[1. Wstęp 1](#_Toc392249510)

[1.1. Problematyka i zakres pracy 1](#_Toc392249511)

[1.2. Cele i zakres pracy 2](#_Toc392249512)

[1.3. Przegląd literatury 3](#_Toc392249513)

[1.4. Układ pracy 4](#_Toc392249514)

[2. Podstawowe informacje o Androidzie 6](#_Toc392249515)

[2.1. Historia Androida 6](#_Toc392249518)

[2.2. Android Studio 6](#_Toc392249519)

[2.3. Android Manifest 7](#_Toc392249520)

[2.4. Aktywności 9](#_Toc392249521)

[2.5. Uprawnienia 11](#_Toc392249522)

[2.6. Zasoby aplikacji 12](#_Toc392249523)

# Wstęp

## Problematyka i zakres pracy

W pierwszym kwartale 2014 roku trzy na pięć sprzedaży telefonów komórkowych nosiło nazwę „inteligentny” [1]. Smartphone[[1]](#footnote-1), jak potocznie   
na urządzenia tego typu się mówi, nie tylko zawładnęły niemałym rynkiem telefonów bezprzewodowych, ale także znacznie poprawiły komfort naszego życia. Od obsługi wiadomości SMS, po zarządzanie pocztą elektroniczną,   
do sterowania temperaturą w naszych mieszkaniach – lista rzeczy, które nie potrafią te urządzenia z każdym dniem zmniejszą jeszcze bardziej.

Nic dziwnego, że tym rynkiem, który jest wart miliardy dolarów rocznie, zainteresowały się wielkie firmy. Swoje rozwiązania w tej dziedzinie technologii   
na przestrzeni ostatnich lat pokazały takie formy jak: Microsoft, Apple czy Google. W swojej pracy zapoznam się jedną z części systemu firmy z Mountain View odpowiedzialną za komunikacji miedzy komponentami zainstalowanymi   
na urządzeniu. Dzięki systemowi nadajników i odbiorników program, które będzie odbierał wiadomości SMS nie musi posiadać dodatkowego wątku odpowiedzialnego za sprawdzenie czy wiadomość tekstowa przyszła, tylko potrzebuje wyczekiwać informacji od części systemu odpowiedzialnej   
za odbieranie krótkich wiadomości. Taki sposób rozwiązania komunikacji między aplikacja w Android przysporzył się do znacznego ograniczenia używanych zasobów przez oprogramowanie działające na tym systemie.

Częściom badawczą mojej pracy było wykonanie aplikacji przedstawiającej działanie tego mechanizmu w Androidzie. Program ten wykorzystuje interfejsy programistyczne, który zostały udostępnione przez firmę Google. Sama aplikacja jest napisana w wersji standardowej języka Java. Niemniej jednak sposób, w jaki został zaprojektowany interfejs programistyczny Androida różni się od tego zaproponowanego przez Oracle. Programiści znający już podstawową edycje Javy zaczynający swoją przygodę z systemem od firmy z Mountain View muszą zapoznać się nie tylko z nazwami udostępnionych klas i metod, ale z sposobem tworzenia interfejsu użytkownika czy komunikacji między systemem a aplikacją. Z tego powodu uważam, że moja praca nie tylko zapoznała czytelnika z systemem rozsyłania i odzierania informacji w Androidzie, ale także pokazała   
go z podstawowymi aspektami tworzenia oprogramowania na ten system.

W części teoretycznej zostaną opisanego takie zagadnienia jak:

* Jak korzystać z powyższych interfejsów
* Komponenty systemu Android, z których można pobierać lub wysyłać informacje potrzebne w działaniu programu
* Klasy poboczne (tj. **IntentFilter**), które są potrzebne w korzystaniu   
  z tych interfejsów
* Zasięg działania obiektów typu **Intent** oraz **BroadcastReceiver**

## Cele i zakres pracy

Głównym celem pracy jest przedstawienie w języku polskim mechanizmu komunikacji między komponentami w systemie Android. Postanowiłem wykorzystać moją prace dyplomową do zaprezentowania tego zagadnienia, ponieważ podczas zaczęcia nauki programowania na ten system natrafiłem   
na dużą, dla sporej grupy ludzi, barierę – barierę językową.

