

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ

SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

J2ME. Java dla urządzeń mobilnych. Ćwiczenia

utor: Krzysztof Rychlicki-Kicior

ISBN: 83-246-0393-X

Format: B5, stron: 148



Poznaj podstawy tworzenia aplikacji dla telefonów komórkowych

- Zainstaluj środowisko programistyczne
- Napisz własny MIDlet
- Połącz aplikację z internetem

Niemal każdy telefon komórkowy umożliwia uruchamianie aplikacji napisanych w języku Java. Jednak tworzenie takich aplikacji różni się od pisania programów na „duże” komputery. Korzystanie z języka Java dla urządzeń mobilnych (J2ME) wymusza na programiście pewne ograniczenia wynikające z mniejszej ilości pamięci, odmiennych metod komunikacji i wielu innych czynników. Tworzenie aplikacji J2ME, zwanych MIDletami, wymaga poznania tych ograniczeń oraz specyficznych zasad programowania.

Książka „J2ME. Java dla urządzeń mobilnych. Ćwiczenia” to przewodnik po podstawowych zagadnieniach związanych z tworzeniem aplikacji dla urządzeń mobilnych. Wykonując zawarte w niej ćwiczenia, nauczysz się korzystać z tej wersji Javy. Stworzysz proste MIDlety i poznasz zasady tworzenia interfejsów użytkownika. Dowiesz się, jak implementować połączenia internetowe i korzystać z danych zawartych w zewnętrznych plikach.

- Instalacja środowiska J2ME Wireless Toolkit
- Podstawowy szkielet MIDletu
- Importowanie klas
- Komponenty interfejsu użytkownika
- Zapis danych do pamięci telefonu
- Pobieranie zawartości stron internetowych
- Odtwarzanie dźwięków

Odczuwasz brak jakiejś funkcji w telefonie komórkowym?

Napisz samodzielnie program, który ją zrealizuje



Spis treści

Wstęp	7
Rozdział 1. Instalacja środowiska WTK	9
1.1. Pobranie komponentów z internetu	9
1.2. Instalacja oprogramowania	11
1.3. Omówienie środowiska J2ME Wireless Toolkit	12
1.3.1. KToolbar	12
1.3.2. OTA Provisioning	13
Rozdział 2. Pierwszy MIDlet	17
2.1. Tworzenie nowego projektu	17
2.2. Podstawowy szkielet MIDletu	18
2.2.1. Importowanie pakietów	18
2.2.2. Deklaracja klasy	19
2.2.3. Deklaracja koniecznych metod	19
2.2.4. Konstruktor	20
2.3. Publikowanie MIDletu	23
Rozdział 3. Interfejs użytkownika w MIDletach	27
3.1. MID Profile — nie wszystkie urządzenia mają takie same możliwości	28
3.2. Słuchacze zdarzeń (Listener) a komponenty	29
3.3. Najczęściej używane komponenty	30
3.3.1. Command — polecenie	30
3.3.2. TextField — pole tekstowe	32
3.3.3. StringItem — (nie)zwykły tekst	35
3.3.4. ChoiceGroup — lista wyboru	38
3.3.5. Image i ImageItem — obrazki	42

3.3.6.	Canvas i TextBox — inne sposoby wykorzystania powierzchni wyświetlacza	48
3.3.7.	Timer — zegar i regulator	54
Rozdział 4.	Zapisywanie i odczytywanie informacji w telefonie	59
4.1.	Sposób I — wykorzystanie plików z archiwów JAR	59
4.2.	Sposób II — wykorzystywanie RMS	64
4.2.1.	Zapisywanie danych	65
4.2.2.	Wczytywanie danych	69
4.2.3.	Wykorzystanie klas ByteArrayInputStream i ByteArrayOutputStream do ułatwienia pracy z RMS	72
Rozdział 5.	Urządzenia mobilne a internet	79
5.1.	Zasady korzystania z klas gniazdek	80
5.2.	Używanie interfejsu HttpURLConnection do pobierania stron internetowych	84
5.3.	SocketConnection — interfejs do ogólnej komunikacji między urządzeniami	93
Rozdział 6.	Odtwarzanie dźwięków	105
6.1.	Pakiet javax.microedition.media	105
Rozdział 7.	Projekt — aplikacja Baza Kontaktów	109
7.1.	Tworzenie interfejsu użytkownika	109
7.2.	Obsługa dostępu do danych	112
7.3.	Obsługa interfejsu użytkownika	120
Dodatek A		127
A.1.	Komentarze	127
A.2.	Podstawowe elementy programu	128
A.2.1.	Zmienne	128
A.2.2.	Operatory relacji	130
A.2.3.	Operator trójargumentowy	131
A.2.4.	Operatory logiczne	131
A.3.	Instrukcje złożone (sterujące)	131
A.3.1.	Instrukcja warunkowa	132
A.3.2.	Instrukcja Switch	132
A.4.	Pętle	133
A.5.	Tablice	135
A.6.	Klasy	135
A.7.	Dziedziczenie	138
A.8.	Interfejsy	140

