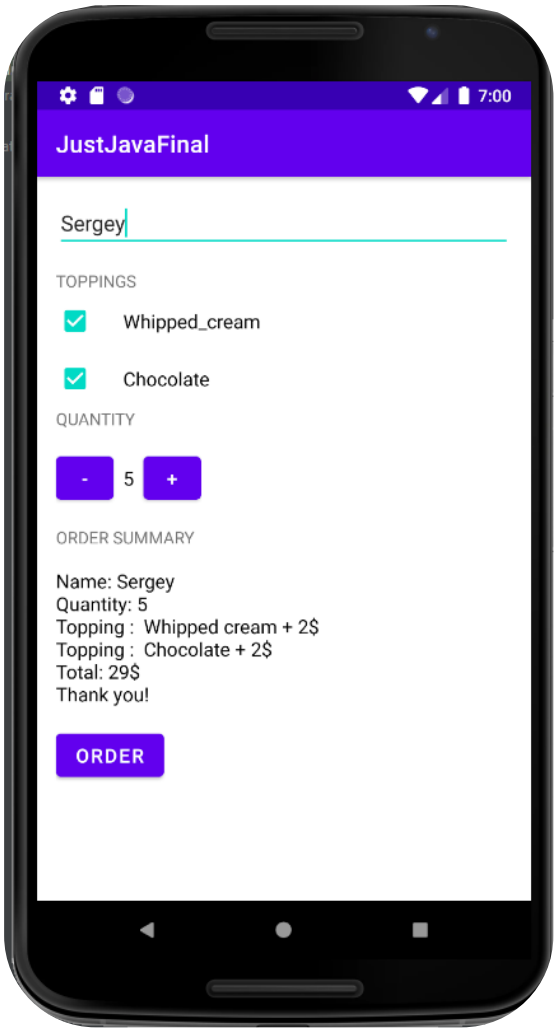
Л7. Елементи управління EditView. Створення обробників подій та прив'язка їх до елементів управління.

Додаємо у додаток JustJava поле для введення імені замовника та після натиснення кнопки Order появляється поле Name (рис. 1).



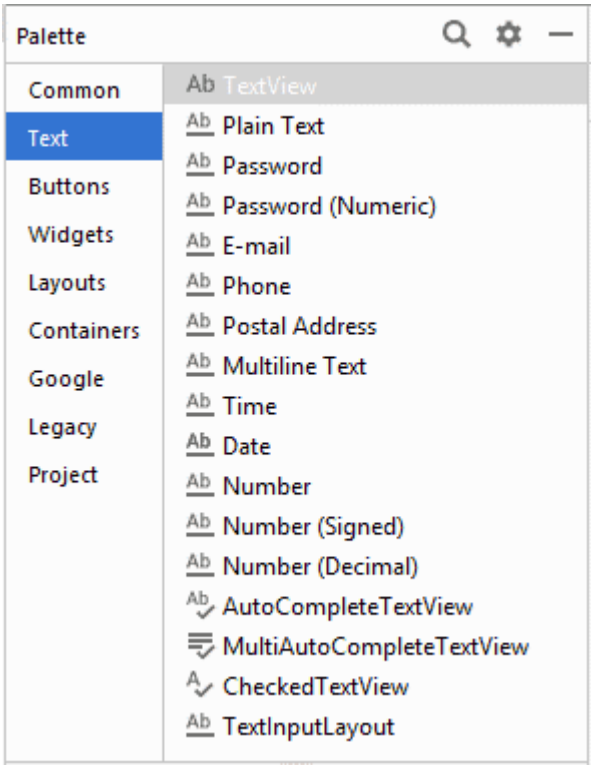


Рисунок 1 – Оновлений дизайн додатку JustJava
Рисунок 2 – Android Studio. Панель інструментів Palette в режимі Design

Компонент **EditText** - це текстове поле, яке використовується, якщо необхідне редагування тексту. Слід зауважити, що **EditText** є спадкоємцем **TextView**.

В Android Studio на панелі інструментів текстові поля можна знайти в категорії **Text** під різними іменами (рис. 2).

Для швидкої розробки текстові поля забезпечили різними властивостями і дали різні імена: Plain Text, Person Name, Password, Password (Numeric), E-mail, Phone, Postal Address, Multiline Text, Time, Date, Number, Number (Signed), NumberDecimal.

