

Л6. Запис повідомлень в журнал. CheckBox

Запис повідомлень в журнал часто допомагає переконатися в тому, що код працює саме так, як потрібно. Ви повідомляєте Android, яку інформацію потрібно зберегти, в своєму коді Java, а потім під час роботи програми переглядаєте результати в журналі Android.

Для збереження повідомлень в журналі використовуються такі методи класу `Android.util.Log`:

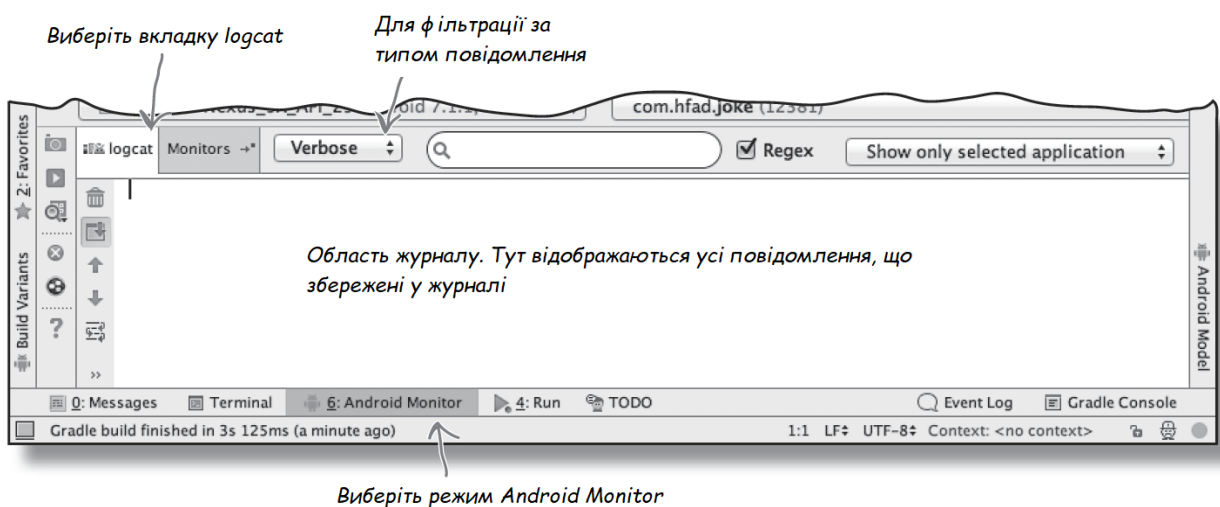
- `Log.v(String tag, String message)` – verbose (подробиці)
- `Log.d(String tag, String message)` – debug (налагодження)
- `Log.i(String tag, String message)` – information (інформація)
- `Log.w(String tag, String message)` – warning (попередження)
- `Log.e(String tag, String message)` – error (помилка).

До кожного повідомлення входить рядкова мітка, яка може використовуватись для ідентифікації джерела повідомлення та самого повідомлення. Наприклад, для збереження детального повідомлення, що надійшло від служби `DelayedMessageService`, використовуємо виклик `Log.v()` такого виду:

```
Log.v("DelayedMessageService", "This is a message");
```

У Java-коді пишемо, наприклад, такий `Log.i` (перший аргумент – це назва класу, з якого зроблено запис у журнал, другий аргумент – це текст повідомлення, що буде відображено)

```
Log.i("EnterpriseActivity.java", "Captain's Log, Stardate 43125.8. We have entered a spectacular binary star system in the Kavis Alpha sector on a most critical mission of astrophysical research.");
```



Android Studio надає засоби для перегляду журналу та фільтрації даних за типами повідомлень. Щоб переглянути зміст журналу, виберіть режим Android

Monitor в нижній частині вікна проекту в Android Studio, а потім перейдіть на вкладку logcat. Обраний рівень буде показувати всі log свого й вищих рівнів, так що рівень подробиць (verbose) покаже максимум інформації, а рівень помилки (error) покаже тільки найважливіші записи.

```
package android.example.com;

import android.os.Bundle;
import android.util.Log;//імпорт класу Log
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import java.util.Locale;

//This app displays an order form to order coffee.
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

Прапорці (check boxes) надають користувачеві набір незалежних варіантів вибору. Користувач може вибрати будь-які варіанти на свій розсуд. Кожен прапорець може встановлюватися або зніматися незалежно від всіх інших прапорців.

Прапорець задається в XML елементом <CheckBox>. Атрибут android:text використовується для визначення тексту, що виводиться рядом з CheckBox:

```
<CheckBox android:id="@+id/checkbox_milk"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/milk" />
<CheckBox android:id="@+id/checkbox_sugar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/sugar" />
```

Щоб перевірити, чи встановлений `CheckBox`, використовується метод `isChecked()`. Якщо метод повертає `true`, значить, прапорець встановлений:

```
CheckBox checkbox=(CheckBox) findViewById(R.id.checkbox_milk);
boolean checked = checkbox.isChecked();
if (checked) {
    //Дія
}
```

Перемикачі (radio buttons) надають набір варіантів, з яких користувач може вибрати один варіант:



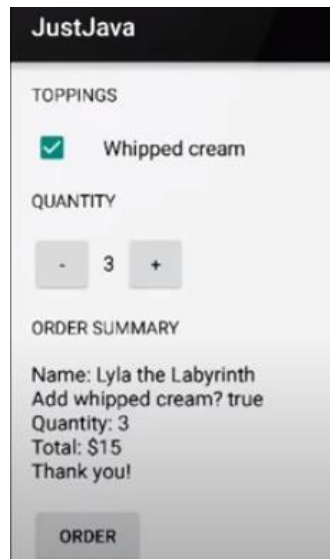
Перемикачі представляються елементом `<RadioGroup>` всередині якого окремі перемикачі визначаються елементами `<RadioButton>`:

```
<RadioGroup android:id="@+id/radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/radio_cavemen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cavemen" />
    <RadioButton android:id="@+id/radio_astronauts"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/astronauts" />
</RadioGroup>
```

Щоб визначити, який перемикач у групі встановлений, використовується метод `getCheckedRadioButtonId()`:


```
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.radioGroup);
int id = radioGroup.getCheckedRadioButtonId();
if (id == -1){
    //Перемикачі не встановлені
}
else{
    switch(id) {
        case R.id.radio_cavemen:
            // Установлений перемикач Cavemen
            break;
        case R.id.radio_astronauts:
            // Установлений перемикач Astronauts
            break;
    }
}}
```

Додаємо елемент CheckBox



Змінюємо функціонал додатку JustJava (Л5, Л6). При натисненні кнопки Order необхідно отримати стан елементу View типу CheckBox, зберегти цей стан у логічну змінну. Можна зробити запис до журналу, щоб переконатись, що крок 1 виконаний вірно. Передаємо логічну змінну в метод createOrderSummary. Метод буде приймати 2 параметри. Новим параметром буде логічна змінна з назвою hasWhippedCream (зі збитими вершками). Змінюємо метод createOrderSummary, щоб використовувалась логічна змінна при виводі тексту на екран.

Common Android Views



CheckBox

Checkbox with text label

```
<CheckBox
    android:id="@+id/notify_me_checkbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/notify_me"
    android:textAppearance="?android:textAppearanceMedium" />
```

 Notify me

1. Створюємо об'єкт типу checkBox та методом findViewById прив'язуємо до checkBox в розмітці.
2. Створюємо булеву змінну і присвоюємо їй стан checkBox, звертаючись до нього і викликаючи відповідний метод:

```
hasWhippedCream = checkBox.isChecked();
```

3. У підсумковому методі замовлення додаємо другий параметр булевого типу і передаємо стан checkBox.
4. Конкатенація переданого аргументу з іншими рядками для виведення на екран.

Частина коду, що реалізовує зазначені дії

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
```

```
<TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAllCaps="true"
    android:text="toppings"/>
```

```
<CheckBox
```

```
    android:id="@+id/check_box"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Whipped_cream"
    android:textSize="16sp"
    android:paddingLeft="24dp"/>
```

```
<TextView
```

```
    android:id="@+id/Quan"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:text="Quantity"
    android:textAllCaps="true"/>
```

```
<LinearLayout
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

```
<Button
```

```
    android:id="@+id/minus_botton_view"
    android:layout_width="48dp"
    android:layout_height="48dp"
    android:layout_marginBottom="16dp"
    android:onClick="countMinus"
    android:text="-"
    android:textSize="16sp" />
```

фрагмент MainActivity.java

```
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();
    displayMessage(createOrderSummary(price, hasWhippedCream));
}

private String createOrderSummary(int price, boolean addWhippedCream) {
    String priceMessage = "Name: Mallyness\n";
    priceMessage += "Add whipped cream? " + addWhippedCream + " \n";
    priceMessage += "Quantity: " + numberOfCofees + " \n";
    priceMessage += "Total: " + price + "$ \n";
    priceMessage += "Thank you!";
    return priceMessage;
}
```

Додаємо вертикальну прокрутку

При великій кількості інформації, яку потрібно помістити на екрані доводиться використовувати смуги прокрутки. В Android існують спеціальні компоненти **ScrollView** і **HorizontalScrollView**, які є контейнерними елементами і успадковуються від ViewGroup. Клас TextView використовує свою власну прокрутку і не потребує додавання окремих смуг прокрутки. Але використання окремих смуг навіть з TextView може поліпшити вигляд вашого додатку та підвищує зручність роботи для користувача.

На панелі інструментів смуги прокрутки можна знайти в розділі Containers.

У контейнери ScrollView і HorizontalScrollView можна розміщувати тільки один дочірній елемент (зазвичай LinearLayout), який в свою чергу може бути контейнером для інших елементів. Віджет ScrollView, незважаючи на свою назву, підтримує тільки вертикальну прокрутку, тому для створення вертикальної і горизонтальної прокрутки необхідно використовувати ScrollView в поєднанні з HorizontalScrollView. Зазвичай ScrollView використовують як кореневий елемент, а HorizontalScrollView як дочірній. Можна і навпаки. Приховати смуги прокрутки можна через атрибут android:scrollbars = "none".

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    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:scrollbars="none">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
```