A.9. Pakiety	141
A.10. Tworzenie obiektów	142
A.10.1. Metody statyczne	142
A.11. Klasa String	144
A.12. Rzutowanie	145
A.13. Konstrukcja try..catch	146
Bibliografia	147



Pierwszy MIDlet



W tym rozdziale zapoznasz się ze schematem typowego MIDletu. Następnie wykorzystasz go do napisania aplikacji wyświetlającej na ekranie emulatora tekst. Osobom, które nie programowały wcześniej w Javie (lub żadnym innym języku programowania), polecam zapoznanie się z Dodatkiem A. Opisano w nim najważniejsze elementy języka Java (głównie te, które wykorzystasz w pracy nad MIDletami).

2.1. Tworzenie nowego projektu

W poniższym ćwiczeniu utworzymy nowy projekt. Wykorzystamy go później do stworzenia MIDletu.

Ć W I C Z E N I E

2.1 Tworzenie projektu przy pomocy środowiska WTK

Uruchom środowisko WTK i utwórz nowy projekt.

1. Otwórz program J2ME Wireless Toolkit.
2. Kliknij przycisk *New Project*.
3. W pole *Project Name* wpisz `projekt1`, a w pole *MIDlet Class Name* — `klasa1`.

4. W okienku *Settings* przejdź na zakładkę *MIDlets* i upewnij się, że w polu *Class* znajduje się wartość `klasa1`. Kliknij przycisk *OK*.

Spróbuj teraz skompilować program poleceniem *Build*. Po wykonaniu wszystkich operacji powinien wyświetlić się ciąg komunikatów:

```
Project settings saved
Building "projekt1"
No sources to compile
Build failed
```

MIDlet nie może zostać skompilowany, gdyż kompilator nie może odnaleźć kodu źródłowego. Można wyciągnąć z tego ważny wniosek — środowisko WTK nie tworzy plików zawierających kody źródłowe. Nie generuje też najprostszych nawet schematów aplikacji. Musimy zatem utworzyć ów schemat sami.

2.2. Podstawowy szkielet MIDletu

2.2.1. Importowanie pakietów

Na początku musimy określić, jakie pakiety należy zaimportować do naszej aplikacji. W pierwszym programie wykorzystamy dwa spośród nich:

```
javax.microedition.lcdui
```

oraz

```
javax.microedition.midlet
```

Jaka jest rola tych pakietów w tworzeniu aplikacji? Pakiet `javax.microedition.midlet`, a w szczególności znajdująca się w nim klasa `MIDlet` (jest to klasa bazowa dla wszystkich klas MIDletów), dostarcza podstawowych funkcji, takich jak zamykanie aplikacji. W pakiecie `javax.microedition.lcdui` znajdują się wszelkiego rodzaju komponenty graficzne, czyli klasy odpowiedzialne za wyświetlanie elementów graficznych (np. pól tekstowych, tekstów, list wyboru itd.).

2.2.2. Deklaracja klasy

Skoro dysponujemy już niezbędnymi elementami, możemy przystąpić do projektowania klasy. Będzie ona dziedziczyła po klasie `MIDlet` — stąd deklaracja:

```
public class klasa1 extends MIDlet
```

Proszę zwrócić uwagę, iż trzeba podać tę samą nazwę, którą wpisaliśmy przy tworzeniu projektu w programie Wireless Toolkit (`klasa1`).

2.2.3. Deklaracja koniecznych metod

Dziedziczenie po klasie `MIDlet` zobowiązuje nas do zdefiniowania trzech metod, które w klasie `MIDlet` zostały oznaczone jako **abstrakcyjne**. Metody abstrakcyjne nie zawierają żadnej treści i zmuszają programistów tworzących klasy pochodne do samodzielnego ich zdefiniowania.