Plain Text - найпростіший варіант текстового поля. При додаванні в розмітку його XML-код буде таким:

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10" />
```

Ems - це розмір шрифту. В елементі з 2-дюймовим шрифтом 1em означає 2in. При використанні елемента **Person Name** в XML додається атрибут inputType, який відповідає за вид клавіатури (тільки літери) при введенні тексту.

```
<EditText
    android:id="@+id/editText2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName" />
```

При використанні **Password** в inputType використовується значення textPassword. Під час введення тексту спочатку показується символ, який замінюється на зірочку. Якщо використовується елемент Password (Numeric) , то у атрибута inputType використовується значення **numberPasswor**d. У цьому випадку на клавіатурі будуть тільки цифри замість літер.

```
<EditText
    android:id="@+id/editText2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPassword" />
```

Атрибут **android:inputType="textEmailAddress"** використовується у елемента **E-mail**. У цьому випадку на клавіатурі з'являється додаткова клавіша з символом @, який обов'язково використовується в будь-якому електронну адресу.

Атрибут **android:inputType = "phone"** використовується у елемента **Phone.** Клавіатура схожа на клавіатуру зі старого кнопкового стільникового телефону з цифрами, а також з кнопками зірочки і грати.

Атрибут **android:inputType = "textMultiLine"** використовується у **Multiline Text** дозволяє зробити текстове поле багаторядковим. Додатково можете встановити властивість Lines (атрибут android:lines), щоб вказати кількість видимих рядків на екрані.

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
android:inputType="textMultiLine"
android:lines="5" />
```

Атрибут **android:inputType = "time"** або **android: inputType = "date"** використовується для введення часу або дати. Атрибут **android:inputType="number|numberSigned|numberDecimal"** призначений для введення цифр та деяких інших символів (0123456789.-).

В Android у багатьох елементів є властивість **Hint** (атрибут **hint**), який працює як текст підказка. Встановіть у даної властивості потрібний текст і з'явиться текстове поле з підказкою.

Додаємо до xml файлу розмітки елемент EditText

```
<EditText
        android:id="@+id/name_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:hint="Name"
        android:inputType="textMultiLine" />
```

Основний метод класу EditText - **getText ()** повертає текст з текстового поля. Значення, що повертається має спеціальний тип **Editable**, а не String .

```
String strCatName = nickNameEditText.getText().toString();
// приводим до типу String
```

Відповідно, для установки тексту використовується метод **setText().** Більшість методів для роботи з текстом успадковані від базового класу TextView: setTypeface (null, Typeface) , setTextSize (int textSize) , SetTextColor (int Color) .

Вносимо зміни до методу **createOrderSummary**. Додаємо рядки

```
EditText nameEditText = findViewById(R.id.name_edit_text);
String priceMessage = "Name: " +  nameEditText.getText() +"\n";
```

```
private String createOrderSummary(int price, boolean addWhippedCream, boolean addChocolate) {
    EditText nameEditText = findViewById(R.id.name_edit_text);
    String priceMessage = "Name: " +  nameEditText.getText() +"\n";
    priceMessage += "Quantity: " + numberOfCofees  + " \n";
    if (addWhippedCream) {
        priceMessage += "Topping :  Whipped cream + 2$ \n";
        price+=2;
    }
    if (addChocolate) {
        priceMessage += "Topping :  Chocolate + 2$ \n";
        price+=2;
    }
    priceMessage += "Total: " + price + "$ \n";
    priceMessage += "Thank you!";
    return priceMessage;
}
```

Змінюємо роботу додатку JustJava. При натисненні кнопки ORDER формується електронний лист зі списком замовлення та ціною (ORDER SUMMARY). На екрані не відображається ORDER SUMMARY (рис. 3).