Z powodu przedstawionego w poprzednim akapicie, moja praca skupi się   
na opisanie jednych z najważniejszych mechanizmów w systemie od Google, mianowicie na klasach **Intent** oraz **BroadcastReceiver**. Te obiekty ułatwiają komunikacje miedzy komponentami aplikacji. Trudno wyobrazić sobie scenariusz gdzie, mając na uwadze jak najmniejsze zużycie procesora i pamięci operacyjnej, na jednoczesnym istnieniu dwóch programów na tym samym urządzeniu, które korzystają z osobnych wątków do wykonania **tej samej długiej operacji**.

Pomniejszym celem mojej pracy dyplomowej jest przedstawienie także takich części programu pisanego pod Androida jak:

* Application manifest
* Activities
* Zarządzenie i rodzaje zasobów, z których korzystają aplikacje Androidowe
* Android background processing
* Android pemissions

Powyższe komponenty są częścią sporej ilości programów mobilnych. Bez znajomości tych technologii trudno, by było zacząć pisać aplikacje Androidowe jak i opisać tytułowy mechanizm.

## Przegląd literatury

Podstawowym źródłem informacji na temat tworzenia aplikacji na Androida jest jego documentacja [2]. Wiedza ukazania na tej stronie internetowej jest przeżysta oraz napisana łatwym do zrozumienia językiem. Stronie nie tylko zawiera opis API[[2]](#footnote-2) Androida, ale także przykłady zastosowań komponentów   
w niej zawartych [3]. O ile dokumentacja dostarczona przez Google jest bardzo dobrze napisana w większości obszarów, to tak niektóre nie zostały w ogóle wspomniane. Przykładem komponentu, który nie został opisany w tutorialach[[3]](#footnote-3), jest klasa **BroadcastReceiver**. Na szczęście to zagadnie zostało opisane przez osoby spoza firmy z Mountain View [4].

O ile wiedza na temat tworzenia aplikacji działający na Androidzie dostępną w Internecie jest duża i najczęściej darmowa, ale za to jest ona rozsiana po całej sieci. Czas spędzany na przeszukiwaniu Internetu w poszukiwaniu dobrej jakości artykułu na temat nas interesujący jest czasami ogromny, dlatego istnienie książek z tej domeny nauki jest bardzo potrzebny. Jednym z najlepszych podręczników mówiących o programowaniu na Androida mim zdaniem należy wolumen autorstwa Reto Meier [5]. Książka ta to jeden z najlepszych kompendiów wiedzy zawartej jednym podręcznym opakowaniu. Zawarte w min jest wszystko, co musi wiedzieć każdy, kto chce programować na system od Google.

W języku polskim trudno o dobry podręcznik traktujący o tworzeniu aplikacji na Androida. Najbliżej mianu „kompendium wiedzy” jest przetłumaczona książka autorstwa Shane Conder oraz Lauren Darcey [5]. Podręcznik ten zawiera informacje potrzebne do bez problemowego napisania pierwszej poważnej aplikacji na Androida.

## Układ pracy

Tematem pracy jest mechanizm rozsyłania informacji o zdarzeniach   
w systemie Android, a głównym cele mojej pracy jest przedstawienie tego problemu w języku polskim. W pierwszym rozdziale zawiera wstęp i cele pracy dyplomowej.

Drugi rozdział to przedstawienie podstawowych informacji potrzebnych   
do opisu tytułowego mechanizmu. Ta część pracy zawiera między innymi wiedzę na temat:

* Manifestu aplikacji Androidowej
* Zasobów, które używa aplikacja pod Androida
* Uprawnień, które może uzyskać program

W trzecim rozdziale został opisany tytułowy mechanizm. Informacje w nim zawarte dotyczą:

* Tworzenia, używania obiektów klas **Intent** oraz **BroadcastRecaiver**
* Opisu identyfikacji **BroadcastReceivera** przez system.
* Zasięgu nadajnika oraz odbiornika
* Natywnych akcji Androida

Czwarty rozdział to opis programu napisanego na potrzeby pracy. Zostało tan opisane architektura aplikacji oraz przykładowe jej zastosowanie.