.....

```
</ScrollView>
```

Зараз для прокрутки актуально використовувати **NestedScrollView**.

Додаємо **CheckBox Chocolate**



Усе по аналогії з попереднім Checkbox

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<ScrollView
```

```
    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:scrollbars="none">
```

```
<LinearLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">
```

```
<TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAllCaps="true"
    android:text="toppings"/>
```

```
<CheckBox
    android:id="@+id/check_box"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Whipped_cream"
    android:textSize="16sp"
    android:paddingLeft="24dp"/>
```

```
<CheckBox
    android:id="@+id/chocolate_check_box"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Chocolate"
    android:textSize="16sp"
    android:paddingLeft="24dp"/>
```

```
<TextView
    android:id="@+id/Quan"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:text="Quantity"
    android:textAllCaps="true"/>
```

```
</ScrollView>
```

```
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));
}

private String createOrderSummary(int price, boolean addWhippedCream, boolean addChocolate) {
    String priceMessage = "Name: Mallyness\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;
}

```

Повідомлення (toast) – це просте спливаюче повідомлення, яке з'являється на екрані. Повідомлення виконують інформаційні функції, користувач не може з ними взаємодіяти. Доки повідомлення знаходиться на екрані, активність залишається видимою та доступною для взаємодії з користувачем. Повідомлення автоматично закривається після закінчення тайм-ауту. Повідомлення створюються тільки в коді активності; визначити їх в макеті неможливо.

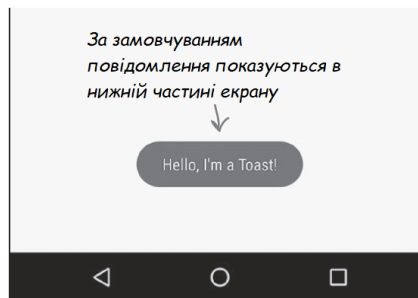
Щоб створити повідомлення, використовується метод `Toast.makeText()` і передають йому три параметра: `Context` (зазвичай для поточної активності), `CharSequence` (текст повідомлення) та `int` (тривалість). Після того як об'єкт повідомлення буде створений, його можна вивести на екран викликом методу `show()`. Приклад коду зі створенням повідомлення, ненадовго з'являється на екрані:

```

CharSequence text = "Hello, I'm a Toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(this, text, duration);
toast.show();

```



- Всі компоненти графічного інтерфейсу є спеціалізаціями узагальненого уявлення. Всі вони представлені субкласом класу **android.view.View**.
- Всі макети є субкласами класу **android.view.ViewGroup**. **ViewGroup** може містити кілька **View**.
- Файл з розміткою XML макета перетворюється в об'єкт **ViewGroup**, що містить ієрархічне дерево **View**.
- У лінійному макеті **View** розміщуються або по горизонталі, або по вертикалі. Напрямок задається атрибутом **android:orientation**.
- Атрибути **android:padding** визначають величину відступів навколо **View**.
- Використовуйте атрибут **android:layout_weight** в лінійному представленні, якщо хочете, щоб **View** займало додаткове місце в макеті.
- Елемент **<imageView>** призначений для виведення графіки.
- Атрибут **android:layout_gravity** вказує, в якій частині доступного простору має перебувати уявлення.
- Атрибут **android:gravity** вказує, в якій частині **View** повинно відобразитися його вміст.
- Елемент **<ToggleButton>** визначає двопозиційний кнопку. Клацаючи на кнопці, користувач вибирає одне з двох станів.
- Елемент **<CheckBox>** визначає прапорець.
- Щоб визначити групу перемикачів, спочатку визначте групу перемикачів елементом **<RadioGroup>**. Окремі перемикачі в групі визначаються елементом **<RadioButton>**.
- Елемент **<ImageButton>** визначає кнопку, яка не містить тексту - тільки зображення.
- Для додавання смуг прокрутки використовуються елементи **<ScrollView>**, **<HorizontalScrollView>**.
- Об'єкт **Toast** представляє тимчасове повідомлення.