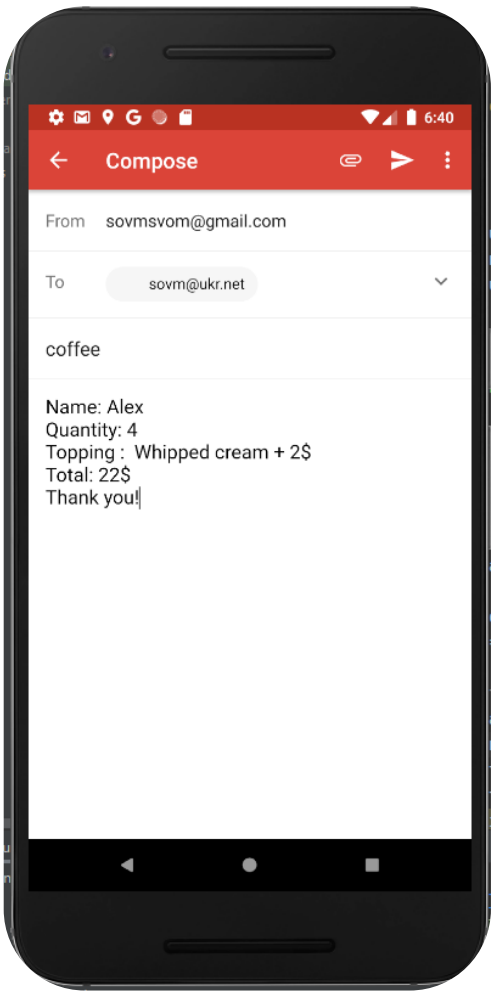


Рисунок 3 –Додаток JustJava. Формування електронного листа замовнику

Для того щоб це реалізувати потрібно розібратись з Intent. Intent це команда другого додатку або його компоненту. Приклади

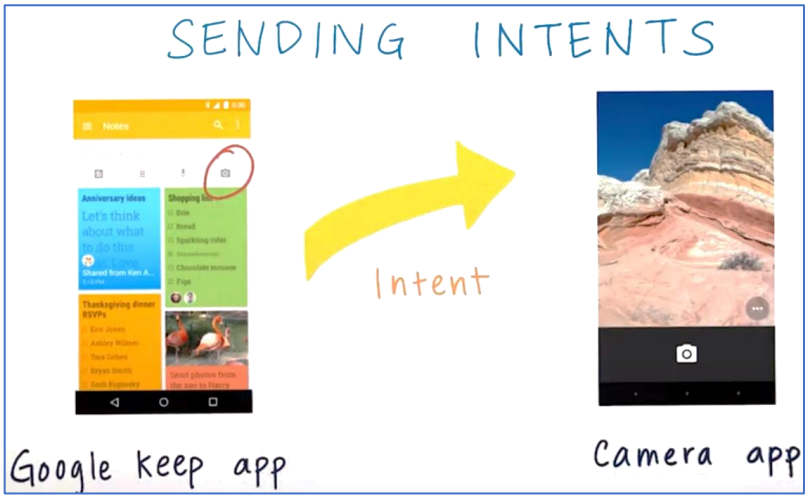


Рисунок 4 – Команда камера з Google keep відправляє Intent другому додатку – Camera

