B.M. Harwani

Android™ na tablecie

Receptury

Najlepsze przepisy dla programistów platformy Android!

Helion

Tytuł oryginału: The Android Tablet Developer's Cookbook

Tłumaczenie: Lech Lachowski

ISBN: 978-83-246-8660-5

Authorized translation from the English language edition, entitled: THE ANDROID TABLET DEVELOPER'S COOKBOOK; ISBN 0321885309; by B.M. Harwani; published by Pearson Education, Inc, publishing as Addison Wesley.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/antare
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem: ftp://ftp.helion.pl/przyklady/antare.zip

Printed in Poland.

Spis treści

O autorze **11**

Wstęp **13**

Część I Techniki interfejsu użytkownika **21**

Rozdział 1 Przegląd aplikacji na tablety z systemem Android **23**

Receptura: wprowadzenie do tabletów z systemem Android 23
Receptura: różnice pomiędzy telefonami Android a tabletami Android 25
Receptura: zapewnianie kompatybilności aplikacji z telefonami i tabletami z systemem Android 26
Receptura: tworzenie urządzeń AVD 27
Receptura: struktura katalogów projektu Android 32
Receptura: konwersja aplikacji z telefonu Android w aplikację na tablet Android 37
Receptura: wymuszanie, aby aplikacja działała jedynie na tabletach ... 48
Receptura: aktywności 49
Receptura: cykl życia aktywności w systemie Android 49
Receptura: rozpoczynanie korzystania z intencji 53
Receptura: przekazywanie danych z jednej aktywności do drugiej 58
Podsumowanie 64

Rozdział 2 Fragmenty **67**

Receptura: wprowadzenie do fragmentów 68
Receptura: cykl życia fragmentu 68
Receptura: tworzenie fragmentów pierwszego planu oraz różnice pomiędzy fragmentami pierwszego planu a fragmentami w tle 70
Receptura: dodawanie i usuwanie fragmentów w przypadku zmiany orientacji urządzenia 78
Receptura: rola klas FragmentManager i FragmentTransaction w obsłudze fragmentów 83
Receptura: tworzenie fragmentów dynamicznie w trakcie wykonywania aplikacji 86

Receptura: implementowanie komunikacji pomiędzy fragmentami 92
Receptura: wyświetlanie opcji za pomocą klasy ListFragment 98
Receptura: wyświetlanie okien dialogowych za pomocą klasy DialogFragment 102
Receptura: konfigurowanie preferencji użytkownika za pomocą klasy PreferenceFragment 109
Podsumowanie 117

Rozdział 3 Paski akcji w działaniu 119
Receptura: różnice pomiędzy menu i paskiem akcji 119
Receptura: przełączanie widoczności paska akcji 120
Receptura: komponenty paska akcji 121
Receptura: wyświetlanie elementów akcji w pasku akcji 121
Receptura: nawigowanie do strony głównej po wybraniu ikony aplikacji 126
Receptura: wyświetlanie widoków akcji w pasku akcji 127
Receptura: wyświetlanie podmenu w pasku akcji 132
Receptura: tworzenie paska zadań z zakładkami 139
Receptura: tworzenie paska akcji z rozwijaną listą 145
Podsumowanie 149

Rozdział 4 Nowe widżety 151
Receptura: wyświetlanie kalendarza w aplikacji Android 151
Receptura: wyświetlanie i wybieranie liczb za pomocą widżetu NumberPicker 154
Receptura: tworzenie stosu obrazów za pomocą widżetu StackView 160
Receptura: wyświetlanie listy opcji za pomocą widżetu ListPopupWindow 165
Receptura: sugerowanie opcji za pomocą widżetu PopupMenu 170
Podsumowanie 172

Część II Zarządzanie zawartością 175

Rozdział 5 Schowek systemowy oraz operacja przeciągnij i upuść 177
Receptura: operacja *przeciągnij i upuść* 177
Receptura: przeciąganie i upuszczanie tekstu 179
Receptura: przeciąganie i upuszczanie obrazów 188
Receptura: wycinanie, kopiowanie i wklejanie tekstu przy wykorzystaniu schowka systemowego 198
Podsumowanie 202

Rozdział 6 Powiadomienia oraz intencje oczekujące 205
Receptura: intencje oczekujące 205
Receptura: rozgłaszanie intencji 207
Receptura: system powiadomień systemu Android 214
Receptura: tworzenie powiadomień 215

Receptura: wykorzystanie klasy Notification.Builder 216
Receptura: pozyskiwanie obiektu klasy NotificationManager 218
Receptura: tworzenie powiadomienia i wykorzystywanie intencji oczekującej w celu rozpoczęcia aktywności 218
Podsumowanie 222

Rozdział 7 Ładowarki 225
Receptura: ładowarki 225
Receptura: dostawca treści 226
Receptura: zastosowanie klasy CursorLoader w celu uzyskania dostępu do informacji przechowywanych przez dostawcę treści Contacts 228
Receptura: Tworzenie niestandardowego dostawcy treści 233
Receptura: wyświetlanie informacji z niestandardowego dostawcy treści 243
Receptura: aktualizowanie i usuwanie informacji przechowywanych w niestandardowym dostawcy treści
Podsumowanie 252

Część III Techniki multimedialne 255

Rozdział 8 Animacje 257
Receptura: typy animacji 257
Receptura: korzystanie z klasy ValueAnimator 259
Receptura: wykorzystanie klasy ObjectAnimator do animowania widoków 267
Receptura: uruchamianie wielu animacji za pomocą klasy AnimatorSet 273
Receptura: animacja poklatkowa 279
Receptura: animacja generująca klatki pośrednie 283
Korzystanie z klasy AlphaAnimation 287
Korzystanie z klasy TranslateAnimation 287
Korzystanie z klasy RotateAnimation 288
Korzystanie z klasy ScaleAnimation 289
Receptura: zastosowanie animacji układu 293
Receptura: gromadzenie i wyświetlanie sekwencji animacji za pomocą klasy AnimationSet 301
Podsumowanie 306

Rozdział 9 Sprzętowa akceleracja grafiki 2D 309
Receptura: akceleracja sprzętowa 309
Receptura: korzystanie z warstw widoku 313
Receptura: poprawa wydajności aplikacji opartych na grafice przy wykorzystaniu klasy SurfaceView 317
Receptura: zastosowanie transformacji z wykorzystaniem klasy TextureView 323
Podsumowanie 326

Rozdział 10 Tworzenie i renderowanie grafiki 327
Receptura: interfejsy API wymagane dla grafiki 327
Receptura: tworzenie i renderowanie prostokąta przy użyciu OpenGL 328
Receptura: zastosowanie kolorów wieloodcieniowych 334
Receptura: rotacja grafiki 337
Receptura: skalowanie grafiki 342
Receptura: przesuwanie grafiki 346
Podsumowanie 349

Rozdział 11 Przechwytywanie audio, wideo i obrazów 351
Receptura: przechwytywanie obrazu z wykorzystaniem wbudowanej intencji 351
Receptura: przechwytywanie obrazu za pomocą kodu Java 356
Receptura: nagrywanie audio z wykorzystaniem wbudowanej intencji 362
Receptura: klasa CamcorderProfile 365
Receptura: klasa MediaRecorder i jej metody 372
Receptura: nagrywanie audio z wykorzystaniem kodu Java 373
Receptura: nagrywanie wideo za pomocą wbudowanej intencji 379
Receptura: nagrywanie wideo z użyciem kodu Java 382
Podsumowanie 389

Część IV Interfejs sieciowy i sprzętowy 391

Rozdział 12 Łączność bezprzewodowa 393
Receptura: wiązanie ze sobą dwóch urządzeń Bluetooth 393
Receptura: ręczne przesyłanie plików z jednego urządzenia na drugie z wykorzystaniem technologii Bluetooth 397
Receptura: łączenie w parę urządzenia Bluetooth z komputerem z systemem Windows 399
Receptura: włączanie lokalnego urządzenia Bluetooth 400
Receptura: wyświetlanie listy powiązanych urządzeń 405
Receptura: przesyłanie plików za pomocą technologii Bluetooth 410
Receptura: standard Wi-Fi 412
Receptura: włączanie i wyłączanie Wi-Fi 414
Receptura: Wi-Fi Direct 418
Podsumowanie 423

Rozdział 13 Rdzenie i wątki 425
Receptura: użyteczność architektury procesorów wielordzeniowych 425
Receptura: użyteczność procesów odzyskiwania pamięci 426
Receptura: wątki 429
Receptura: używanie wielu wątków 432
Receptura: korzystanie z klasy AsyncTask 437
Podsumowanie 442

Rozdział 14 Klawiatury i sensory 443
Receptura: zmiana klawiatury i metody wprowadzania danych w systemie Android 443
Receptura: sensory 445
Receptura: lista sensorów obsługiwanych przez urządzenie 448
Receptura: korzystanie z akcelerometru 450
Receptura: korzystanie z czujnika zbliżeniowego 455
Receptura: korzystanie z żyroskopu 458
Podsumowanie 461

Część V Eksploracja sieci WWW 463

Rozdział 15 JSON 465
Receptura: JSON 465
Receptura: wykorzystywanie obiektu JSONObject do przechowywania informacji 468
Receptura: zagnieżdżanie obiektów JSONObject 471
Receptura: korzystanie z tablicy JSONArray 473
Receptura: korzystanie z klas JsonReader oraz JsonWriter 478
Receptura: wykorzystywanie usług sieciowych JSON w aplikacjach Android 483
Podsumowanie 488

Rozdział 16 Klasa WebView 489
Receptura: klasa WebView i jej metody 489
Receptura: wyświetlanie stron WWW za pomocą kontrolki WebView 491
Receptura: korzystanie z klasy WebViewClient 496
Receptura: korzystanie z klasy WebViewFragment 499
Podsumowanie 507

Część VI Zaawansowane techniki systemu Android 509

Rozdział 17 Obsługa małych ekranów 511
Receptura: czynniki decydujące o obsłudze różnych ekranów i gęstości 511
Receptura: zapewnianie obsługi dla różnych wersji platformy 513
Receptura: wykorzystanie pakietu Android Support Library do zapewnienia obsługi starszych wersji systemu 518
Receptura: dostosowywanie aplikacji do orientacji ekranu za pomocą kotwiczenia kontrolek 524
Receptura: obsługa orientacji ekranu przy użyciu alternatywnych układów 528
Podsumowanie 532

Rozdział 18 Widżety ekranu głównego 535
Receptura: widżety aplikacji oraz widżety ekranu głównego 535
Receptura: metody cyklu życia widżetu aplikacji 538
Receptura: tworzenie widżetów ekranu głównego 539
Receptura: aktualizowanie widżetu ekranu głównego za pomocą kontrolki Button 547
Receptura: zastosowanie klasy AlarmManager do częstej aktualizacji widżetu ekranu głównego 551
Podsumowanie 554

Rozdział 19 Android Beam 555
Receptura: standard NFC 555
Receptura: znaczniki NFC 556
Receptura: struktura wykorzystywana do wymiany informacji za pomocą znaczników NFC 557
Receptura: odczytywanie danych ze znaczników NFC 560
Receptura: zapisywanie danych do znacznika NFC 566
Receptura: korzystanie z funkcji Android Beam 570
Receptura: przesyłanie danych za pomocą funkcji Android Beam 571
Podsumowanie 575

Rozdział 20 Analityka i śledzenie aplikacji 577
Receptura: analizowanie i śledzenie aplikacji 577
Receptura: wykorzystanie biblioteki EasyTracker do śledzenia aplikacji Android 578
Receptura: wykorzystanie narzędzia GoogleAnalytics do śledzenia aplikacji Android 587
Podsumowanie 589

Skorowidz 591

4

Nowe widżety

W tym rozdziale poznasz nowe widżety, które są dostępne od API poziomu 11. Nauczysz się wyświetlać w aplikacji Android kalendarz za pomocą widżetu `CalendarView` oraz zakres liczb przy użyciu widżetu `NumberPicker`. Dowiesz się także, jak wyświetlać stos obrazów, wykorzystując widżet `StackView`. Na koniec nauczysz się wyświetlać listę opcji, stosując widżet `ListPopupWindow`, oraz sugestie za pomocą widżetu `PopupMenu`.

