Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
"Архітектура комп'ютерів"
для студентів спеціальностей
8.05010201 "Комп’ютерні системи та мережі" та
8.05010203 "Спеціалізовані комп'ютерні системи"
всіх форм навчання
Частина 2

2014

Методичні вказівки до виконання лабораторних робіт з дисципліни "Архітектура комп'ютерів" для студентів спеціальностей 8.05010201 "Комп’ютерні системи та мережі" та 8.05010203 "Спеціалізовані комп'ютерні системи" всіх форм навчання. Частина 2 / Укл. Скрупський С.Ю. – Запоріжжя: ЗНТУ, 2014. – 46 с.

Укладач: С.Ю. Скрупський, к.т.н., доцент

Рецензент: О.І. Вершина, к.т.н., доцент

Відповідальний за випуск: С.Ю. Скрупський, к.т.н., доцент

Затверджено:
на засіданні кафедри
"Комп’ютерні системи та
мережі"
Протокол № 1 від 22.08.2014

ЗМІСТ

1 ЛАБОРАТОРНА РОБОТА № 1. ПРОГРАМУВАННЯ ТИПОВОГО ДОДАТКУ ДЛЯ МОБІЛЬНОЇ ПЛАТФОРМИ ANDROID

Мета роботи: знайомство з платформою Android та середовищем Eclipse, здобуття базових навичок програмування додатку з урахуванням особливостей мобільної платформи.

1.1 Порядок виконання роботи

1.1.1 Запустіть середовище розробки Eclipse. Створіть новий проект для платформи Android: File|New|Android Application Project. Налаштування проекту виконайте згідно рис. 1.1,1.2.