Ostatnim rozdziałem jest podsumowanie, w którym zaprezentowano rezultaty pracy. Wynika z nich przede wszystkim, że bez mechanizmu **Intentów** oraz **BroadcastReveiverów** aplikacje pisane pod Androida wymagałyby urządzeń   
z dużo ilością pamięci operacyjnej oraz szybkim procesorem. Głównym rezultatem pracy jest ukazanie wad i zalet mechanizmu rozsyłania informacji   
o zdarzaniach w systemie od firmy Google oraz jego opis w języku polskim.

# Podstawowe informacje o Androidzie



## Historia Androida

W roku 2008 na Mobile World Congress w Barcelonie firma Google pokazała pierwsze prototypowe urządzenie z ich nowym systemem mobilnym na pokładzie nazwanym Android [7]. Był to telefon przypominający popularne   
w tamtym okresie produkty spółki Research In Motion (teraz BlackBerr). Kilka miesięcy później firma HTC wypuściła do sprzedaży pierwszy telefon w tym systemem [8] – HTC Dream, znany bardziej, jako T-Mobile G1.

Od tego czasu Androida działa na ponad 1 miliardzie urządzeń [9], miał kilkanaście dużych wydań oraz jest obecny na prawie co drugim sprzedanym smartphone’ie [10].

Sukces Androida może przypisać modelowi biznesowemu, jaki przyjęła firma z Mountain View:

* System ten jest dostępny, jako projekt open-source, czyli każdy kto tylko ma wiedzie i możliwości może pobrać jego kod i stworzyć własną wersje Androida
* Może działać na prawie każdym urządzeniu wyposażonym   
  w mikroprocesor, na przykład telefony komórkowe, telewizory,   
  a nawet lodówki
* Tworzenie aplikacji na niego jest całkowicie darmowe. Każdy może wejść na stronę dla developerów Androida i pobrać narzędzia potrzebne do tego

## Android Studio

Android Studio to oficjalne **IDE**[[4]](#footnote-4) do tworzenia programów działających pod kontrolą Androida. Składa się z dwóch cześci:

* **SDK[[5]](#footnote-5)** dla systemu Android\
* Intelij IDEA Community Edition, zmodyfikowany przez firmę Google na potrzeby programowanie na ich system. Przykładowy zrzut ekranu przestawiący te program widać na Rysunku 1.

Rysunek 1 Przykładowy zrzut ekranu z Intelij Android Studio

## Android Manifest

Android Manifest to plik XML, który zawiera opis atrybutów aplikacji.   
W nim są zawarte informacje takie jak: nazwa aplikacji, uprawnienia jakie posiada aplikacja czy **BroadcastRecaivery**. Tabela 1 przedstawia przykładowe znaczniki, które są dostępne oraz ich krótki opis.

Tabela Przykładowe znaczniki, któte są dostępne w Android Manifest

|  |  |
| --- | --- |
| **Nazwa znaczka** | **Opis** |
| <action> | Zawiera nazwę akcji, przy której będzie uruchamiany **BroadcastReceiver**. Używany w <intent-filter> |
| <activity> | Deklaruje aktywności dostępne w programie. Jeżeli aktywność nie zostanie opisana przez ten znacznik nie bedzie widoczna dla systemu i nie zostanie wyświetlona. |
| <application> | Definiuje atrybuty aplikacji takie jak: jej nazwa, ikonę, czy może być debatowana, aktywności używane przez aplikacje, serwisy. |
| <intent-filter> | Może być zdefiniowany wewnątrz znaczników *<activity>*, *<service>* oraz *<reciver>*. Odpowiada  za zdefiniowanie akcji, na które wcześniej wymienione komponenty będą mogły odpowiadać. |
| <manifest> | Główny element w pliku XML’owym. |
| <receiver> | Jeden z dwóch sposobów na zdefiniowanie obiektu typu **BroadcastReceiver**. |
| <service> | Używany go deklaracji obkiertów klasy **Service**. Tak jak w przypadku *<activity>*, jeżeli serwis nieopisany tutaj, system nie będzie w wiedział o jego istnieniu. |
| <uses-permission> | Definiuje uprawnienia, które muszą być udzielone aplikacji, by mogła działać poprawnie. |