Metody abstrakcyjne, o których mowa, to:

```
protected void startApp()  
protected void pauseApp()  
protected void destroyApp(boolean unconditional)
```

Są one bezpośrednio związane ze stanem, w jakim znajduje się `MIDlet`. W momencie uruchamiania aplikacji wywołana zostaje metoda `startApp()`. Jeśli chcemy zwolnić zaalokowane wcześniej zasoby, możemy użyć metody `destroyApp()` przed zakończeniem działania aplikacji. Metoda `pauseApp()` jest swoistym ewenementem, gdyż istnienie takiej metody ma sens tylko w telefonach komórkowych (lub innych urządzeniach z możliwością wykonywania połączeń telefonicznych). Zostaje ona wywołana w chwili, gdy ktoś dzwoni i telefon musi zająć się obsługą rozmowy. Jeśli zatem nasz `MIDlet` wykonuje jakieś operacje w tle, np. pobieranie pliku z internetu, w metodzie `pauseApp()` możemy zapisać informację o stanie pobierania w pamięci telefonu. Działanie aplikacji nie jest jednak zatrzymane na zawsze — w pewnym momencie rozmowa kończy się i telefon powraca do zatrzymanego programu. Wtedy to zostaje ponownie wywołana metoda `startApp()`. Uważny Czytelnik (lub osoba, która zajrzała do dalszych rozdziałów książki) zauważy, że są czynności, które aplikacja może wykonać tylko raz — np. utworzenie komponentów graficznych. Wykonanie tego procesu kilka razy (w przypadku przerywania aplikacji) spowodowałoby

w najlepszym razie dziwne zachowanie naszego MIDletu. W związku z tym tworzenie interfejsu i inne operacje jednorazowe wykonuje się w specjalnej metodzie, zwanej konstruktorem.

2.2.4. Konstruktor

W poprzednim akapicie została wyjaśniona rola konstruktora w działaniu MIDletu. Jego deklaracja wygląda prosto:

```
public klasa1()
```

Warto pamiętać, iż konstruktor jest wywoływany PRZED metodą `startApp()`, co może być przydatne np. przy alokowaniu zasobów lub tworzeniu wątków.

Po omówieniu schematu możemy przystąpić do pisania pierwszej aplikacji — MIDletu wypisującego na ekranie krótki tekst. W tym celu najpierw wykonamy ćwiczenie 2.2 — utworzenie pliku, w którym umieścimy kod źródłowy.

Ć W I C Z E N I E

2.2

Utworzenie pliku, w którym znajdzie się kod źródłowy MIDletu

Przygotuj plik, w którym zostanie zapisany kod programu. Musi on znajdować się w określonej lokalizacji (zgodnej z ustawieniami projektu).

1. Uruchom program *Eksplorator Windows*.
2. Przejdź do katalogu `c:\WTK22\apps\projekt1\src` (jeśli środowisko WTK zostało zainstalowane w domyślnym katalogu).
3. Utwórz plik o nazwie `klasa1.java` (w systemie Windows kliknij prawym przyciskiem myszy puste pole w folderze, następnie z menu wybierz opcję *Nowy/Dokument tekstowy* i wpisz nazwę pliku).
4. Kliknij dwukrotnie ikonę nowo utworzonego pliku. Jeśli nie instalowałeś wcześniej żadnego środowiska programistycznego Javy, system poprosi Cię o wybranie programu, który będzie używany przy edycji plików o rozszerzeniu `.java`. Dla naszej pracy nie ma to znaczenia — ważne jest tylko, aby edytor zapisywał tekst bez żadnych dodatkowych znaczników (jak w przypadku formatu `.txt`).



W systemie Windows należy zwrócić uwagę, czy włączone jest pokazywanie rozszerzeń plików znanych typów. W tym celu, w oknie Eksploratora Windows z paska narzędzi wybierz opcję *Narzędzia/ Opcje Folderów*, następnie wybierz zakładkę *Widok* i upewnij się, że usunięte zostało zaznaczenie opcji *Ukryj rozszerzenia znanych typów plików*. Jeśli opcja ta jest włączona, wtedy wykonanie kroku 3. z poprzedniego ćwiczenia spowoduje utworzenie pliku *klasa1.java.txt*, co, rzecz jasna, nie jest naszym celem.

Proces tworzenia pliku kodu źródłowego jest zawsze taki sam — znajduje się on w podkatalogu */src* katalogu projektu (np. *c:\WTK22\apps\projekt1\src*). Kod, który trzeba zawrzeć w pliku *klasa1.java*, znajduje się poniżej (oczywiście, bez numerów linii).