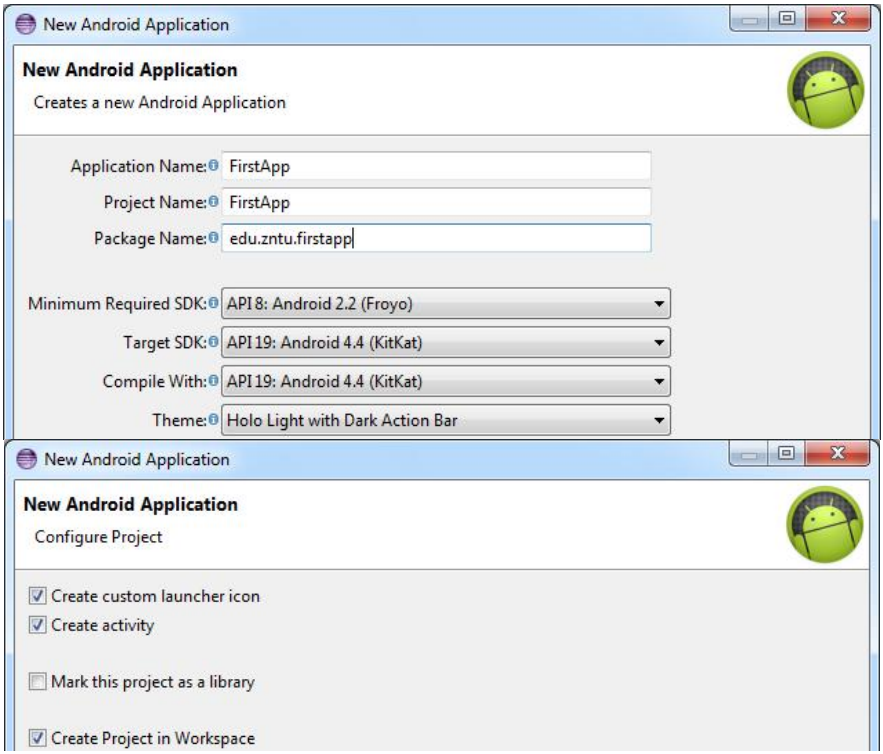


Рисунок 1.1 – Налаштування проекту

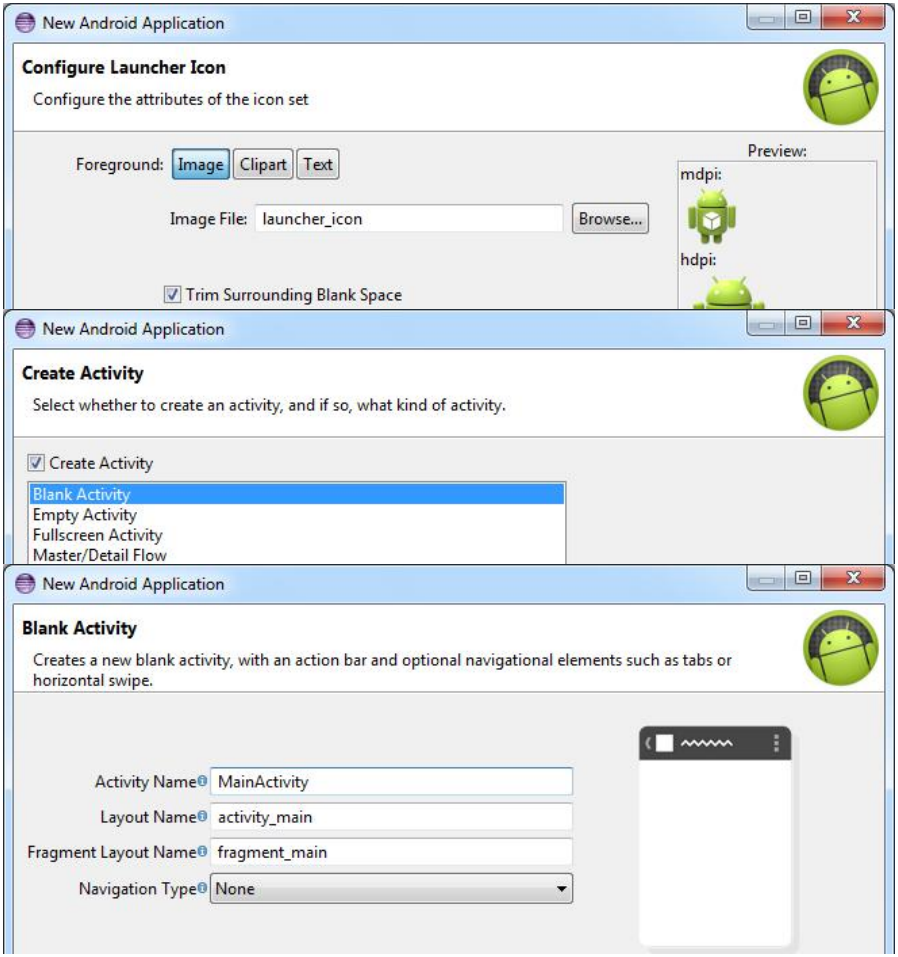
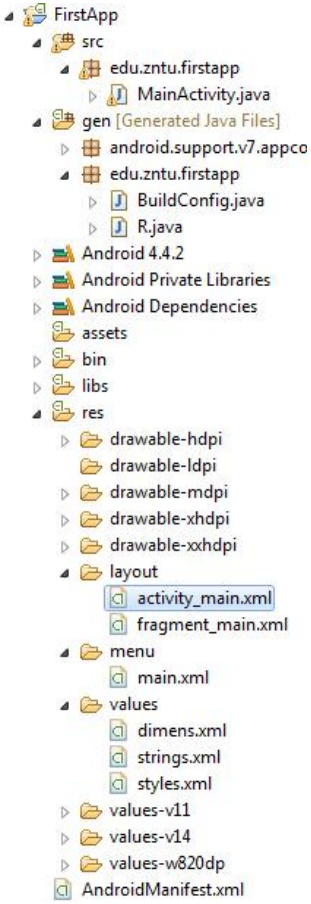


Рисунок 1.2 – Подальші кроки налаштування проекту

В результаті Eclipse створить новий проект типового додатку для мобільної платформи Android. Вміст проекту буде таким, як наведено на рис. 1.3 (з незначними змінами в залежності від версії середовища).