Receptura: wyświetlanie kalendarza w aplikacji Android

Aby wyświetlić kalendarz w aplikacji Android, skorzystasz z widżetu `CalendarView`. Jest to konfigurowalny widżet, który wyświetla i wybiera daty. Domyślnie wyświetlany jest kalendarz na bieżący miesiąc, ale możesz przewinąć do konkretnej daty. Kiedy chcesz wybrać datę, po prostu ją kliknij.

Aby zapoznać się z kalendarzem, utwórz projekt Android o nazwie *CalendarViewApp*. Aplikacja domyślnie będzie wyświetlać kalendarz na bieżący miesiąc. Użytkownik może przewinąć kalendarz, aby zobaczyć daty z innego, wybranego miesiąca. Wybrana data będzie wyświetlana za pomocą komunikatu `Toast`. Aplikacja będzie również zawierać kontrolkę `Button`; po jej kliknięciu pokaże się okno dialogowe *DatePickerDialog*, umożliwiające użytkownikowi wyświetlenie kalendarza na wybrany miesiąc.

Ponieważ w Twojej aplikacji potrzebne są zarówno przycisk, jak i kalendarz, musisz w pliku układu aktywności zdefiniować kontrolkę `Button` oraz widżet `CalendarView`. Po zdefiniowaniu tych dwóch elementów plik układu aktywności *activity_calendar_view_app.xml* będzie wyglądał tak, jak przedstawiono w listingu 4.1.

Listing 4.1. Kod wpisany w pliku aktywności activity_calendar_view_app.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent" >
    <Button android:id="@+id/date_picker_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Otwórz kontrolkę daty"
        android:textSize="@dimen/text_size" />
    <CalendarView
        android:id="@+id/calendar_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Dla celów dostępu i identyfikacji w kodzie Java kontrolkom `Button` i `CalendarView` zostały przypisane odpowiednio identyfikatory `date_picker_button` i `calendar_view`. Teraz musisz napisać kod Java, który będzie:

- wyświetlał widżet `CalendarView` zdefiniowany w pliku układu aktywności,
- wiązał nasłuchiwacz zdarzeń `setOnClickListener` z kontrolką `Button` w celu wyświetlenia okna dialogowego *DatePickerDialog*,
- wiązał nasłuchiwacz `OnDateSetListener` z oknem dialogowym *DatePickerDialog*, by możliwe było wyświetlenie kalendarza dla wybranej daty za pomocą widżetu `CalendarView`,
- wiązał nasłuchiwacz zdarzeń z widżetem `CalendarView`, żeby wyświetlać na ekranie wybraną datę.

Aby zrealizować poniższe zadania, wpisz w pliku aktywności Java *CalendarViewAppActivity.java* kod przedstawiony w listingu 4.2.

Listing 4.2. Kod wpisany w pliku aktywności Java CalendarViewAppActivity.java

```
package com.androidtablet.calendarviewapp;

import android.os.Bundle;
import android.app.Activity;
import android.widget.CalendarView;
import android.widget.CalendarView.OnDateChangeListener;
import android.widget.Toast;
import java.util.Calendar;
import android.app.DatePickerDialog;
import android.widget.DatePicker;
import android.widget.Button;
import android.view.View;
import android.view.View.OnClickListener;

public class CalendarViewAppActivity extends Activity {
    private CalendarView calendarView;
    private int yr, mon, dy;
    private Calendar selectedDate;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_calendar_view_app);
        Calendar c = Calendar.getInstance();
        yr = c.get(Calendar.YEAR);
        mon = c.get(Calendar.MONTH);
        dy = c.get(Calendar.DAY_OF_MONTH);
        Button datePickerButton = (Button) findViewById(
            R.id.date_picker_button);
        calendarView = (CalendarView) findViewById(
            R.id.calendar_view);
        datePickerButton.setOnClickListener(new
            OnClickListener() {
            public void onClick(View v) {
                new DatePickerDialog(CalendarViewAppActivity.
                    this, dateListener, yr, mon, dy).show();
            }
        });
        calendarView.setOnDateChangeListener(new
            OnDateChangeListener() {
            @Override
            public void onSelectedDayChange(CalendarView view,
                int year, int month, int dayOfMonth) {
                Toast.makeText(getApplicationContext(),"Wybrałeś datę
                    "+dayOfMonth+"."+(month+1)+"."+ year, Toast.LENGTH_SHORT). show();
            }
        });
    }
    private DatePickerDialog.OnDateSetListener dateListener =
        new DatePickerDialog.OnDateSetListener() {
        public void onDateSet(DatePicker view, int year, int
            monthOfYear, int dayOfMonth){
            selectedDate=Calendar.getInstance();
            yr=year;
            mon=monthOfYear;
            dy=dayOfMonth;
            selectedDate.set(yr, mon, dy);
            calendarView.setDate(selectedDate.getTimeInMillis());
        }
    };
}
```

Jak możesz zauważyć w powyższym kodzie, dostęp do widżetu `CalendarView` uzyskiwany jest z pliku układu, a sam widżet mapowany jest na obiekt `calendarView` klasy `CalendarView`. Ponadto dostęp do kontrolki `Button` o identyfikatorze `date_picker_button` jest uzyskiwany z pliku układu, a sama kontrolka mapowana na obiekt `datePickerButton` klasy `Button`. Nasłuchiwacz `setOnClickListener` został skojarzony z kontrolką `Button`, a jego metoda wywołania zwrotnego `onClick` jest wykonywana po kliknięciu tej kontrolki. W metodzie wywołania zwrotnego `onClick` wywoływane jest okno dialogowe *DatePickerDialog* w celu wyświetlenia bieżącej daty.

Nasłuchiwacz `OnDateSetListener` jest powiązany z oknem dialogowym kontrolki daty (ang. *Date Picker*), więc kiedy jakaś data zostanie wybrana w tym oknie dialogowym, widżet `CalendarView` będzie wyświetlał kalendarz na dany miesiąc i rok.

Nasłuchiwacz `setOnDateChangeListener` jest powiązany z widżetem `CalendarView`. Kiedy jakaś data zostaje wybrana lub zmieniona w tym widżecie, wywoływana jest metoda wywołania zwrotnego `onSelectedDayChange()`. Wykorzystując tę metodę, wyświetlasz wybraną datę za pomocą komunikatu `Toast`. Należy pamiętać, że miesiące liczone są od 0, więc przed wyświetleniem wartości danego miesiąca należy dodać do jego liczby 1.

Po uruchomieniu tej aplikacji zobaczysz, że widżet `CalendarView` wyświetla kalendarz na bieżący miesiąc (patrz rysunek 4.1, górny obrazek). Aby wyświetlić kalendarz na żądany miesiąc, wybierz przycisk *Otwórz kontrolkę daty*, który otwiera okno dialogowe *DatePickerDialog*. W tym oknie dialogowym możesz wybierać datę z kalendarza (patrz rysunek 4.1, środkowy obrazek). Po wybraniu daty i kliknięciu *Gotowe* wyświetlony zostanie kalendarz dla tej daty. Data wybrana z widżetu `CalendarView` jest wyświetlana za pomocą komunikatu `Toast`, co pokazano na rysunku 4.1 (dolny obrazek).

Receptura: wyświetlanie i wybieranie liczb za pomocą widżetu NumberPicker

Podczas czytania tej receptury nauczysz się wyświetlać widżet `NumberPicker`, który pokazuje liczby z określonego przedziału. Liczba wybrana z widżetu `NumberPicker` jest wyświetlana za pomocą kontrolki `TextView`. Utwórz nowy projekt Android o nazwie *NumberPickerApp*.

W tej aplikacji będziesz chciał po prostu wyświetlić kontrolkę `TextView` i widżet `NumberPicker`. Widżet `NumberPicker` będzie wyświetlał liczby z określonego przedziału, a kontrolka `TextView` — liczbę wybraną z widżetu `NumberPicker`. Aby zdefiniować kontrolkę `TextView` i widżet `NumberPicker`, wpisz w pliku układu aktywności *activity_number_picker_app.xml* kod przedstawiony w listingu 4.3.

Listing 4.3. Kod wpisany w pliku układu aktywności activity_number_picker_app.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:text="Wybierz liczbę z widżetu NumberPicker"
        android:id="@+id/numberview"
        android:textSize="@dimen/text_size"
        android:textStyle="bold" />
    <NumberPicker android:id="@+id/numberpicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Rysunek 4.1. Widżet CalendarView wyświetlający kalendarz na bieżący miesiąc (górny obrazek). Okno dialogowe DatePickerDialog otwarte po kliknięciu przycisku Otwórz kontrolkę daty (środkowy obrazek). Widżet CalendarView wyświetlający kalendarz dla daty wybranej z DatePicker (dolny obrazek)

Jak widać, kontrolka `TextView`, do której przypisany został identyfikator `numberview`, jest inicjowana w celu wyświetlenia tekstu `Wybierz liczbę z widżetu NumberPicker`. Tekst wyświetlany za pomocą kontrolki `TextView` będzie prezentowany pogrubioną czcionką o rozmiarze zdefiniowanym w zasobie wymiarów `text_size`. W celu uzyskania dostępu do kodu Java i identyfikacji tego kodu widżetowi `NumberPicker` należy przypisać identyfikator `numberpicker`.

W głównym pliku aktywności Java musisz wpisać kod, który będzie wykonywał następujące zadania.

- Będzie uzyskiwał dostęp do kontrolki `TextView` i widżetu `NumberPicker` z pliku układu i mapował je na odpowiednie obiekty.
- Będzie ustawiał maksymalne i minimalne wartości liczbowe, które mają być wyświetlane za pomocą widżetu `NumberPicker`.
- Będzie powiązywał z widżetem `NumberPicker` nasłuchiwacz zdarzeń, który ma nasłuchiwać, czy zmienia się bieżąca wartość w tym widżecie.
- Będzie wyświetlał za pomocą kontrolki `TextView` liczbę wybraną z widżetu `NumberPicker`.

Aby wykonać powyższe zadania, wpisz w pliku aktywności Java *NumberPickerAppActivity.java* kod przedstawiony w listingu 4.4.