Ć W I C Z E N I E

2.3 Program wypisze na ekranie emulatora tekst „Witaj świecie!”

Napisz program, który wyświetli na ekranie podany w kodzie tekst. Dodatkowo, okienko programu będzie miało określony tytuł.

```
1: import javax.microedition.lcdui.*;
2: import javax.microedition.midlet.MIDlet;
3:
4: public class klasa1 extends MIDlet
5: {
6:     public klasa1()
7:     {
8:         Form formatka = new Form("Pierwszy MIDlet");
9:         formatka.append("Witaj świecie!");
10:        Display ekran = Display.getDisplay(this);
11:        ekran.setCurrent(formatka);
12:    }
13:    public void startApp()
14:    {
15:    }
16:    public void pauseApp()
17:    {
18:    }
19:    public void destroyApp(boolean unconditional)
20:    {
21:    }
22: }
```

Można zauważyć pewne różnice między powyższym kodem a opisanym wcześniej schematem aplikacji. Przede wszystkim, z pakietu `javax.microedition.midlet` zaimportowaliśmy tylko klasę `MIDlet`. Kolejną różnicę widać w definicjach metod — zamiast słowa kluczowego `protected`, widzimy słowo `public`. W naszym przypadku zastosowanie obydwu wariantów nie ma znaczenia — różnica między nimi została opisana w Dodatku. Najistotniejszym fragmentem kodu są oczywiście wiersze 8 – 11, gdyż to w nich zawarte są kolejne etapy tworzenia interfejsu graficznego. W wierszu 8. tworzymy pojemnik na inne graficzne komponenty, czyli formę, zwaną też formatką lub formularzem. Następnie do formatki dodajemy tekst. W tym momencie dysponujemy przygotowaną do wyświetlenia formą. Aby ją jednak wyświetlić, konieczne będzie utworzenie obiektu **menedżera ekranu**, który odpowiada za zarządzanie wyświetlaniem formatek. Każdy `MIDlet` posiada swój obiekt menedżera, który pobieramy w wierszu 10. Wreszcie, w wierszu 11. wyświetlamy przy pomocy metody `setCurrent()` utworzony obiekt formy. Te oraz inne komponenty graficzne zostaną omówione w następnym rozdziale.

Po zapisaniu kodu źródłowego możemy przystąpić do jego kompilacji i uruchomienia przy pomocy poznanych wcześniej poleceń `Build` i `Run`. Korzystając z przycisku opcji, należy wybrać opcję *Launch*. Na rysunku 2.1 widać nasz pierwszy `MIDlet` w akcji.

Rysunek 2.1.

MIDlet w działaniu



W tym momencie tworzenia aplikacji dysponujemy już działającym programem. Nadszedł czas na opublikowanie naszego wiekopomnego dzieła w internecie.

2.3. Publikowanie MIDletu

Istnieją dwie ogólne metody przenoszenia MIDletu z komputera na telefon komórkowy. Pierwsza polega na przesłaniu programu przy użyciu różnego rodzaju kabli (USB) bądź połączeń bezprzewodowych (IrDA) i wykorzystaniu oprogramowania dołączonego do telefonu. Takie rozwiązanie posiada jedną niezaprzeczalną zaletę. Program można wgrywać wiele razy bez ponoszenia kosztów pobierania z internetu. Niestety, opisanie wszystkich możliwych sytuacji przesyłania danych jest trudne, w związku z tym zajmować się będziemy jedynie publikowaniem MIDletu w internecie.



Do wykonania poniższego ćwiczenia niezbędny jest dostęp do serwera, który ma przypisane następujące typy MIME dla rozszerzeń:

`.jad` — `text/vnd.sun.j2me.app-descriptor`

`.jar` — `application/java-archive`

Jeśli opisany poniżej proces zakończy się niepowodzeniem, poproś administratora o ustawienie tych typów MIME. Oczywiście, konto na serwerze musi mieć specjalny katalog (najczęściej o nazwie `www` lub `htdocs`), w którym umieszczone pliki są ogólnie dostępne i mogą być pobierane przez każdego.

W poniższym ćwiczeniu przygotujemy MIDlet do wysłania na serwer.

Ć W I C Z E N I E

2.4 Przygotowanie MIDletu do publikacji w internecie

Zmień ustawienia projektu konieczne do jego prawidłowego działania. Utwórz archiwum JAR.

1. Uruchom środowisko Wireless Toolkit, a następnie otwórz projekt *projekt1*.
2. Kliknij przycisk *Settings* i przejdź na zakładkę *Required*.

3. W polu *MIDlet-Jar-URL* wpisz adres internetowy URL do pliku z archiwum Javy (rozszerzenie *.jar*). W tym celu sprawdź, jaki jest adres internetowy do katalogu, w którym będziesz umieszczał MIDlet (np. <http://www.serwer.org/katalog/>), a następnie dopisz do tej ścieżki nazwę archiwum w postaci *nazwa_projektu.jar* (w naszym przypadku *projekt1.jar*). Można również wypełnić pole *Vendor*, wpisując w nie swoje imię i nazwisko. Jeśli będziesz chciał uruchomić MIDlet na normalnym telefonie, przejdź na zakładkę *API Selection* i z listy *Target Platform* wybierz opcję *MIDP 1.0*. Potwierdź zmiany, klikając *OK*.
4. Utwórz archiwum JAR, wybierając opcję *Create Package* z menu *Project/Package*.