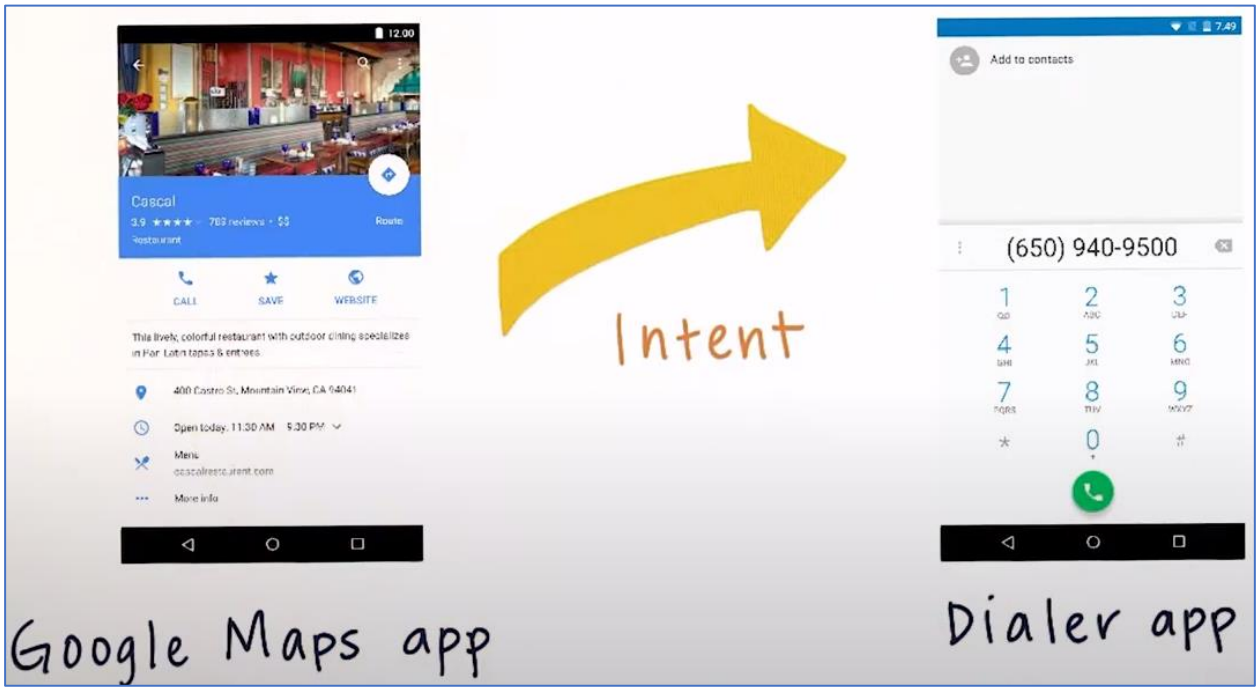


Рисунок 5 – Кнопка Call відправляє Intent додатку Dialer.

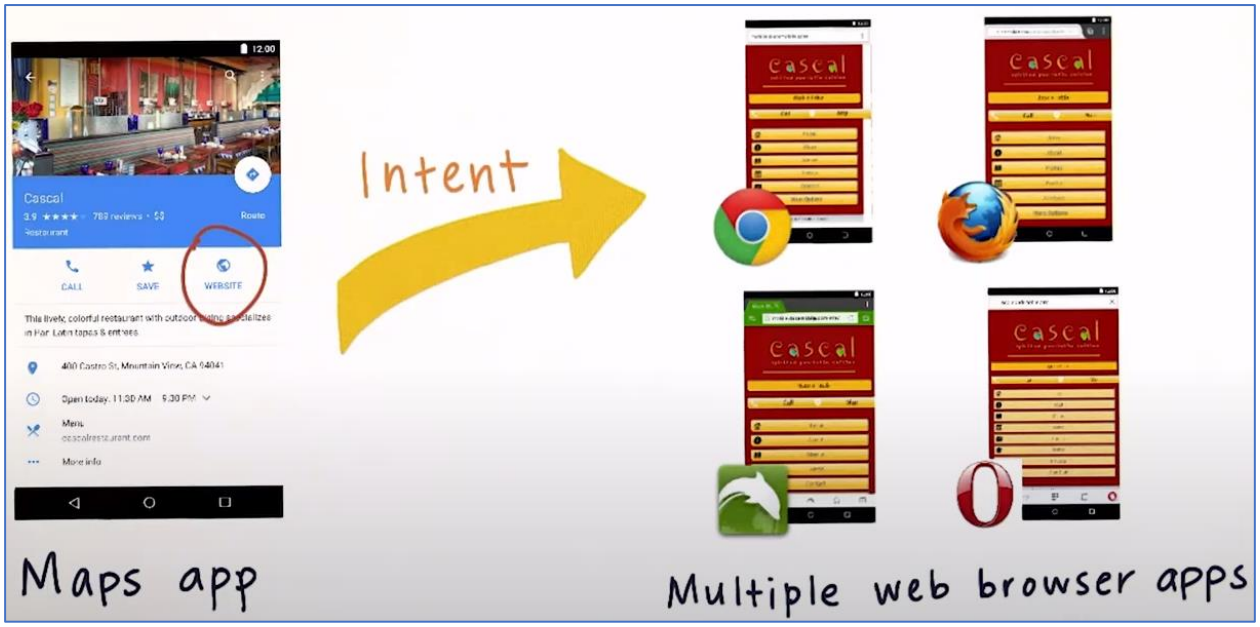
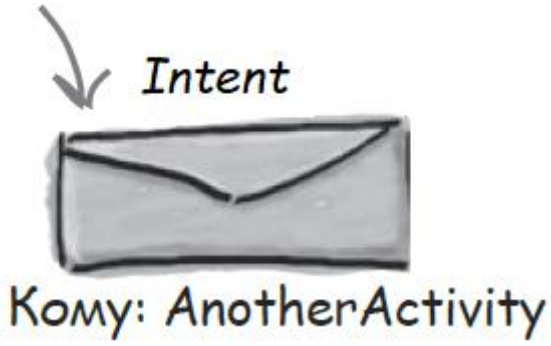


Рисунок 6 – При натисненні кнопки WEBSITE система пропонує вибрати 1 з 4 браузерів і після цього загрузиться веб сторінка ресторану.