Listing 4.4. Kod wpisany w pliku aktywności Java NumberPickerAppActivity.java

```
package com.androidtablet.numberpickerapp;

import android.os.Bundle;
import android.app.Activity;
import android.widget.NumberPicker;
import android.widget.TextView;

public class NumberPickerAppActivity extends Activity {
    TextView numberView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_number_picker_app);
        numberView = (TextView)findViewById(R.id.numberview);
        NumberPicker numberPicker = (NumberPicker) findViewById(R.id.numberpicker);
        numberPicker.setMaxValue(100);                                          #1
        numberPicker.setMinValue(0);                                            #2
        numberPicker.setWrapSelectorWheel(true);
        numberPicker.setOnValueChangedListener( new NumberPicker.
            OnValueChangeListener() {
            @Override
            public void onValueChange(NumberPicker picker, int
                oldVal, int newVal) {
                numberView.setText("Wybrałeś liczbę "+
                    newVal);
            }
```

```
        });
    }
}
```

Jak możesz zauważyć, z pliku układu uzyskiwany jest dostęp do kontrolki `TextView` o identyfikatorze `numberview`, a sama kontrolka jest mapowana na obiekt `TextView` o nazwie `numberView`. Analogicznie z pliku układu uzyskiwany jest dostęp do widżetu `NumberPicker` o identyfikatorze `numberpicker`, a sam widżet jest mapowany na obiekt `NumberPicker` o nazwie `numberPicker`. Minimalne i maksymalne wartości, które maja być wyświetlane za pomocą widżetu `NumberPicker`, zostały ustawione odpowiednio na `0` i `100`.

Metoda `setWrapSelectorWheel()` ma ustawioną wartość `true`, aby kółko selektora obejmowało minimalne i maksymalne wartości, które są wyświetlane za pomocą widżetu `NumberPicker`. Kiedy zakres wartości (czyli wartość maksymalna – wartość minimalna) wyświetlany za pomocą widżetu `NumberPicker` jest większy niż wartość liczbowa wyświetlana w kółku selektora, obejmowanie zakresu jest włączone domyślnie. (Kółko selektora obejmuje maksymalne i minimalne wartości domyślnie).

Nasłuchiwacz `setOnValueChangedListener` jest powiązany z widżetem `NumberPicker`. Kiedy w tym widżecie zmienia się bieżąca wartość, wywoływana jest metoda wywołania zwrotnego `onValueChange`. W tej metodzie za pomocą kontrolki `TextView` wyświetlana jest nowo wybrana liczba z widżetu `NumberPicker`.

Po uruchomieniu danej aplikacji kontrolka `TextView` będzie wyświetlać użytkownikowi komunikat tekstowy o treści `Wybierz liczbę z widżetu NumberPicker`. Widżet ten pokazuje przypisaną wartość minimalną w możliwej do edycji formie. Im mniejsza wartość pokazana powyżej, tym większą wartość widać poniżej (patrz rysunek 4.2, górny obrazek). Możesz zmienić liczbę, przewijając w górę lub w dół i klikając mniejszą lub większą wartość pokazaną powyżej lub poniżej. Kiedy klikniesz wybraną liczbę, jest ona wyświetlana za pomocą kontrolki `TextView`, tak jak pokazano na rysunku 4.2 (dolny obrazek).

Za pomocą widżetu `NumberPicker` możesz wyświetlić dowolny zakres wartości. W celu wyświetlenia np. wartości nieparzystych z zakresu od 1 do 19, możesz zamienić w listingu 4.4 instrukcje `#1` i `#2` na następujący kod:

```
String[] stringArray = new String[10];
int n=1;
for(int i=0; i<10; i++){
    stringArray[i] = Integer.toString(n);
    n+=2;
}
numberPicker.setMaxValue(stringArray.length-1);
numberPicker.setMinValue(0);
numberPicker.setDisplayedValues(stringArray);
```

Rysunek 4.2. Widżet NumberPicker wyświetlający liczby od ustalonej wartości minimalnej (górny obrazek) oraz wybrana liczba wyświetlona za pomocą kontrolki TextView (dolny obrazek)

Jak widać, zdefiniowana została tablica `String` o nazwie `stringArray`, w której przechowywane są wartości 1, 3, 5... 19. Wartość `min` widżetu `NumberPicker` jest ustawiona na 0. Wartość `max` tego widżetu ma być równa długości `stringArray -1`, ponieważ chcesz wyświetlić wszystkie elementy tablicy `stringArray`. Następnie za pomocą metody `setDisplayedValues()` wartości z tablicy `stringArray` wyświetlane są przez widżet `NumberPicker`.

Ponieważ bieżący motyw w danej aplikacji Android wywodzi się z motywu `Theme_Holo` lub `Theme_Holo_Light`, widżet `NumberPicker` wygląda tak, jak pokazano na rysunku 4.2 (czyli bieżąca wartość może być edytowana za pomocą mniejszej lub większej wartości wyświetlonej odpowiednio powyżej i poniżej widżetu `NumberPicker`). Jeśli zmienisz motyw swojej aplikacji, możesz zmienić wygląd widżetu `NumberPicker`. Przykładowo poniższe instrukcje zastosowane w pliku *AndroidManifest.xml* spowodują, że bieżący motyw aplikacji wywodził się będzie z motywu `Theme`.

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Black.NoTitleBar" >
```

Przedstawione instrukcje zmienią motyw aplikacji na `Theme.Black.NoTitleBar` i dlatego zmianie ulegnie wygląd widżetu `NumberPicker`. Innymi słowy, widżet `NumberPicker` wyświetli w edytowalnej postaci bieżącą wartość wraz z przyciskami zwiększania i zmniejszania wartości odpowiednio powyżej i poniżej (patrz rysunek 4.3, górny obrazek). Zmieniona wartość bieżąca zostanie wyświetlona za pomocą kontrolki `TextView`, tak jak pokazano na rysunku 4.3 (dolny obrazek).

Rysunek 4.3. Po zmianie motywu aplikacji widżet NumberPicker z czarnym tłem, przyciskami zwiększenia i zmniejszania wartości (górny obrazek) oraz wybrana liczba wyświetlona za pomocą kontrolki TextView (dolny obrazek)

Receptura: tworzenie stosu obrazów za pomocą widżetu StackView

Widżet `StackView` pomaga aranżować elementy w formie stosu kart, w którym znajdująca się na wierzchu karta może zostać przełożona i odsłoni kartę leżącą pod nią. W stos, poza obrazami, możesz układać również obiekty składające się tekstu i innych danych.

Czytając tę recepturę, nauczysz się układać w stos obrazy w widżecie `StackView`. Utwórz projekt Android o nazwie *StackViewApp*. Jedyną kontrolką do zdefiniowania w pliku układu aktywności jest widżet `StackView`. Po zdefiniowaniu tego widżetu plik układu aktywności *activity_stack_view_app.xml* będzie wyglądał tak, jak pokazano w listingu 4.5.

Listing 4.5. Kod wpisany w pliku układu aktywności activity_stack_view_app.xml

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <StackView
        android:id="@+id/stackview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:animateLayoutChanges="true">
    </StackView>
</FrameLayout>
```

Dla celów identyfikacji i uzyskiwania dostępu do widżetu `StackView` w kodzie Java widżetowi przydzielony został identyfikator `stackview`. Wartość atrybutu `android:animateLayoutChanges` została ustawiona na `true`, więc zmiany pojawiające się w układzie nie będą przeszkadzały w uruchomieniu klasy `LayoutTransition`.

Aby reprezentować element stosu, który chcesz układać w widżecie `StackView`, musisz w folderze *res/layout* zdefiniować plik XML. Kliknij prawym przyciskiem myszy folder *res/layout* w oknie Package Explorer i dodaj plik XML o nazwie *item.xml*. Ponieważ chcesz układać w stos jedynie obrazy, w pliku *item.xml* zdefiniowana zostanie tylko kontrolka `ImageView`. Po zdefiniowaniu tej kontrolki będzie wyglądał tak, jak przedstawiono w listingu 4.6.

Listing 4.6. Kod wpisany w pliku item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <ImageView
        android:id="@+id/imageview"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"
        android:src="@drawable/ic_launcher" />
</FrameLayout>
```

Możesz zauważyć, że kontrolka `ImageView`, której przypisano identyfikator `imageview`, jest inicjowana w celu wyświetlenia pliku *ic_launcher.png*. Ogólnie rzecz biorąc, chcesz wyświetlić pięć obrazów za pomocą kontrolki `StackView`. Te pięć obrazów to tutaj pliki o nazwach *prod1.png*, *prod2.png*, *prod3.png*, *prod4.png* oraz *prod5.png*; skopiuj je do folderów *res/drawable*. Nadszedł czas, aby w pliku aktywności Java wpisać kod, który będzie:

- uzyskiwał dostęp do widżetu `StackView` z pliku układu i mapował ten widżet na obiekt `StackView`,
- definiował tablicę zawierającą identyfikatory zasobów dla obrazów, które skopiowałeś do folderów *res/drawable*. Tablica będzie działać jako źródło danych, dostarczając obrazy, które chcesz wyświetlać,
- definiował niestandardowy adapter o nazwie `ImageAdapter` rozszerzający klasę abstrakcyjną `BaseAdapter` w celu zdefiniowania zawartości, która ma być wyświetlona za pomocą kontrolki `StackView`,
- wyświetlał zawartość adaptera (obrazy) za pomocą `StackView`; wykorzystując metodę `setAdapter()`, będzie ustawiał adapter `ImageAdapter` dla obiektu `StackView`.

Aby wykonać wymienione zadania, w pliku aktywności Java *StackViewAppActivity.java* wpisz kod przedstawiony w listingu 4.7.

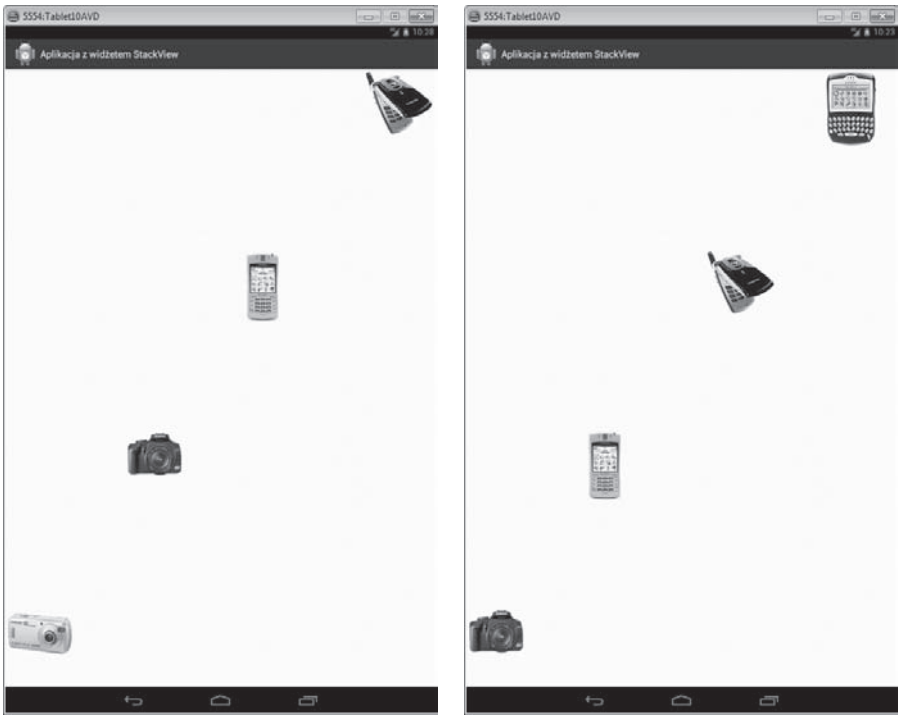
Listing 4.7. Kod wpisany w pliku aktywności Java StackViewAppActivity.java