Powyższy listing przedstawia przykładowy Android Manifest.

|  |
| --- |
| 1 <?xml version=**"1.0"** encoding=**"utf-8"**?>  2 <manifest xmlns:android=**"http://schemas.android.com/apk/res/android"**  3 package=**"org.karolgurecki.autotask"**  4 android:versionCode=**"1"**  5 android:versionName=**"1.0"**>  6 <uses-sdk android:minSdkVersion=**"16"**/>  7 <application android:label=**"@string/app\_name"**  8  android:theme=**"@android:style/Theme.DeviceDefault"**>  9 <activity android:name=**".ui.activities.Main"**  10 android:label=**"@string/app\_name"**>  11<intent-filter>  12<action android:name=**"android.intent.action.MAIN"**/>  13<category android:name=**"android.intent.category.LAUNCHER"**/>  14</intent-filter>  15</activity>  16<activity android:name=**".ui.activities.NewTaskActivity"**  17android:label=**"@string/newTaskButtonText"**/>  18<activity android:name=**".ui.activities.TaskListActivity"**  19android:label=**"@string/taskListButtonText"**/>  20<activity android:name=**".ui.activities.TaskList"**  21android:label=**"Task List"**/>  22<service android:name=**".service.StartUpService"**  23android:label=**"Autotask Service"**/>  24<receiver android:name=**"org.karolgurecki.autotask.**  25 **service.StartUpBroadcastReceiver"**>  26<intent-filter>  27<action android:name=**"android.intent.action.**  **ACTION\_EXTERNAL\_APPLICATIONS\_AVAILABLE"**/>  28<action android:name=**"android.intent.action.**  **BOOT\_COMPLETED"**/>  29</intent-filter>  30</receiver>  31</application>  32<uses-permission android:name=  **"android.permission.READ\_EXTERNAL\_STORAGE"**/>  33<uses-permission android:name=  **"android.permission.WRITE\_EXTERNAL\_STORAGE"**/>  34<uses-permission android:name=  **"android.permission.RECEIVE\_BOOT\_COMPLETED"**/>  35<uses-permission android:name=**"android.permission.BLUETOOTH"**/>  36</manifest>  Listing Przykładowy plik AndrodManifest.xml |

## Aktywności

Za interfejs użytkownika aplikacji w Androidzie odpowiadają obiekty klasy **Activity**. Większość programów posiada wiele luźno ze sobą powiązanych aktywności.

Zwykle aplikacje mają jeden główny obiekt **Activity**, który jest uruchamiany zaraz po jej włączeniu. Aby system wiedział, że dana aktywność jest „główna” musi posiadać **IntentFilter**, który zawiera „android.intent.action.MAIN” jako akcje oraz kategorie „android.intent.category.LAUNCHER”   
– przykład takiego zastosowania widać na listingu pierwszych w liniaka 12 i 13.

Powyższy sposób działa jedynie, gdy chcemy, aby jakaś aktywność była uruchomiona jako pierwsza. Aby wyświetlić inne Activity, na przykład po naciśnięciu przycisku, trzeba utworzyć **Intent** używając obiektu **Class** aktywności, którą chcemy pokazać. Listing 2 przedstawia przykład uruchomienia aktywności z kodu programu.

|  |
| --- |
| 1 Intent intent **=** **new** Intent**(**Main**.this,**TaskListActivity**.**class**);**  2 startActivity**(**intent**);** |

Listing Tworzenie Activity przy pomocy kodu

Każde **Activity** w Androidze ma kilka możliwych stanów, w których może sie znaleść. Sekwecje przechodzenia miedzy nimi widać na rysunku 1.