Щоб запустити одну активність з іншої використовують **Intent**. Intent можна розглядати як свого роду «намір виконати якусь операцію». Це різновид повідомлень, що дозволяє зв'язати різнорідні об'єкти (наприклад, активності) на стадії виконання. Якщо одна активність хоче запустити іншу, вона відправляє для цього **Intent** системі Android. Android запускає другу активність та передає їй Intent.

Процедура створення та відправки Intent складається всього з двох рядків коду. Для початку створіть Intent:

Intent intent = new Intent (this, Target.class);



Перший параметр повідомляє Android, від якого об'єкта надійшов Intent; для позначення поточної активності використовується ключове слово this. У другому параметрі передається ім'я класу активності, яка повинна отримати Intent.

Після того як Intent буде створений, він передається Android наступним викликом: **startActivity(intent);**

```
startActivity(intent);
```
startActivity() запускає активність , що задана в intent

Цей виклик наказує Android запустити активність, яка визначається Intent. При отриманні Intent Android переконується в тому, що все правильно, і наказує активності запуститися. Якщо знайти активність не вдалося, викликається виключення ActivityNotFoundException. Новий Intent створюється командою

Intent intent = new Intent (this, Target.class);

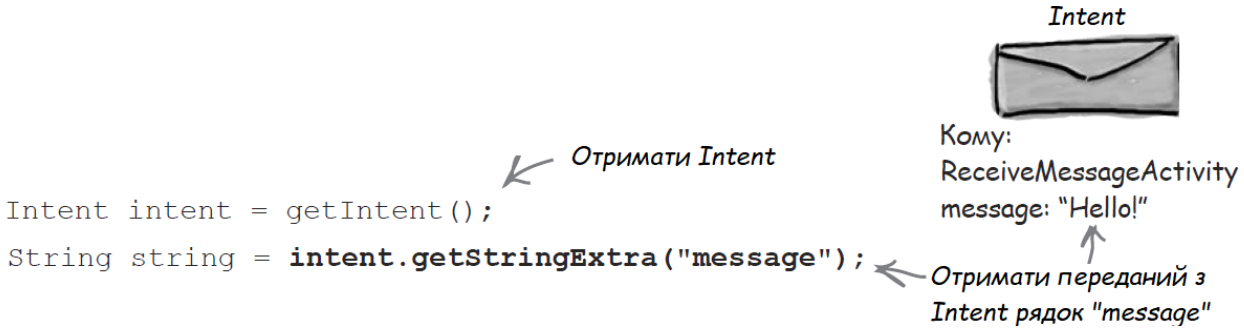
У Intent також можна додати додаткову інформацію, яка повинна передаватися одержувачу. В цьому випадку активність, яка отримала Intent, зможе на нього якось зреагувати. Для цього використовується метод **putExtra ():**

intent.putExtra ("повідомлення", призначення);

де повідомлення – ім'я ресурсу для переданої інформації, а значення – саме значення. Перевантаження методу **putExtra ()** дозволяє передавати значення багатьох можливих типів. Наприклад, це може бути примітиви (Boolean або int), масив примітивів або String. Багаторазові виклики putExtra () дозволяють включити в Intent кілька примірників додаткових даних.

Коли Android наказує Activity запуститися, активність повинна якимось чином отримати додаткову інформацію, яка була відправлена системі Android в Intent. Для вирішення цього використовують методи. Перший метод: **getIntent(). GetIntent()** повертає Intent, що запустив активність; з отриманого Intent можна прочитати будь-яку інформацію, відправлену разом з ним. Конкретний спосіб читання залежить від типу відправленої інформації. Наприклад, якщо відомо, що Intent має рядкове значення з ім'ям «message», то використовуйте такий виклик:

Intent intent = getIntent();
String string = intent.getStringExtra("message");



З Intent можна читати не тільки рядкові значення. Наприклад,

int intNum = intent.getIntExtra("name", default_value);

може використовуватися для отримання значення name. Параметр default_value вказує, яке значення int має використовуватися за замовчуванням.