```
package com.androidtablet.stackviewapp;

import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.StackView;
import android.widget.BaseAdapter;

public class StackViewAppActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_stack_view_app);
        StackView stackView = (StackView)this.findViewById(
            R.id.stackview);
        stackView.setAdapter(new ImageAdapter(this));
    }

    public class ImageAdapter extends BaseAdapter {
        private Context contxt;
```

```
        Integer[] images = {
            R.drawable.prod1,
            R.drawable.prod2,
            R.drawable.prod3,
            R.drawable.prod4,
            R.drawable.prod5
        };
        public ImageAdapter(Context c) {
            contxt = c;
        }

        public int getCount() {
            return images.length;
        }

        public Object getItem(int position) {
            return position;
              }

        public long getItemId(int position) {
            return position;
        }

        public View getView(int position, View view, ViewGroup
            parent) {
            if (view == null) {
                LayoutInflater vi = (LayoutInflater)
                    getBaseContext().getSystemService(
                    Context.LAYOUT_INFLATER_SERVICE);
                view = vi.inflate(R.layout.item, null, false);
            }
            ImageView imageView = (ImageView) view.findViewById(
                R.id.imageview);
            imageView.setImageResource(images[position]);
            return view;
        }
    }
}
```

Adapter `ImageAdapter` jest przypisany do kontrolki `StackView`, aby mogła ona uzyskiwać dostęp do metod tego adaptera w celu wyświetlania zawartości (obrazów). Metody adaptera — `getCount()`, `getItem()` oraz `getItemId()` — są wykorzystywane do określenia liczby obrazów, które mają być wyświetlone, oraz unikatowego identyfikatora konkretnego obrazu. Metoda `getView()` jest stosowana do pobrania właściwego widoku lub obrazu w określonej pozycji. Uzyskiwany jest dostęp do kontrolki `ImageView` zdefiniowanej w pliku *item.xml*, a kontrolka ta wykorzystywana do wyświetlenia obrazów za pomocą `StackView`.

Po uruchomieniu tej aplikacji zobaczysz stos elementów, tu obrazów (patrz rysunek 4.4, lewy obrazek). Kiedy przełożysz obrazek znajdujący się na froncie, obrazki znajdujące się dalej przesuną się ku przodowi, tak jak pokazano na rysunku 4.4 (prawy obrazek).



Rysunek 4.4. Widżet StackView wyświetlający obrazy (lewy obrazek) oraz ukryte obrazy wyświetlone z przodu po przełożeniu frontowych obrazów (prawy obrazek)

Po uruchomieniu tej aplikacji na telefonie rozmiar obrazów może być odpowiedni. Jednak na ekranie tabletu obrazy będą bardzo małe. Aby skalować obrazy zgodnie z rozmiarem ekranu urządzenia, musisz zmodyfikować plik *item.xml*. Otwórz plik *item.xml* znajdujący się w folderze *res/layout* i zmodyfikuj go według listingu 4.8. Zmodyfikowane zostały jedynie fragmenty kodu zaznaczone pogrubioną czcionką. Reszta pozostaje taka sama jak w listingu 4.6.

Listing 4.8. Kod wpisany w pliku item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <ImageView
        android:id="@+id/imageview"
        android:layout_width="@dimen/image_width"
        android:layout_height="@dimen/image_height"
        android:src="@drawable/ic_launcher" />
</FrameLayout>
```

Obraz (lub obrazy), który będzie wyświetlany za pomocą widżetu `StackView`, ma przypisaną szerokość i wysokość za pomocą zasobów wymiarów odpowiednio `image_width` i `image_height`.

Aby zdefiniować zasoby wymiarów `image_width` i `image_height`, otwórz plik *dimens.xml* znajdujący się w folderze *res/values*. Zakładamy, że plik wymiarów *dimens.xml* istnieje już w folderze *res/values* Twojej aplikacji. Zakładamy również, że zistnieją w folderze *res* dwa foldery o nazwach *values-sw600dp* i *values-sw720dp* i oba te foldery zawierają plik wymiarów o nazwie *dimens.xml*.

Aby zdefiniować szerokość i wysokość dla aplikacji uruchamianej na telefonie, otwórz plik *dimens.xml* znajdujący się w folderze *res/values* i wpisz w nim następujący kod.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="image_width">100dp</dimen>
    <dimen name="image_height">200dp</dimen>
</resources>
```

Jak możesz zauważyć, na telefonie widżet `StackView` będzie wyświetlał obrazy o szerokości i wysokości odpowiednio 100 dp i 200 dp.

Następnie w celu zdefiniowania szerokości i wysokości dla obrazów w aplikacji uruchamianej na 7-calowym tablecie otwórz plik wymiarów *dimens.xml* znajdujący się w folderze *res/values-sw600dp* i wpisz w nim następujący kod.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="image_width">200dp</dimen>
    <dimen name="image_height">300dp</dimen>
</resources>
```

Powyższy kod będzie przydzielał szerokość 200 dp oraz wysokość 300 dp obrazom wyświetlanym za pomocą widżetu `StackView` na 7-calowych tabletach. W celu zdefiniowania wymiarów obrazów dla 10-calowych tabletów otwórz plik wymiarów *dimens.xml* znajdujący się w folderze *res/values-sw720dp* i wpisz w nim następujący kod.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="image_width">300dp</dimen>
    <dimen name="image_height">400dp</dimen>
</resources>
```

Powyższy kod sprawi, że w aplikacji uruchamianej na 10-calowych tabletach obrazy w widżecie `StackView` będą miały szerokość 300 dp i wysokość 400 dp.

Po uruchomieniu tej aplikacji na 10-calowym tablecie widżet `StackView` będzie wyglądał tak, jak pokazano na rysunku 4.5 (lewy obrazek). Porównując rysunek 4.5 z rysunkiem 4.4 (górny obrazek), możesz zauważyć, że na tablecie obrazy są większe i wyraźniejsze. Kiedy przełożysz frontowy obrazek, obrazy znajdujące się z tyłu przesuną się do przodu, co zostało pokazane na rysunku 4.5 (prawy obrazek).

Rysunek 4.5. Widżet StackView wyświetlający powiększone obrazy (po lewej). Obrazy z tyłu przesuwają się do przodu, kiedy przekładasz frontowy obraz (po prawej)

Receptura: wyświetlanie listy opcji za pomocą widżetu ListPopupWindow

Możesz użyć widżetu `ListPopupWindow`, aby zakotwiczyć go w widoku hosta i wyświetlić listę opcji. Po przeczytaniu tej receptury będziesz umiał kotwiczyć widżet `ListPopupWindow` w kontrolce `EditText`. Kiedy użytkownik kliknie kontrolkę `EditText`, pojawi się widżet `ListPopupWindow` wyświetlający listę opcji. Po wybraniu przez użytkownika opcji z `ListPopupWindow` opcja ta zostanie przypisana do kontrolki `EditText`. Utwórz nowy projekt Android o nazwie *ListPopupWindowApp*.

Skoro chcesz zakotwiczyć widżet `ListPopupWindow` w kontrolce `EditText`, zdefiniuj tę kontrolkę w pliku układu. Po jej zdefiniowaniu plik układu aktywności *activity_list_popup_window_app.xml* będzie wyglądał tak, jak przedstawiono w listingu 4.9.

Listing 4.9. Kod wpisany w pliku układu aktywności activity_list_popup_window_app.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/product_name"
        android:hint="Wprowadź nazwę produktu"
        android:textSize="@dimen/text_size" />
</LinearLayout>
```

Jak możesz zauważyć, kontrolce `EditText` został przypisany identyfikator `product_name`. W tej aplikacji będziesz w kontrolce `EditText` prosił użytkownika o wpisanie nazwy produktu. W kontrolce wyświetlany jest tekst `Wprowadź nazwę produktu`. Tekst ten będzie prezentowany czcionką o rozmiarze zdefiniowanym w zasobie wymiarów `text_size`.

Domyślny rozmiar elementów listy wyświetlanych w kontrolce `ListView` jest odpowiedni dla telefonów, ale za mały dla tabletów. Aby zmienić rozmiar elementów listy kontrolki `ListView` zgodnie z rozmiarem ekranu danego urządzenia, dodaj w pliku *res/layout* jeszcze jeden plik XML o nazwie *list_item.xml*. W tym pliku wpisz kod przedstawiony w listingu 4.10.

Listing 4.10. Kod wpisany w pliku list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="6dp"
    android:textSize="@dimen/text_size"
    android:textStyle="bold" />
```

Zgodnie z powyższym kodem elementy listy kontrolki `ListView` zostaną oddzielone spacjami o szerokości 6 dp i będą wyświetlane pogrubioną czcionką o rozmiarze zdefiniowanym w zasobie wymiarów `text_size`.

Następnie musisz napisać kod Java, który wykona następujące zadania.

- Będzie uzyskiwał dostęp do kontrolki `EditText` z pliku układu i mapował ją na obiekt `EditText`.
- Będzie definiował obiekt widżetu `ListPopupWindow`.
- Będzie definiował adapter `ArrayAdapter` i wiązał go z listą produktów, które chcesz wyświetlić za pomocą widżetu `ListPopupWindow`.
- Będzie kojarzył adapter `ArrayAdapter` z widżetem `ListPopupWindow` w celu wyświetlenia listy produktów zdefiniowanej w tym adapterze.
- Będzie ustawiał wysokość i szerokość widżetu `ListPopupWindow`.

- Będzie przypisywał naturę modalną do widżetu `ListPopupWindow` (co oznacza, że kontrolka nie będzie wracać do elementu wywołującego, dopóki widżet `ListPopupWindow` nie zostanie zwolniony). Widżet `ListPopupWindow` może zostać zwolniony przez wybranie produktu z `ListPopupWindow` lub kliknięcie dowolnego obszaru poza widżetem `ListPopupWindow`.
- Będzie kotwiczył `ListPopupWindow` w kontrolce `EditText`.
- Będzie wiązał nasłuchiwacz `setOnItemClickListener` z kontrolką `EditText`, aby po kliknięciu tej kontrolki otwierał widżet `ListPopupWindow` pokazujący listę produktów.
- Będzie ustawiał klasę aktywności w taki sposób, aby implementowała nasłuchiwacz `OnItemClickListener`. Wybrana z `ListPopupWindow` opcja ma być przydzielana do kontrolki `EditText`.

Aby wykonać wymienione zadania, wpisz w głównym pliku aktywności Java *ListPopupWindowAppActivity.java* kod przedstawiony w listingu 4.11.

Listing 4.11. Kod wpisany w pliku aktywności Java ListPopupWindowAppActivity.java

```
package com.androidtablet.listpopupwindowapp;

import android.os.Bundle;
import android.app.Activity;
import android.widget.ListPopupWindow;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView;
import android.view.View.OnClickListener;

public class ListPopupWindowAppActivity extends Activity
    implements OnItemClickListener {
    EditText productName;
    ListPopupWindow listPopupWindow;
    String[] products={"Aparat", "Laptop", "Zegarek","Smartfon", "Telewizor"};

