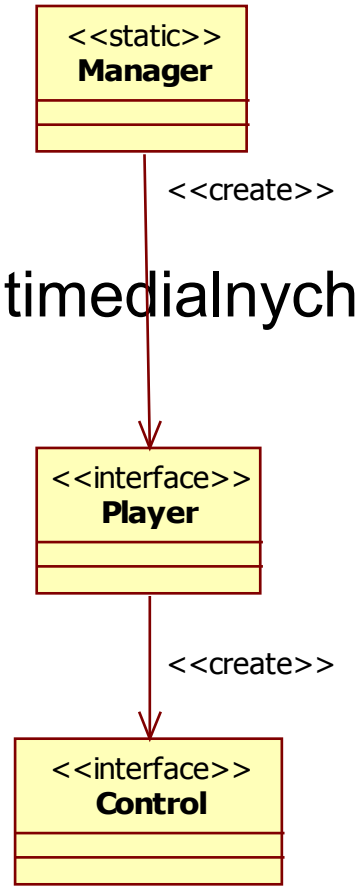
MIDP 2.0 Media API

- Zapewnia podstawowe API do odtwarzania multimediów
  - Tylko dźwięk
  - Rozszerzane przez Mobile Media API (JSR-135)
- Nie gwarantuje obsługi żadnych formatów
  - Jedynie proste generowanie odgłosów

```
static void playTone(int note, int duration, int volume)
```

MIDP 2.0 Media API

- **Manager**
  - Klasa statyczna
  - Dostarcza informacji na temat możliwości multimedialnych urządzenia
  - Umożliwia tworzenie Playerów
- **Player**
  - Odtwarza konkretny zasób multimedialny
  - Pozwala na sterowanie odtwarzaniem
  - Posiada stany
  - Pozwala na pobieranie kontrolerów
- **Control**
  - Pozwala na regulację parametrów odtwarzania, np. głośności



Odtwarzanie dźwięku

```
try {
  Player p = Manager.createPlayer("http://adres.pl/ding.wav");
  p.setLoopCount(5);
  p.start();
}
catch (IOException ioe) { }
catch (MediaException me) { }
```

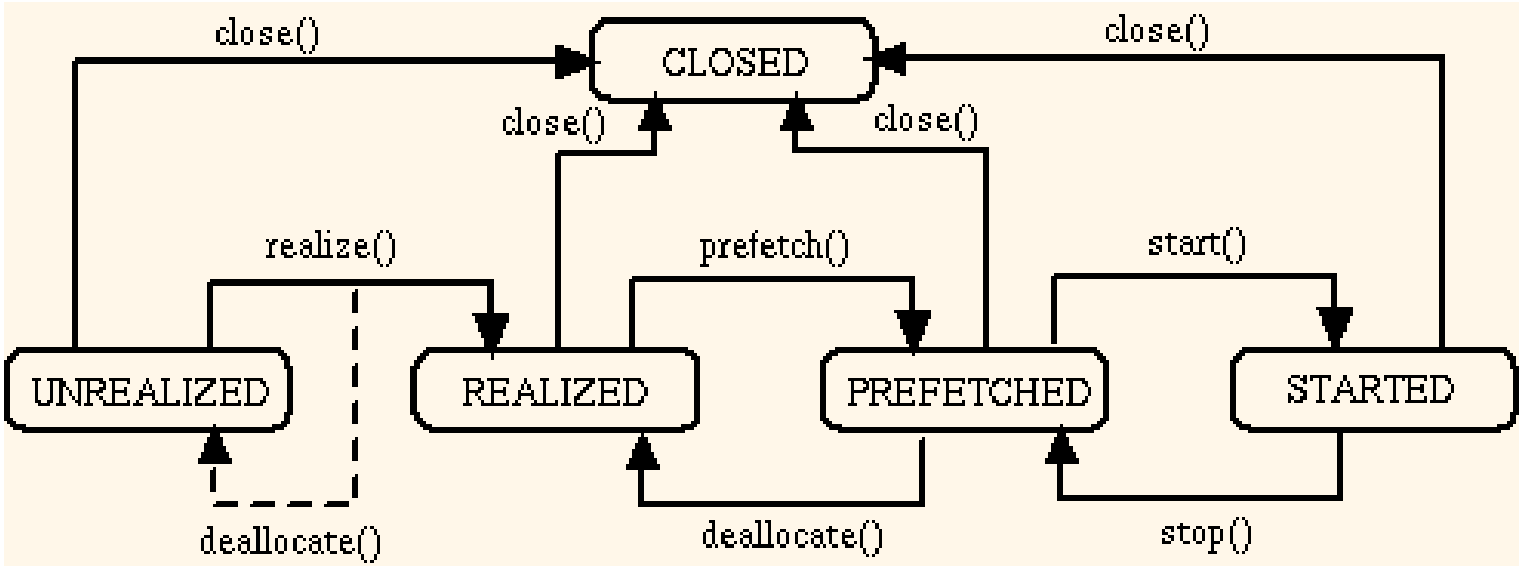
```
try {
  InputStream is = getClass().getResourceAsStream("/ding.wav");
  Player p = Manager.createPlayer(is, "audio/X-wav");
  p.start();
}
catch (IOException ioe) { }
catch (MediaException me) { }
```

Obsługiwane formaty

static String[]	**getSupportedContentTypes**(String protocol) Return the list of supported content types for the given protocol.
static String[]	**getSupportedProtocols**(String content_type) Return the list of supported protocols given the content type.

- Content Type – typ MIME, np.:
  - Dźwięk WAV: **audio/x-wav**
  - Dźwięk AU: **audio/basic**
  - Dźwięk MP3: **audio/mpeg**
  - Dźwięk MIDI: **audio/midi**
  - Tone sequence: **audio/x-tone-seq**

Player – stany



Rozszerzenia

MIDP 2.0 to nie koniec...

Przykłady rozszerzeń

- Grafika 3D
  - Mobile 3D Graphics API – M3G (JSR-184)
  - Mascot Capsule
- Grafika i multimedia
  - Mobile Media API (JSR-135)
  - Nokia UI API
- Dostęp do plików
  - FileConnection (JSR-75)
- Dostęp do danych PIM (kontakty itp.)
  - PIM (JSR-75)
- Łączność
  - Java APIs for Bluetooth (JSR-82)
  - Wireless Messaging API – WMA (JSR-120)
- Wiele innych...