При отриманні Intent система Android повинна визначити, яка активність (або активності) може цей Intent обробити. Цей процес називається **дозволом Intent**. При використанні явного Intent процес розв'язання такий: в самому Intent явно зазначено, для якого компонента він призначений, тому у Android є чіткі інструкції, що з ним робити. Наприклад, наступний код явно наказує Android запустити **ReceiveMessageActivity:**

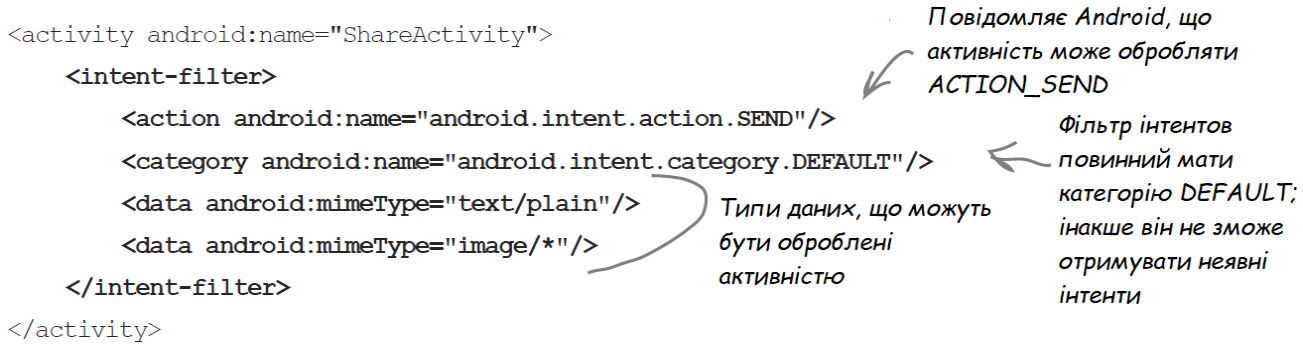
```
Intent intent = new Intent (this, ReceiveMessageActivity.class);
startActivity(intent);
```

При використанні неявного Intent система Android використовує інформацію, що міститься в Intent, для визначення того, які компоненти можуть його отримати. Для цього Android перевіряє фільтри Intent, що містяться в примірниках **AndroidManifest.xml.**

Фільтр Intent вказує, які типи Intent можуть оброблятися кожним компонентом. Наприклад, такий запис відноситься до активності, здатної обробляти ACTION_SEND. Ця активність приймає дані з MIME-типами text / plain або image:

```
<activity android:name="ShareActivity">
     <intent-filter>
       <action android:name="android.intent.action.SEND"/>
       <category android:name="android.intent.category.DEFAULT"/>
       <data android:mimeType="text/plain"/>
       <data android:mimeType="image/*"/>
     </intent-filter>
</activity>
```

Фільтр Intent також включає категорію. Категорія надає додаткову інформацію про активність: наприклад, чи може вона запускатися браузером або чи є вона головною точкою входу додатку. Фільтр Intent повинен включати категорію **android.intent.category.DEFAULT**, якщо він збирається приймати неявні Intent. Якщо активність не має фільтру Intent або не включає категорію з ім'ям **android.intent.category.DEFAULT**, це означає, що активність не може запускатися неявним Intent. Вона може бути запущена тільки явним Intent із зазначенням повного імені компонента (з включенням пакета).




```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
        <data android:mimeType="image/*"/>
    </intent-filter>
</activity>
```

Отримавши неявний Intent, Android порівнює інформацію з Intent з інформацією, що міститься в фільтрах Intent з файлу AndroidManifest.xml кожної програми.

Спочатку Android розглядає фільтри Intent, що включають категорію **android.intent.category.DEFAULT**:

```
<intent-filter>
     <category android:name="android.intent.category.DEFAULT"/>
...
</intent-filter>
```

Фільтри Intent без цієї категорії пропускаються, так як вони не можуть отримувати неявні Intent. Потім Android зіставляє Intent з фільтрами Intent, порівнює дії та тип MIME з Intent з зазначеними в фільтрах. Припустимо, якщо в Intent зазначено дію **Intent.ACTION_SEND**

```
Intent intent = new Intent(Intent.ACTION_SEND);
```

Android буде розглядати тільки ті активності, для яких указаний фільтр Intent з дією android.intent.action.SEND:

```
<intent-filter>
     <action android:name="android.intent.action.SEND"/>
...
</intent-filter>
```

Аналогічним чином, якщо для Intent установлений тип MIME text/plain:

```
intent.setType ("text/plain");
```

Android буде розглядати тільки ті активності, які підтримують цей тип даних:

```
<intent-filter>
     <data android:mimeType="text/plain"/>
...
</intent-filter>
```

Якщо тип MIME в Intent не вказаний, то Android намагається вирахувати його на підставі даних, що містяться в Intent. Після того як порівняння Intent з фільтрами Intent, призначених компонентам, буде завершено, Android дивиться, скільки збігів вдалося знайти. Якщо знайдено лише один збіг, Android запускає компонент (в нашому випадку це активність) та передає йому Intent. Якщо буде знайдено декілька збігів, Android просить користувача вибрати один з варіантів.