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_popup_window_app);
        productName = (EditText) findViewById(
            R.id.product_name);
        listPopupWindow = new ListPopupWindow(
            ListPopupWindowAppActivity.this);
        listPopupWindow.setAdapter(new ArrayAdapter(
            ListPopupWindowAppActivity.this,
            R.layout.list_item, products));
        listPopupWindow.setAnchorView(productName);
        listPopupWindow.setWidth(300);
```

```
        listPopupWindow.setHeight(400);
        listPopupWindow.setModal(true);
        listPopupWindow.setOnItemClickListener(
            ListPopupWindowAppActivity.this);
        productName.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                listPopupWindow.show();
            }
        });
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        productName.setText(products[position]);
        listPopupWindow.dismiss();
    }
}
```

Jak możesz zauważyć, w powyższym kodzie użyty został adapter `ArrayAdapter`, który działa jak źródło danych dla widżetu `ListPopupWindow`. Adapter `ArrayAdapter` wykorzystuje kontrolkę `TextView` do reprezentowania w widoku widoków potomnych (czyli wyświetla elementy tablicy `products` za pomocą kontrolki `TextView`). Wykorzystany wcześniej konstruktor `ArrayAdapter` składa się z niżej wymienionych elementów.

- `ListPopupWindowAppActivity.this` — bieżący kontekst.
- `R.layout.list_item` — wskazuje kontrolkę `TextView`, którą zdefiniowałeś w pliku *list_item.xml*. Kontrolka `TextView` będzie wykorzystana do wyświetlenia każdego z elementów w widżecie `ListPopupWindow`. Elementy tablicy `products` są pakowane w widok przed tym, zanim zostaną przypisane do danego widżetu w celu wyświetlenia. Dlatego też `R.layout.list_item` po prostu zamienia ciągi zdefiniowane w tablicy `products` w kontrolkę `TextView` w celu wyświetlenia w widżecie `ListPopupWindow`.
- `products` — działa jako źródło danych.

Po uruchomieniu tej aplikacji otrzymujesz kontrolkę `EditText` z komunikatem `Wprowadź nazwę produktu` (patrz rysunek 4.6, górny obrazek). Kliknij tę kontrolkę, a wyświetlony zostanie widżet `ListPopupWindow` z listą produktów (patrz rysunek 4.6, środkowy obrazek). Produkt wybrany z `ListPopupWindow` pojawi się w kontrolce `EditText` (patrz rysunek 4.6, dolny obrazek).

Rysunek 4.6. Kontrolka EditText z prośbą o podanie nazwy produktu (górny obrazek). Pokazujący dostępne opcje widżet ListPopupWindow, który został wyświetlony po kliknięciu kontrolki EditText (środkowy obrazek). Produkt wybrany z ListPopupWindow pojawia się w kontrolce EditText (dolny obrazek)

Receptura: sugerowanie opcji za pomocą widżetu PopupMenu

Widżet `PopupMenu` wyświetla menu w modalnym wyskakującym okienku (ang. *pop-up window*). Możesz zakotwiczyć go w widoku i wykorzystać do wyświetlania wymaganych elementów menu lub opcji. W tej recepturze zakotwiczymy widżet `PopupMenu` w kontrolce `EditText` w celu wyświetlania sugestii podczas wpisywania danych w tej kontrolce. Różnica pomiędzy poprzednią recepturą a tą jest taka, że lista opcji jest wyświetlana za pomocą widżetu `PopupMenu`, a nie `ListPopupWindow`.

Utwórz nowy projekt Android o nazwie *PopupMenuApp*. Ponieważ chcesz zakotwiczyć `PopupMenu` w kontrolce `EditText`, należy ją zdefiniować w pliku układu *activity_popup_menu_app.xml*, wykorzystując kod przedstawiony w listingu 4.12.

Listing 4.12. Kod wpisany w pliku układu aktywności activity_popup_menu_app.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/product_name"
        android:hint="Wprowadź nazwę produktu"
        android:textSize="@dimen/text_size" />
</LinearLayout>
```

Jak możesz zauważyć, kontrolka `EditText`, której przypisano identyfikator `product_name`, jest skonfigurowana do wyświetlania komunikatu `Wprowadź nazwę produktu`.

Elementy menu lub opcje dla widżetu `PopupMenu` będziesz definiował w pliku XML. Innymi słowy, menu będzie wypełniane za pomocą pliku XML. Do folderu *res/menu* dodaj plik XML o nazwie *popupmenu.xml*. Ponieważ chcesz wyświetlić nazwy produktów w formie sugestii w kontrolce `EditText`, musisz zdefiniować elementy menu w postaci nazw produktów w pliku *popupmenu.xml*. Te elementy menu są definiowane w pliku *popupmenu.xml* w sposób przedstawiony w listingu 4.13.

Listing 4.13. Kod wpisany w pliku popupmenu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:id="@+id/group_popupmenu">
        <item android:id="@+id/camera"
            android:title="Aparat"
            android:textSize="@dimen/text_size" />
        <item android:id="@+id/laptop"
            android:title="Laptop"
            android:textSize="@dimen/text_size" />
        <item android:id="@+id/watch"
```

```
            android:title="Zegarek"
            android:textSize="@dimen/text_size" />
        <item android:id="@+id/smartphone"
            android:title="Smartfon"
            android:textSize="@dimen/text_size" />
        <item android:id="@+id/television"
            android:title="Telewizor"
            android:textSize="@dimen/text_size" />
    </group>
</menu>
```

Jak możesz zauważyć, produkty `Aparat`, `Laptop`, `Zegarek`, `Smartfon` i `Telewizor` są zdefiniowane jako elementy menu w pliku *popupmenu.xml*. Każdej nazwie produktu przypisany został również unikatowy identyfikator.

Musisz teraz napisać kod Java wykonujący zadania, takie jak:

- uzyskanie dostępu do kontrolki `EditText` zdefiniowanej w pliku układu i zmapowanie jej na obiekt `EditText`,
- zdefiniowanie obiektu `PopupMenu` i wypełnienie elementów menu lub nazw produktów zdefiniowanych w pliku *popupmenu.xml* w celu wyświetlenia za pomocą widżetu `PopupMenu`,
- powiązanie nasłuchiwacza `setOnClickListener` z kontrolką `EditText` w celu nasłuchiwania zdarzeń kliknięcia w tej kontrolce,
- wyświetlenie `PopupMenu`, kiedy użytkownik kliknie kontrolkę `EditText`,
- powiązanie nasłuchiwacza `setOnMenuItemClickListener` z `PopupMenu`,
- przypisanie elementów menu (produktów) do kontrolki `EditText`, kiedy któryś z nich zostanie wybrany z `PopupMenu`.

Aby wykonać wymienione zadania, wpisz w głównym pliku aktywności Java *PopupMenuAppActivity.java* kod przedstawiony w listingu 4.14.

Listing 4.14. Kod wpisany w pliku aktywności Java PopupMenuAppActivity.java

```
package com.androidtablet.popupmenuapp;

import android.os.Bundle;
import android.app.Activity;
import android.widget.EditText;
import android.view.View.OnClickListener;
import android.view.View;
import android.widget.PopupMenu;
import android.view.MenuItem;