Rysunek 2 Cykl życia aktywności. Żródło: [11]

Podczas tworzenia nowego klasy, które rozszerza **Activity** musimy napisać własną implemencje metodyonCreate(). Reszta stanów aktywności posiad swoje domyślne definicje, ale nic nie stoi na przeszkodzie, aby programista napisał własne.

Trzeba także wspomnieć, że system posiada tak znany **Back Stack**, czyli stos aktywności, które zostały uruchomiono przez użytkownika. Ta kolekcja jest kolejkowana przy pomocy algorytmu **LIFO[[6]](#footnote-6)**, czyli ostatnia aktywność na stosie jest aktualnie wyświetlana przez system.

## Uprawnienia

Z powodu bezpieczeństwa w Androidzie zastosowanio system uprawnień (ang. *Premissions*). Domyślnie aplikacja ma podstawowe przywileje – może jedynie zarządzać swoimi danymi. Programinista wiedząc, że będzie potrzebować dostępu na przykład do stanu modułu Bluetooth, musi powiadomić o tym fakcie system. Odbywa się to przy pomocy dodania znacznika *<use-permission>*   
w AndroidManifest.xml. Android posiada szeger wbudowanych uprawnień, których przykłady widzimy w tabeli 2.

Tabela Przykładowe uprawnienie, które udostępnia Android

|  |  |
| --- | --- |
| **Nazwna przywileju** | **Opis** |
| android.permission.ACCESS\_WIFI\_STATE | Umożliwia aplikacji dostęp  do informacji na temat sieci WiFi |
| android.permission.BATTERY\_STATS | Pozwala na odczyt stanu baterii. |
| android.permission.BLUETOOTH\_ADMIN | Umożliwia odnadywanie urządzeń przez protokół Bluetooth oraz łaczen ie z nimi. |
| android.permission.INTERNET | Potrzebne, aby program mógł korzystać z zasobów dostępnych w Internecie. |
| android.permission.VIBRATE | Umożliwia do funkcji wibracji  w urządzeniu. |
| android.permission.RECEIVE\_SMS | Pozwala na odsczyt wiadomości SMS, które przychodzą do urządzenia. |

API Androida pozwala także na definiowane własnych uprawnień przez programiste. Pozwała ono do zablokowania dostępu do takich elementów aplikacji jak: serwisy, aktywności czy broadcastReceivery przez inne aplikacje. Aby tego dokonać programista musi:

* dodać znacznik *<permission>* w AndroidManifest.xml z definicją upowarznienia
* deklaracji elementu, który chce zabezpieczyć musi alrybutowi *android:permission* nadać wartość taką samą jak nazwa przywileju, który wcześniej zdefiniował

## Zasoby aplikacji

1. Smartphone (z ang. *Inteligentny telefon*) – urządzenie elektroniczne, które można zmieścić w kieszenie posiadające możliwość wykonywania połączeń głosowych oraz pozwalające na instalowanie i uruchamianie oprogramowania firm trzecich. [↑](#footnote-ref-1)
2. API (skrót od *Application Programming Interface* – z ang. *interfejs programowania aplikacji*) – podprogramy, struktury danych, klasy obiektów, które udostępnia dany program, biblioteka, system operacyjny w celu komunikacji w innym oprogramowaniem. [↑](#footnote-ref-2)
3. Tutorial (z ang. *Samouczek*) – zestaw instrukcji pozwalający na łatwe nauczenie się określonego zagadnienia. [↑](#footnote-ref-3)
4. IDE (skrót od *integrated development environment*, z ang. *zintegrowane środowski programistyczne*) – jedna lub wiele aplikacji pozwalających na edycje, budowanie oraz testowanie kodu programu przez programiste [↑](#footnote-ref-4)
5. SDK (skrót od *Software Development Kit*) – zastaw narzędzi ułatwiający tworzenie programów na dane środowisko [↑](#footnote-ref-5)
6. LIFO (skrót od *last in, firsrt out*) – algorytm kolekowaania elementów na liście jedno wymiarorej. Oznacze, że element, który jako ostatni został dodany do tej kolekcji, będzie zdjęcy jako pierwszy przy próbie pobrania z niej danych. [↑](#footnote-ref-6)