Константи дії

ACTION_ANSWER - Відкриває активність, пов'язану з вхідними дзвінками. Ця дія обробляється стандартним екраном для приймання дзвінків;
ACTION_CALL – робить звернення через телефон;
ACTION_DELETE - запускає активність, за допомогою якої можна видалити дані, зазначені у шляху URI всередині наміру;
ACTION_EDIT - показує дані для редагування користувачем;
ACTION_INSERT - відкриває активність для вставки в курсор нового елемента, вказаного за допомогою шляху URI. Дочірня активність, викликана з цією дією, повинна повернути URI, що посилається на вставлений елемент;
ACTION_HEADSET_PLUG - підключення навушників;
ACTION_MAIN - запускається як початкова активність завдання;
ACTION_PICK - завантажує дочірню Активність, що дозволяє вибрати елемент із джерела даних, вказаний за допомогою шляху URI. При закритті повинен

повертатися URL, що посилається на вибраний елемент. Активність, яка буде запущена, залежить від типу вибраних даних, наприклад під час передачі шляху **content://contacts/people** викликається системний список контактів;
ACTION_SEARCH - запуск активності для пошуку. Пошуковий запит зберігається у вигляді рядка у додатковому параметрі наміру за ключем SearchManager.QUERY;
ACTION_SEND -завантажує екран для надсилання даних, вказаних у намірі. Контакт-одержувач має бути вибраний за допомогою отриманої активності. Використовуйте метод setType, щоб вказати тип MIME для даних, що передаються. Ці дані повинні зберігатися у параметрі наміру extras із ключами EXTRA_TEXT або EXTRA_STREAM, залежно від типу. У випадку з електронною поштою стандартний додаток в Android також приймає додаткові параметри за ключами EXTRA_EMAIL, EXTRA_CC, EXTRA_BCC та EXTRA_SUBJECT. Використовуйте дію ACTION_SEND лише у випадках, коли дані потрібно передати віддаленому адресату (а не іншій програмі на тому самому пристрої);
ACTION_SENDTO - відкриває активність для надсилання повідомлень контакту, вказаному на шляху URI, який передається через намір;
ACTION_SYNC - синхронізує дані сервера з мобільним пристроєм;
ACTION_TIMEZONE_CHANGED - зміна часового поясу;
ACTION_VIEW - найпоширеніша спільна дія. Для даних, що передаються за допомогою шляху URI у намірі, виконується пошук найбільш відповідного способу виведення. Вибір програми залежить від схеми даних. Стандартні адреси http: будуть відкриватися в браузері, адреси tel: - у додатку для до дзвону, geo: - у програмі Google Maps, а дані про контакт - покажуться у додатку для керування контактною інформацією;
ACTION_WEB_SEARCH - відкриває активність, яка веде пошук в інтернеті, ґрунтуючись на тексті, переданому за допомогою шляху URI (зазвичай запускається браузер).

Константи категорій

CATEGORY_BROWSABLE - активність може бути безпечно викликана браузером, щоб показати посилання, наприклад, зображення або поштове повідомлення;
CATEGORY_HOME - активність показує Home Screen, перший екран, який користувач бачить після увімкнення пристрою та завантаження системи, або коли натискає клавішу HOME;
CATEGORY_LAUNCHER - активність може бути початковою діяльністю завдання зі списку програм у групі Application Launcher пристрою.

Методи

Для роботи з категоріями у класі Intent визначено групу методів:

addCategory() - поміщає категорію в об'єкт Intent;
removeCategory() - видаляє категорію, яка була додана раніше;
getCategories() - отримує набір всіх категорій, що знаходяться зараз в об'єкті Intent.