public class PopupMenuAppActivity extends Activity {
    EditText productName;
    PopupMenu popupMenu;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_popup_menu_app);
        productName = (EditText) findViewById(
            R.id.product_name);
        popupMenu = new PopupMenu(PopupMenuAppActivity.this,
            productName);
        popupMenu.getMenuInflater().inflate( R.menu.popupmenu,
            popupMenu.getMenu());
        productName.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                popupMenu.setOnMenuItemClickListener(new
                    PopupMenu.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem
                        item) {
                        productName.setText(item.toString());
                        return true;
                    }
                });
                popupMenu.show();
            }
        });
    }
}
```

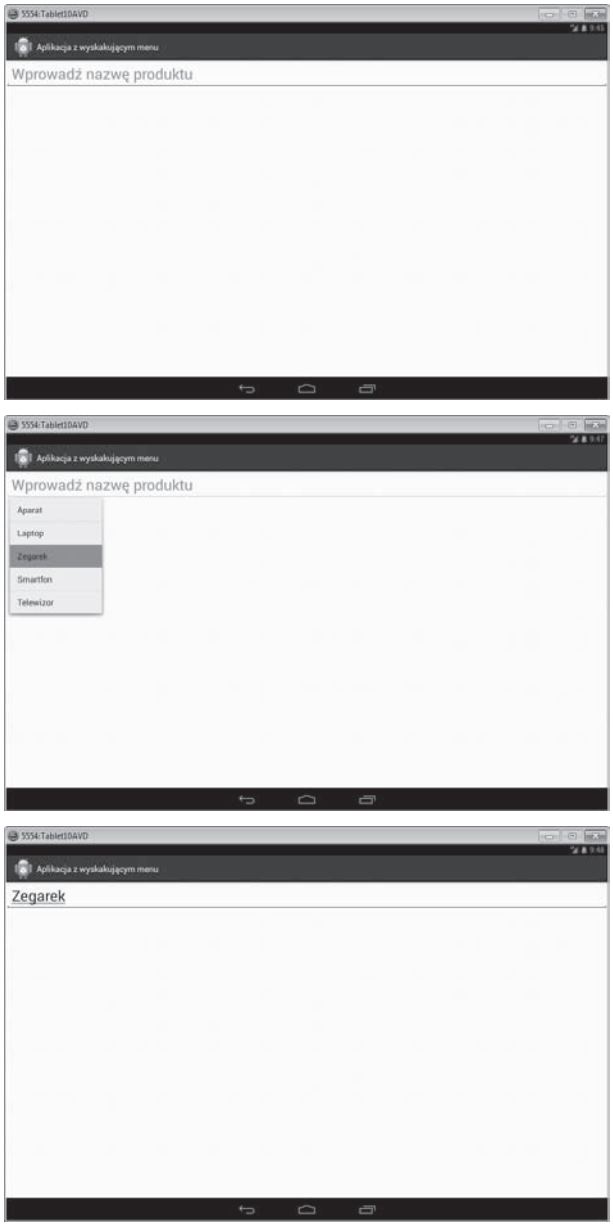
Po uruchomieniu tej aplikacji najpierw pojawia się kontrolka `EditText`. Wyświetla ona komunikat z prośbą, aby użytkownik wprowadził nazwę produktu (patrz rysunek 4.7, górny obrazek). Kiedy użytkownik kliknie kontrolkę `EditText`, pojawi się widżet `PopupMenu` wyświetlający nazwy produktów w formie elementów menu (patrz rysunek 4.7, środkowy obrazek). Użytkownik wybiera z `PopupMenu` produkt, który jest przypisywany do kontrolki `EditText` (patrz rysunek 4.7, dolny obrazek).

Możesz zauważyć, że `PopupMenu` pojawia się pod widokiem kotwicy (kontrolką `EditText`), ponieważ pod tą kontrolką znajduje się dużo miejsca. Jeśli nie byłoby wystarczająco dużo wolnej przestrzeni pod kontrolką, widżet `PopupMenu` pojawiłby się ponad widokiem kotwicy.

Podsumowanie

W tym rozdziale nauczyłeś się wyświetlać w aplikacji Android kalendarz za pomocą widżetu `CalendarView` i zobaczyłeś, w jaki sposób wyświetlana jest data wybrana z tego kalendarza. Dowiedziałeś się również, jak wyświetlać zakres liczb za pomocą widżetu `NumberPicker`. Poznałeś procedurę wyświetlania stosu obrazów przy wykorzystaniu widżetu `StackView`. Na koniec nauczyłeś się wyświetlać listę opcji za pomocą widżetu `ListPopupWindow` oraz wyświetlać sugestię przy użyciu widżetu `PopupMenu`.

Kolejny rozdział koncentruje się na przedstawieniu klas `ClipData` i `DragEvent`. Dowiesz się, czym jest schowek systemowy oraz poznasz procedurę **przeciągania i upuszczania** tekstu i obrazów.



Rysunek 4.7. Kontrolka EditText z prośbą o podanie nazwy produktu (górny obrazek). Pokazujący dostępne opcje widżet PopupMenu, który został wyświetlony po kliknięciu kontrolki EditText (środkowy obrazek). Produkt wybrany z PopupMenu pojawia się w kontrolce EditText (dolny obrazek)

Skorowidz

A

ActionBar, *Patrz:* pasek akcji
activity, *Patrz:* aktywność
adapter
  ArrayAdapter, 145, 166, 168
  baseadapter, 145
  ImageAdapter, 162
  NfcAdapter, 563
  SimpleCursorAdapter, 145
adres
  URI, 232, 237, 251
  URL, 489, 491, 496
ADT, 32
akceleracja
  sprzętowa, 309, 323, 324
    włączanie, 310
    wyłączanie, 310
akcelerometr, 445, 446, 447, 450, 461
akcja
  element, 119, 121, 123
    dodawanie, 124
    wyświetlanie, 121
  MediaStore.ACTION_IMAGE_CAPTURE, 351, 352, 354
  MediaStore.ACTION_VIDEO_CAPTURE, 379
  MediaStore.Audio.Media.RECORD_SOUND_ACTION, 362, 363, 365
  pasek, *Patrz:* pasek akcji
  widok, 121
    niestandardowy, 127
    zwijanie, 128
aktywność, 49, 68, 429
  cykl życia, 49, 51
  danych przekazywanie, 58, 59
  inicjowanie, 50, 53, 54
  konfiguracyjna widżetu, 542
  kończenie, 50, 53
  na pierwszym planie, 50
  nazwa, 34
  niszczenie, 50
  stos, *Patrz:* stos aktywności
  tytuł, 121
  w tle, 50
  zawieszanie, 50, 52
algorytm szeregowania wątków, *Patrz:* wątek planista
Allocation Tracker, 426
Android Beam, 555, 570, 572
  przesyłanie danych, 571
Android Developer Tools, *Patrz:* ADT
Android Support Library, 511, 518, 521
Android Virtual Device, *Patrz:* AVD
Androida wersja, 513
animacja, 257, 258, 267
  alfa, 283, 287
  długość, 258
  generowania klatek pośrednich, 258, 283, 286
  interpolator, *Patrz:* interpolator
  liczba powtórzeń, 258
  odwracanie, 257, 259
  poklatkowa, 258, 279
  powtarzanie, 257, 259
  przesunięcia, 283, 287, 297
  restart, 259
  rotacji, 283, 288, 289
  sekwencja, *Patrz:* animacja złożona
  skalowania, 283, 289
  układu, 258, 293, 296
  widoku, 257, 258
  właściwości, 257
  złożona, 273, 275, 301
API, 327, 393
  OpenGL ES, *Patrz:* OpenGL ES
  poziom uruchomienia aplikacji, 221, 310, 514, 515, 518
aplikacja
  analiza, 577
  asynchroniczna, 225
  ekranu głównego, 535
  logo, 119, 121, 126

aplikacja
personalizowanie, 109
strona główna, 126
śledzenie, 577
Widget Preview, 542
wyłącznie dla tabletów, 48
app widget, *Patrz:* widżet aplikacji
application programming interface, *Patrz:* API
atrybut
checkable, 136
checkableBehavior, 137
match_parent, 27
minSdkVersion, 120, 514, 517, 520, 566
requiresSmallestWidthDp, 48
resizeMode, 541
showAsAction, 132
targetSdkVersion, 514, 520, 566
updatePeriodMillis, 541
windowActionBarOverlay, 120
wrap_content, 27
audio, 362, 372
kodowanie, 365, 372
nagrywanie, 363, 365, 373
przechwytywanie, 365, 372
AVD, 27

B

background fragment, *Patrz:* fragment w tle
biblioteka
EasyTracker, 577, 578, 582, 585
json.org, 465
bitmapa, *Patrz też:* obraz
alternatywna, 513
Bluetooth, 393
łączenie urządzeń, 394
łączenie z komputerem z systemem Windows, 399
przesyłanie plików, 397, 410
zasięg, 556
broadcast receiver, *Patrz:* odbiornik rozgłoszeniowy

C

clip, *Patrz:* wycinek
ContentResolver, 242, 251
czujnik
ciśnienia atmosferycznego, 446
grawitacji, 446
odległości, 446
orientacji, 445
oświetlenia, 446, 455
pola magnetycznego, 445, 446
przyspieszenia liniowego, 446
siły ciężkości, 461
temperatury, 446
wilgotności, 446
zbliżeniowy, 443, 445, 455
czytnik kart elektronicznych, 556

D

Dalvik, 426
Dalvik Debug Monitor Service), *Patrz:* DDMS
DDMS, 426, 427
debuger, 427
density, *Patrz:* gęstość
dialog
asynchroniczny, 102
modalny, 103, 109
synchroniczny, 102
dostawca
kontaktów, 232
treści, 225, 226, 227
Browser, 227
CallLog, 227
Contacts, 225, 227, 228
dodawanie wiersza, 242
funkcjonalność, 237
Media Store, 227
niestandardowy, 233, 234, 243, 246, 248, 252
Settings, 227
utrzymywanie zawartości, 246, 248
dp, 27, 512
drag and drop, *Patrz:* operacja przeciągnij i upuść
drop zone, *Patrz:* strefa upuszczania

E

ekran, 68, 511
duży, 26, 37, 48, 512
ekstraduży, 26, 37, 48, 280, 512
gęstość, *Patrz:* gęstość
mały, 26, 512
normalny, 26, 512
orientacja, 524, 525, 526, 527, 528, 529, 531
preferencji, 109
rozdzielczość pikselowa, *Patrz:* rozdzielczość pikselowa
szerokość, 68
najmniejsza, 38
wysokość, 68
element
rysowalny, 513
supports-screens, 48
explicit intent, *Patrz:* intencja jawna

F

flat color, *Patrz:* kolor jednoodcieniowy
folder
  assets, 36
  bin, 36
  gen, 34
  gen/com.androidtablet.firstandroidtabletapp/R.java, 35
  res, 36
  res/anim, 281
  res/drawable-hdpi, 36
  res/drawable-ldpi, 36
  res/drawable-mdpi, 36
  res/drawable-xhdpi, 36
  res/drawable-xxhdpi, 36
  res/layout, 36, 529
  res/layout/activity_first_android_tablet_app.xml, 36
  res/layout-land, 529
  res/layout-large/, 37
  res/layout-sw600dp/, 37
  res/layout-sw720dp/, 38
  res/layout-xlarge/, 37
  res/menu, 36
  res/values, 36, 236
  res/values-sw600dp, 236, 280
  res/values-sw720dp, 236, 280
  res/values-v11, 37
  res/values-v14, 37
  src, 34
  src/com.androidtablet.firstandroidtabletapp, 34
  src/com.androidtablet.firstandroidtabletapp/FirstAndroidTablet
  ↪AppActivity.java, 34
foreground fragment, *Patrz:* fragment pierwszego planu
format
  JSON, *Patrz:* JSON
  NDEF, *Patrz:* NDEF
fragment, 67, 68, 512
  cykl życia, 68
  DialogFragment, 103
  dołączanie do aktywności, 70
  interfejs użytkownika, 67
  komunikacja, 94
  ListFragment, 98, 99
  menedżer, *Patrz:* menedżer fragmentów
  nieaktywny, 70
  niewidoczny
    dla użytkownika, 70
  niszczenie, 70
  odłączanie
    od aktywności, 70
  pierwszego planu, 70
  PreferenceFragment, 109
    wyświetlanie przy użyciu kodu, 110
    wyświetlanie za pomocą pliku XML, 109
  stos, *Patrz:* stos fragmentów
  tworzenie, 70, 78
    dynamiczne, 85, 86
    statyczne, 86
  usuwanie, 78
  w tle, 70
  widoczny
    dla użytkownika, 70
  widoku tworzenie, 70
  zapisywanie, 70

G

garbage collector, *Patrz:* pamięć odzyskiwanie
gęstość, 25, 36, 511, 512
  hdpi, 25, 513
  ldpi, 513
  mdpi, 25, 513
  xhdpi, 25, 513
  xxhdpi, 513
Google Analytics, 578, 587
Google Analytics SDK, 577, 579
Google Nexus 7, 10, 23, 24
Google Play, 572
GPU, 309, 327
grafika, 309, 327
  2D, 327
  3D, 327
  optymalizacja, 382
  przesuwanie, 346
  renderowanie, 327, 328, 339
  rotacja, 337, 339, 340, 342
  skalowanie, 342
graphical user interface, *Patrz:* GUI
GUI, 309, 317

H

home screen widget, *Patrz:* widżet ekranu głównego

I

implicit intent, *Patrz:* intencja niejawna
intencja, 49, 53, 351
  dane, 207
    rozszerzone, 60
  filtr, 561
  jawna, 53