W wyniku powyższych operacji powstały dwa pliki gotowe do wysłania na serwer. Niestety, musimy utworzyć jeszcze jeden plik — stronę internetową zawierającą odnośnik do MIDletu, która jest wymagana przez niektóre modele telefonów, a także przez nasz emulator. Kod HTML strony znajduje się poniżej; należy zapisać go pod nazwą *projekt1.html* i umieścić w podkatalogu */bin* naszego projektu.

```
<html>
<head>
<title>projekt1</title>
</head>
<body>
<a href="http://www.serwer.com/katalog/projekt1.jad">projekt1.jad</a>
</body>
</html>
```

Oczywiście, atrybut *href* odnośnika należy zmodyfikować zgodnie z adresem URL serwera.

Możemy teraz przystąpić do kolejnego ćwiczenia, dzięki któremu będziemy w stanie pobrać i uruchomić MIDlet. W tym celu wykonamy ćwiczenie 2.5. Ze względu na różnorodność programów do obsługi kont FTP i rodzajów serwerów będzie to raczej zestaw wskazówek przydatnych przy wysłaniu MIDletu na serwer.

Ć W I C Z E N I E

2.5 Wysyłanie programu na serwer

Wyślij pliki MIDletu na serwer FTP. Musisz posiadać konto, które umożliwia publikowanie plików wszystkim internautom.

1. Uruchom program, przy pomocy którego zamierzasz wysłać pliki na serwer (np. Total Commander).
2. Połącz się z serwerem i przejdź do katalogu, w którym umieścisz MIDlet (zgodnie z określonymi wcześniej zasadami).
3. Wyślij na serwer pliki: *projekt1.html*, *projekt1.jad*, *projekt1.jar*.
4. Rozłącz się z serwerem.

To już koniec pracy nad pierwszym MIDletem! Nie pozostało nam nic innego, jak wykonanie ćwiczenia 2.6, w którym pobierzemy naszą aplikację przy użyciu programu OTA Provisioning.

Ć W I C Z E N I E

2.6 Pobieranie MIDletu z internetu i uruchamianie w programie OTA Provisioning

Pobierz MIDlet, korzystając z programu OTA Provisioning. Następnie uruchom go i przetestuj działanie.

1. Uruchom program OTA Provisioning, korzystając ze skrótu znajdującego się w *Menu Start*.
2. Wybierz opcję *Apps*, a następnie uruchom program *Install Application*.
3. W polu tekstowym podaj adres URL do pliku **stryony internetowej**. Jeśli podasz adres do pliku JAD, emulator zwróci błąd. Wybierz opcję *Menu* i z listy — opcję *Go*.
4. Dalej postępuj tak, jak w ćwiczeniu 1.4 w krokach 9 – 11 (z uwzględnieniem nazwy MIDletu).

Gratulacje! Jeśli wszystko zrealizowałeś zgodnie z wytycznymi, proces pobierania i instalowania MIDletu powinien zakończyć się powodzeniem, a efekt jego działania powinien być identyczny jak w przypadku

uruchamiania MIDletu przy użyciu WTK. Jeśli jednak występują błędy, upewnij się, że:

- ❑ Adres URL do serwera we wszystkich miejscach jest wpisany prawidłowo,
- ❑ Na serwerze znajdują się wszystkie pliki potrzebne do prawidłowego działania MIDletu,
- ❑ Adres URL jest poprawny (w razie wątpliwości co do jego poprawności, poproś administratora serwera o pomoc).

Trzeba pamiętać o tym, iż nie sztuką jest napisać MIDlet działający w środowisku WTK. Sztuką jest napisać MIDlet, który będzie działał na wszystkich (lub przynajmniej większości) telefonach, czyli urządzeniach docelowych. Najczęściej to właśnie z działaniem telefonów pojawiają się największe problemy. My, programiści — użytkownicy J2ME, rzadko mamy wpływ na sposób implementacji J2ME w poszczególnych modelach telefonów. W związku z tym, jeśli na niektórych (zwłaszcza starszych) telefonach MIDlet nie działa (a teoretycznie powinien), nie zawsze winy należy doszukiwać się po stronie programu. Czasem wynika ona z zainstalowanych w telefonach wersjach Javy.