1) Переписуємо метод submitOrder(View view)[коментуємо метод displayMessage та додаємо рядки Intent]

```
public void submitOrder(View view) {
   int price = calculatePrice();
   //Log.v("MainActivity", "The price is " + price);

   CheckBox whippedCreamCheckBox = (CheckBox) findViewById(R.id.check_box);
    boolean hasWhippedCream = whippedCreamCheckBox.isChecked();

    CheckBox chocolateCheckBox = (CheckBox) findViewById(R.id.chocolate_check_box);
    boolean hasChocolate = chocolateCheckBox.isChecked();

     //displayMessage(createOrderSummary(price, hasWhippedCream, hasChocolate));
     String message = (createOrderSummary(price, hasWhippedCream, hasChocolate));
     Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.parse("mailto:"+
     "sovm@ukr.net"));
      intent.putExtra(Intent.EXTRA_SUBJECT,"coffee");
      intent.putExtra(Intent.EXTRA_TEXT, message);
      if (intent.resolveActivity(getPackageManager()) != null) {
          startActivity(intent);
      } }
```

2) коментуємо TextView order_summary_text_view, price_text_view

```
                <!--
                ****
                -->
<!--
  <TextView
      android:id="@+id/order_summary_text_view"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginBottom="16dp"
      android:text="ORDER SUMMARY" />

   <TextView
      android:id="@+id/price_text_view"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:text="0"
        android:textColor="#000000"
        android:textSize="16sp" />
-->
    <Button
        android:id="@+id/order_botton_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:textSize="16sp"
        android:onClick="submitOrder"
        android:text="ORDER" />
```

MainActivity.java (додаток JustJava)

```
package android.example.com;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {

    int numberOfCofees = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    /**
     * This method is called when the order button is clicked.
     */
    public void submitOrder(View view) {
        int price = calculatePrice();
        //Log.v("MainActivity", "The price is " + price);
```

```
    CheckBox whippedCreamCheckBox = (CheckBox) findViewById(R.id.check_box);
    boolean hasWhippedCream = whippedCreamCheckBox.isChecked();

    CheckBox chocolateCheckBox = (CheckBox) findViewById(R.id.chocolate_check_box);
    boolean hasChocolate = chocolateCheckBox.isChecked();

    //displayMessage(createOrderSummary(price, hasWhippedCream, hasChocolate));
    String message = (createOrderSummary(price, hasWhippedCream, hasChocolate));
    Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.parse("mailto:" + "sovm@ukr.net"));
    intent.putExtra(Intent.EXTRA_SUBJECT,"coffee");
    intent.putExtra(Intent.EXTRA_TEXT, message);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }

}

private int calculatePrice() {
    int basePrice = 5;
    return numberOfCofees * basePrice;
}

/**
 * This method displays the order on the screen.
 */
private String createOrderSummary(int price, boolean addWhippedCream, boolean addChocolate) {
    EditText nameEditText = findViewById(R.id.name_edit_text);
    String priceMessage = "Name: " +  nameEditText.getText() +"\n";
    priceMessage += "Quantity: " + numberOfCofees  + " \n";
    if (addWhippedCream) {
        priceMessage += "Topping :  Whipped cream + 2$ \n";
        price+=2;
    }
    if (addChocolate) {
        priceMessage += "Topping :  Chocolate + 2$ \n";
        price+=2;
    }
    priceMessage += "Total: " + price + "$ \n";
    priceMessage += "Thank you!";
    return priceMessage;
}

/**
 * This method displays the given quantity value on the screen.
 */
private void displayQuantity(int number) {
    TextView quantityTextView = (TextView) findViewById(R.id.quantity_text_view);
    quantityTextView.setText("" + number);
}

public void countPlus(View view) {
```

```
        numberOfCofees++;
        display(numberOfCofees);
    }

    public void countMinus(View view) {

        if(numberOfCofees>1) numberOfCofees--;
        display(numberOfCofees);
    }

    /**
     * This method displays the given quantity value on the screen.
     */
    private void display(int number) {
        TextView quantityTextView = (TextView) findViewById(R.id.quantity_text_view);
        quantityTextView.setText("" + number);
    }

    /**
     * This method displays the given price on the screen.
     */
   // private void displayPrice(int number) {
    //    TextView priceTextView = (TextView) findViewById(R.id.price_text_view);
    //   priceTextView.setText(java.text.NumberFormat.getCurrencyInstance(new Locale("ua",
"UA")).format(number));
    //}
    /**
     * This method displays the given text on the screen.
     */
    /*private void displayMessage(String message) {
        TextView orderSummaryTextView = (TextView) findViewById(R.id.price_text_view);
        orderSummaryTextView.setText(message);
    }*/
}
```

http://developer.alexanderklimov.ru/android/theory/intent.php
https://betacode.net/10425/android-intent
https://betacode.net/12587/example-of-an-explicit-android-intent-calling-another-intent
https://betacode.net/12583/example-of-implicit-android-intent-open-a-url-send-an-email