  nasłuchiwanie, 205, 208
  niejawna, 54
  oczekująca, 205, 206, 218, 548
  PendingIntent, 548, 553
  regularna, 205
  rozgłaszanie, 205, 207, 209, 212
  wiązka dodatkowa, 563

intent, *Patrz:* intencja
interfejs
  CharSequence, 198
  definiowanie, 90
  LoaderManager.Loader ↪Callbacks, 225
  programowania aplikacji, *Patrz:* API
  SpinnerAdapter, 145
  SurfaceHolder, 321
  SurfaceHolder.Callback, 317, 356, 357, 382
  TextureView.Surface ↪TextureListener, 323
  TypeEvaluator, 260
  użytkownika, *Patrz:* UI
  graficzny, *Patrz:* GUI
  fragmentu, *Patrz:* fragment interfejs użytkownika
  kontrolka, 26, 40, 70
  View.OnDragListener, 178
Internet, 490
interpolator, 257, 296, 300
  accelerate_interpolator, 298
  AccelerateDecelerate ↪Interpolator, 300
  AccelerateInterpolator, 300, 305
  AnticipateInterpolator, 300
  AnticipateOvershoot ↪Interpolator, 300
  BounceInterpolator, 300
  CycleInterpolator, 300
  DecelerateInterpolator, 300
  LinearInterpolator, 300
  liniowy, 296
  OvershootInterpolator, 300

J

JavaScript Object Notation, *Patrz:* JSON
jednostka dp, 27, 512
JSON, 465
  przechowywanie informacji, 468
  tablica, *Patrz:* tablica JSONArray

K

kalendarz, 151
kamera, 323, 325, 326, 351
  domyślna, 324
karta
  czytnik, 556
  kredytowa, 556
  SD, 360, 375, 379
klasa
  AlarmManager, 551
  AlphaAnimation, 286, 287
  AnimationDrawable, 283
  AnimationSet, 301
  AnimatorSet, 273, 275
  AppWidgetProvider, 536
  AsyncTask, 425, 437, 438
  BluetoothAdapter, 400
  BroadcastReceiver, 208, 211, 536, 538, 545
  Build, 514, 515
  Button, 153
  CalendarView, 153
  CamcorderProfile, 365, 368, 388
  Camera, 356
  ClipboardManager, 198
  ClipData, 178, 198
  ContentResolver, 237, 251
  Context, 208
  CursorLoader, 225, 228
  DialogFragment, 102
  DragEvent, 178
  DragShadowBuilder, 178
  FragmentActivity, 524
  FragmentManager, 83
  FragmentTransaction, 83, 85
  GLSurfaceView, 328
  GLSurfaceView.Renderer, 328
  Handler, 430
  JsonReader, 478, 479
  JsonWriter, 478, 479
  LayoutAnimation ↪Controller, 293, 295, 296
  ListFragment, 98
  LoaderManager, 225
  MediaRecorder, 362, 372, 373, 379, 382
  MyGLSurfRenderer, 336
  Notification, 215
  Notification.Builder, 216
  NotificationManager, 215, 218
  ObjectAnimator, 267
  ObjectAnimator, 258, 267, 271
  PendingIntent, 205, 206
  PreferenceFragment, 109
  RemoteViews, 536
  RotateAnimation, 286, 288, 289
  ScaleAnimation, 286, 289
  SensorManager, 445, 450
  SharedPreferences, 109
  SQLiteQueryBuilder, 237
  SurfaceTexture, 323
  SurfaceView, 309, 317, 323, 356, 357, 382
  TextureView, 323
  TranslateAnimation, 286, 287, 305
  ValueAnimator, 257, 259
  WebSettings, 490
  WebView, 491

WebViewClient, 489, 496
WebViewFragment, 489, 499, 502
WebViewFragment ↪Activity, 502, 503
wewnętrzna, 550
WifiManager, 412, 414
klatka pośrednia, 258
klawiatury konfiguracja, 443
klucz username, 58
kolejka
komunikatów, 430
MessageQueue, 430
kolor
jednoodcieniowy, 327, 334
wieloodcieniowy, 327, 334, 337
komunikacja
małego zasięgu, *Patrz:* NFC
peer-to-peer, 556
komunikat, 536
asynchroniczny, *Patrz:* intencja
dziennika, 130
NDEF, 557, 571
NdefMessage, 557
wymagający uwagi użytkownika, *Patrz:* powiadomienie
konstruktor obiektu JSONObject, 466
kontakty, 227, 228, 231
kontrolka
Button, 55, 58, 62, 107, 109, 151, 153, 199, 209, 210, 235, 236, 242, 248, 249, 261, 285, 352, 500
widżetu, 547
EditText, 109, 165, 170, 235, 236, 248, 249, 250, 496, 500
Fragment, 500
GridView, 188
ImageView, 160, 161, 162, 275, 285, 352
skalowanie, 301
interfejsu użytkownika, *Patrz:* interfejs użytkownika kontrolka
kotwiczenie, 524, 525
ListView, 98, 99, 179, 231
rozmiar elementów, 166
selectedopt TextView, 71
TextView, 44, 47, 55, 62, 71, 98, 107, 140, 141, 154, 275, 492, 500
animacja, 267, 272
animowanie, 262
ToggleButton, 272, 279, 283
WebView, 489, 490, 496, 499, 501, 503
kursor, 225

L

lista opcji, 165, 170
loader, *Patrz:* ładowarka

Ł

ładowarka, 225
wywołanie zwrotne, 226

M

magnetometr, *Patrz:* czujnik pola magnetycznego
maszyna wirtualna
Dalvik, 426
urządzenia fizycznego, 427
mechanizm
ContentResolver, *Patrz:* ContentResolver
menedżer
fragmentów, 83
pakietów, 521
menu, 119
przepełnienia, 121
metoda
adaptera, 162
addCallback, 321
addPreferencesFrom ↪Resource, 113
AsyncTasksAppActivity.java, 439
beginObject, 479
beginTransaction, 85
canGoBack, 490
canGoBackOrForward, 491
canGoForward, 490
clearCache, 491
clearHistory, 491
Context.getSystemService, 445
delete, 237, 251
doInBackground, 438, 440
enableForeground ↪Dispatch, 563
endObject, 479
get, 467, 474
getAction, 178, 208
getActionBar, 120
getActivity, 74, 206
getAddress, 402
getAnimatedValue, 259
getBondedDevices, 407
getBoolean, 115
getCount, 162
getDefaultAdapter, 400
getDefaultSensor, 445, 448, 450
getExtras, 60
getFragmentManager, 83
getHolder, 321
getInt, 115
getItem, 162

getItemId, 134, 162
getLoaderManager, 225
getName, 402
getPrimaryClip, 198
getSensorList, 448
getSettings, 490
getState, 402
getString, 115
getStringExtra, 208
getSurfaceTexture, 323
getSystemService, 198, 218, 412, 418
getType, 237
getWifiState, 414
glClear, 333
glClearColor, 330, 331, 332
glColorPointer, 336
glDrawArrays, 331, 333
glEnableClientState, 332
glLoadIdentity, 341
glMatrixMode, 341
glPopMatrix, 344
glPushMatrix, 344
glRotatef, 341
glScalef, 344
glVertexPointer, 333
goBack, 490
goBackOrForward, 491
goForward, 490
handleMessage, 430
has, 467
hasNext, 479
initLoader, 225, 226
insert, 237
invalidate, 260, 313
isEnabled, 402
isHardwareAccelerated, 310
isNull, 467
isWifiEnabled, 414
join, 474
length, 467, 474
loadURL, 490
name, 479
nextName, 479
nextString, 479
notifyChange, 237
ofFloat, 259, 265, 267, 269
ofInt, 259, 267
ofObject, 259, 267
onAccuracyChanged, 447, 454
onActivityCreated, 70
onAttach, 70
onCancelled, 438
onCreate, 50, 70, 241, 321
onCreateLoader, 226
onCreateView, 70, 73, 104
onDeleted, 538, 542, 544
onDestroy, 50, 70
onDestroyView, 70
onDetach, 70
onDisabled, 538, 542, 544
onDrawFrame, 328, 331, 334
onEditorAction, 495
onEnabled, 538, 542
onItemClick, 81
onKey, 495
onListItemClick, 246, 252
onLoaderReset, 226
onLoadFinished, 226
onNavigationItem ↪Selected, 146
onOptionItemSelected, 126
onOptionSelected, 82, 92
onOptionsItemSelected, 134
onPause, 50, 70, 321
onPauseMySurfaceView, 321, 322
onPostExecute, 438, 440
onPreExecute, 438, 440
onProgressUpdate, 438, 440
onQueryTextChange, 128, 130
onQueryTextSubmit, 128
onReceive, 209, 211, 542
onResume, 50, 70, 321
onSensorChanged, 447, 454
onStart, 50, 70
onStop, 50, 70
onSurfaceChanged, 328, 331, 334
onSurfaceCreated, 328, 330, 334
onSurfaceTexture ↪Available, 325
onSurfaceTexture ↪Destroyed, 325
onSurfaceTextureSize ↪Changed, 325
onSurfaceTexture ↪Updated, 325
onTabSelected, 143
onTabUnselected, 144
onUpdate, 538, 542
peek, 479
PendingIntent.getActivity, 206
PendingIntent.get ↪Broadcast, 206
PendingIntent.getService, 206
play, 273
playSequentially, 273, 277
playTogether, 273, 276
prepare, 383
publishProgress, 440
put, 467, 474
putExtra, 207
query, 237, 251
registerListener, 447
release, 356, 372, 373, 383
reload, 490
remove, 467
replace, 86
sendBroadcast, 206, 208
setAction, 207

setAudioEncoder, 372, 373, 383
setAudioSource, 372, 373, 382
setAutoCancel, 216
setContentIntent, 217
setContentText, 217
setContentTitle, 217
setDisplayShowHome ↪Enabled, 120
setDisplayShowTitle ↪Enabled, 120
setDisplayZoomControls, 490
setDuration, 258
setJavaScriptEnabled, 490
setLayerType, 313
setListNavigationCall ↪backs, 145
setMaxDuration, 383
setMaxFileSize, 383
setName, 402
setNavigationMode, 145
setOutputFile, 372, 373, 383
setOutputFormat, 372, 373, 382
setPreviewDisplay, 356, 372, 383
setPrimaryClip, 198
setRepeatCount, 258
setRepeatMode, 258
setSavePassword, 490
setSmallIcon, 216
setSupportZoom, 490
setTextScaleX, 265
setTextZoom, 490
setTicker, 216
setVideoEncoder, 383
setVideoFrameRate, 382
setVideoSource, 382
setWebViewClient, 503
setWhen, 216
setWifiEnabled, 414
setWrapSelectorWheel, 157
skipValue, 479
start, 283, 372, 373, 383, 385
startActivity, 54, 206
startDiscovery, 407
startDrag, 178, 179
startPreview, 356
startService, 206
statyczna, 267
stop, 283, 372, 373, 383, 385
stopPreview, 356
supportMultipleWindows, 490
surfaceChanged, 358, 384
surfaceCreated, 322, 358, 384
surfaceDestroyed, 358, 384
takePicture, 356
toString, 474
update, 237
value, 479
wywołania zwrotnego onDragEvent, 178
mikrofon, 351, 362
motyw
Theme, 158
Theme.Black.NoTitleBar, 159
Theme.Holo, 120
Theme.Holo.NoAction Bar, 120
Theme_Holo, 158
Theme_Holo_Light, 158

N

nasłuchiwacz zdarzeń, 177
addUpdateListener, 262
przeciągania, 178
sensora, 443, 447, 450
setOnClickListener, 153, 261, 363
setOnEditActionListener, 496
setOnErrorListener, 385
setOnInfoListener, 385
setOnQueryTextListener, 128
setOnValueChanged ↪Listener, 157
setSurface-TextureListener, 325
TabListener, 143
NDEF, 557
near field communication, *Patrz:* NFC
NFC, 555, 556, 560, 570
zasięg, 556
NFC data exchange format, *Patrz:* NDEF
NFC tag, *Patrz:* znacznik NFC
notification, *Patrz:* powiadomienie

O

obiekt
ActionBar, 120
AnimatorSet, 273
BroadcastReceiver, 536, 542
Builder, 273
ClipData, 198, 199
ClipData.Item, 198
ClipDescription, 198
JSONObject, 465
konstruktor, 466
przechowywanie informacji, 468
pusty, 477
serializacja, 468
zagnieżdżanie, 471
pendIntent, 206
SharedPreferences, 115
tworzenie, 58

obraz, 160, 162, 227
  podgląd, 351
  przechwytywanie, 351, 356, 360
  przeciągnij i upuść, 188, 199
  rozmiar, 163, 164, 190
  zapisywanie, 360, 361
obszar powiadomień, 205, 214
odbiornik rozgłoszeniowy, 205, 207, 208, 210, 211, 212, 429
  dodawanie dynamiczne, 212
okno
  DialogFragment, 103, 105, 107
  dialogowe, 103, 105, 107, 109
  LogCat, 52
  modalne, 103, 109, 170
metoda, 127
onResumeMySurfaceView, 321
OpenGL ES, 327, 328
operacja
  kopiuj wklej, 198
  przeciągnij i upuść, 177, 179
    obraz, 188, 199
    tekst, 179
  wytnij, 198

P

pamięć
  alokowana dynamicznie, 426
  odzyskiwanie, 425, 426
para klucz-wartość, 465
pasek
  akcji, 119, 121, 122, 134, 143
    komponenty, 121
    tytuł, 121
    widoczność, 120
    widok niestandardowy, 127
    wyświetlanie podmenu, 132
    z rozwijaną listą, 145, 146
    zakładka, 121, 143
  systemowy, 205, 214
  tytułu, 119
  zadań z zakładkami, 139, 140
pending intent, *Patrz:* intencja oczekująca
piksel, 25
  niezależny od urządzenia, 25
pixel resolution, *Patrz:* rozdzielczość pikselowa
playlista, 379
plik
  .jar, 35
  ActionBarOnOlderApp ↪Activity.java, 516
  activity_action_bar_on_older_app.xml, 515
  activity_action_bar_submenu.xml, 132
  activity_alternate_layout_app.xml, 528
  activity_blue_tooth_paired_list_app.xml, 405
  activity_btfile_transfer_app.xml, 410
  activity_consume_jsonweb ↪service_app.xml, 484
  activity_fragment_on_older_app.xml, 518
  activity_handle_orienta ↪tion_app.xml, 525
  activity_jsonreader_writer_app.xml, 480
  activity_multiple_threads_app.xml, 434
  activity_pref_fragment.xml, 113
  activity_sensor_acc_app.xml, 451
  activity_sensor_gyro ↪scope_app.xml, 458
  activity_web_view_app.xml, 491
  activity_web_view_frag_app.xml, 501
  activity_wi_fi_app.xml, 415
  AlternateLayoutApp ↪Activity.java, 531
  analytics.xml, 582
  AndroidManifest.xml, 27, 37, 64, 83, 127, 233, 514
  audio, 227, 365, 372
  BlueToothAppActivity.java, 402
  BlueToothPairedListApp ↪Activity.java, 407
  BTFileTransferApp ↪Activity.java, 411
  build, 37
  dimens.xml, 37, 67, 236, 276
  główny aktywności Java, 61
  JSONArrayAppActivity.java, 476
  JSONReaderWriterApp ↪Activity.java, 481
  konfiguracyjny, 37
  konfiguracyjny widżetu, *Patrz:* widżet plik konfiguracyjny
  layout.xml, 38
  manifestu, 211
  multimedialny, 227
  MultipleThreadsApp ↪Activity.java, 435
  MyFragmentActivity.java, 519

popupmenu.xml, 170
preferences.xml, 110, 112
Proguard-project.txt, 37
przesyłanie
Bluetooth, 397, 410
ReceiveBroadcastActivity.java, 211
SensorGyroscopeApp↪Activity.java, 459
SensorsListAppActivity.java, 449
strings.xml, 36
ThreadAppActivity.java, 431
układu, 36, 113, 242
dla tabletu, 41
dla telefonu, 41
o orientacji pionowej, 38, 44, 45
o orientacji poziomej, 38
tworzenie, 41
wideo, 227, 365, 372
WifiAppActivity.java, 416
WiFiBroadcastReceiver.java, 422
WiFiDirectAppActivity.java, 420
XML, 27
menu, 36
podmenu w pasku akcji, 132
pole
RF, 555, 556
wyboru, 109
zaznaczania, 136
połączenie nieodebrane, 227
powiadomienie, 205, 215
szuflada, 205, 214
tworzenie, 215, 216, 217, 218
preference view, *Patrz:* widok preferencji
preferencji grupowanie w kategorie, 109
procesor
czterordzeniowy, 425
dwurdzeniowy, 425
graficzny, *Patrz:* GPU
wielordzeniowy, 425
projekt Android, 32
przeglądarka, 490, 491
historia, 227, 490, 491
zakładka, 227
przycisk opcji, 109, 137
dźwięki dzwonka, 109

R

radio frequency identification, *Patrz:* RFID
rekord NdefRecord, 557, 559
RFID, 555
rozdzielczość pikselowa, 512

S

schowek systemowy, 198, 199
SDK, 577
sensor, 443
dokładność, 447
sprzętowy, 445
tryb
próbkowania, 455
przerwania, 455
TYPE_ACCELERO↪METER, 446
TYPE_AMBIENT_TEMPERATURE, 446
TYPE_GRAVITY, 446
TYPE_GYROSCOPE, 446
TYPE_LINEAR_ACCELE↪RATION, 446
TYPE_MAGNETIC_FIELD, 446
TYPE_PRESSURE, 446
TYPE_PROXIMITY, 446
TYPE_RELATIVE_HUMIDITY, 446
TYPE_ROTATION_VECTOR, 446
metoda, 139
sklep Google Play, *Patrz:* Google Play
smooth coloring, *Patrz:* kolor wieloodcieniowy
Software Development Kit, *Patrz:* SDK
stos, 160
aktywności, 49, 94
fragmentów, 94
obrazów, 160
strefa upuszczania, 177, 178
strona WWW, 490, 491
sw, *Patrz:* ekran szerokość najmniejsza
system powiadomień, 214
szuflada powiadomień, *Patrz:* powiadomienie szuflada

T

tablet, 25, 38, 67
tablica JSONArray, 473, 474, 475, 478
telefon, 25, 38, 67
transmisja danych, 465

U

UI, 24, 26, 70, 429, 511
controls, *Patrz:* interfejs użytkownika kontrolka
układ, 68
alternatywny, 513, 528, 529
definiowanie, 37
dla podaktywności, 62
nazwa, 34
plik, *Patrz:* plik układu
wieloekranowy, 26

uniform resource identifier, *Patrz:* zapytanie URI
URI, *Patrz:* zapytanie URI
urządzenie
  Bluetooth, 393, 397, 399, 405, 408
    włączanie, 400, 402
  konfiguracja, 27
  NFC, 555
  PIN, 395
  powiązane, 393, 395, 405
  preferencje, 227
  ustawienia, 227
  wirtualne
    Android, *Patrz:* AVD
user interface, *Patrz:* UI
usługa
  DDMS, *Patrz:* DDMS
  sieciowa JSON, 483

W

warstwa
  LAYER_TYPE_HARD ↪WARE, 313, 314
  LAYER_TYPE_NONE, 313, 315
  LAYER_TYPE_SOFT ↪WARE, 313
  programowa, 309
  sprzętowa, 309
wątek, 432
  główny, 429, 430
  planista, 425, 426
  szeregowanie, *Patrz:* wątek planista
  w tle, 429, 437
wiadomość powitalna, 55
wideo, 372
  kodowanie, 365, 372, 382
  nagrywanie, 379, 382
  przechwytywanie, 351, 365, 372
widok
  AdapterView, 293
  akcji, *Patrz:* akcja widok
  hosta, 165
  preferencji, 109
    CheckBoxPreference, 109, 110, 115
    EditTextPreference, 109, 111, 115
    ListPreference, 109, 111, 112
    Preference, 109
    PreferenceCategory, 109
    PreferenceScreen, 109
    RingtonePreference, 109, 111, 117
    wyświetlanie za pomocą pliku XML, 109
  przeciąganie, 177, 178
  SurfaceView, 382, 385
  warstwa, 313
widżet
  aktualizacja, 551
  aktywność konfiguracyjna, 542
  aplikacji, 535, 536
    cykl życia, 538
    host, 535
  CalendarView, 151, 154
  cykl życia, 542
  definicja, *Patrz:* widżet plik konfiguracyjny
  ekranu głównego, 535, 539
  ikona, 542
  instancja, 536, 538, 545
  kontrolka Button, 547
  ListPopupWindow, 165, 170
  ListView, 71
  NumberPicker, 154, 157
  plik konfiguracyjny, 539
  PopupMenu, 170
  SearchView, 130
  StackView, 160
  usuwanie, 538
  WebView, *Patrz:* kontrolka WebView
  wyboru, 71, 145
wielozadaniowość, 425, 432
Wi-Fi, 393, 412, 414
Wi-Fi Direct, 418
wycinek, 198

Z

zapytanie URI, 227, 228
zasób
  alternatywny, 513
  wymiarów, 236, 276, 280, 285, 318
zaznaczanie i zamiatanie, 426
zdarzenie, 49
  aplikacji, 207
  nasłuchiwanie, 128, 177
  onActivityCreated, 70
  onAttach, 70
  onCreate, 50, 70
  onCreateView, 70
  onDestroy, 50, 70
  onDestroyView, 70
  onDetach, 70
  onPause, 50, 70
  onResume, 50, 70
  onStart, 50, 70
  onStop, 50
  systemowe, 207
znacznik, 555
  NFC, 556, 557, 560, 561
    odczyt danych, 572
    zapisywanie danych, 566, 572
  odczyt, 555
  zapis, 555

Ż

żyroskop, 443, 446, 458
  dokładność, 461

Obowiązkowa lektura każdego programisty!

Android swoją karierę rozpoczął od telefonów. Nie minęło wiele czasu, gdy pojawiły się pierwsze tablety pracujące w tym systemie. Dzisiaj ich wybór i możliwości przyprawiają o zawrót głowy. Deweloper musi obecnie wziąć pod uwagę wiele konfiguracji sprzętowych — różne rozdzielczości ekranu i różnorodny osprzęt wymagają dbałości o detale i przetestowania aplikacji w kilku środowiskach. Jak sobie z tym poradzić? Jak tworzyć rozwiązania działające na różnych tabletach oraz jak wykorzystać potencjał Androida?

Na te i wiele innych pytań odpowie ta rewelacyjna książka, należąca do cieszącej się uznaniem serii „Receptury". Szczególny nacisk położono w niej na tablety pracujące pod kontrolą systemu Android w wersji 4.2.2. Dowiesz się, jak przygotować środowisko pracy, wyświetlać powiadomienia oraz tworzyć animacje. Ponadto zapoznasz się z możliwościami sprzętowej akceleracji grafiki 2D, przechwytywania dźwięku oraz materiałów wideo. A potem opanujesz korzystanie z formatu JSON w celu przechowywania informacji oraz obsługiwanie małych ekranów. Ostatni rozdział książki został poświęcony analizie i śledzeniu sposobu wykorzystania Twojej aplikacji — dzięki temu przekonasz się, do jakich zakątków świata dotarło Twoje dzieło. Książka ta powinna znaleźć się na półce każdego autora aplikacji dla Androida — ogromna baza gotowych do użycia fragmentów kodu ułatwi Ci pracę!

Dzięki tej książce:

- stworzysz urządzenie AVD
- sprawdzisz możliwości OpenGL
- wykorzystasz łączność bezprzewodową
- odkryjesz tajniki standardu NFC

Addison Wesley

cena: 99,00 zł
ISBN 978-83-246-8660-5
9 788324 686605

ALWAYS LEARNING
PEARSON