src –код додатку, буде в цій папці та її підпапках;
gen – генеруються середовищем розробки для програми. Тут краще нічого не чіпати. (Якщо цієї папки немає – змініть який-небудь файл в проекті і натисніть кнопку зберегти);
assets і res – папки для файлів-ресурсів різного типу;
AndroidManifest.xml – маніфест або конфіг-файл програми;
bin – генеруються середовищем розробки та містять виконавчі файли додатку;
libs – файли підключаємих до проекту бібліотек;
drawable – ці папки містять файли-зображення, що використовуються у проекті. Назва папки залежить від роздільної здатності екрана смартфону;
layout – папка містить групи об'єктів (видів, View) у вигляді xml. Елементи інтерфейсу користувача (кнопки, поля вводу та ін.) налаштовуються саме у цих файлах.
menu – містить налаштування пунктів основного меню додатку та контектного меню
values – містить файли констант (стилі, строки).

Рисунок 1.3 – Вміст типового проекту Android

1.1.2 Типовий додаток повинен містити хоча б один файл layout. В даному випадку – це activity_main.xml, що використовується для налаштування інтерфейсу додатку. Fragment_main.xml потрібний для налаштування фрагментів, зараз він нам не знадобиться, закрийте його. Будь-який xml можна редагувати у текстовому вигляді, або за допомогою графічного редактора. В даному випадку зручніше використати графічний редактор (вкладка Graphical layout). Вміст

activity_main.xml, відкритого у графічному редакторі, зображено на рис. 1.4.

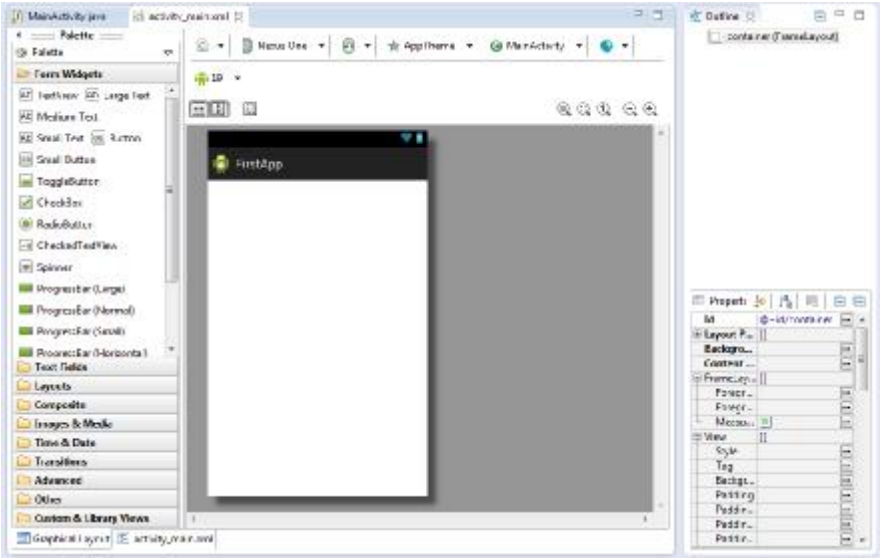


Рисунок 1.4 – Налаштування головного layout додатку

В лівій панелі Palette розміщені види (View) та віджети (Widget), тобто елементи інтерфейсу, що можна переносити на layout (екран) додатку. В центрі розміщений layout, що зараз редагується. Елементи інтерфейсу на телефоні після запуску додатку будуть відображені саме так, як на layout. В правому нижньому куті відображено панель властивостей об'єкта, що редагується (вона контекстна). В правому верхньому куті розміщено дерево контейнера: розмітки, види та віджети у формі ієрархічного дерева (для швидкого доступу).

Перш ніж виносити об'єкти на layout, треба вибрати тип розмітки. Для цього клацніть правою кнопкою миші по container|change layout… після цього зі списку вибираємо Relative layout (відносне позиціонування об'єкта щодо інших елементів, що є більш зручним для простих додатків). Крім відносного позиціонування існують такі:

LinearLayout – відображає View-елементи у вигляді одного рядка (якщо він Horizontal) або одного стовпця (якщо він Vertical);

TableLayout – відображає елементи у вигляді таблиці по рядках і стовпцях;

AbsoluteLayout – для кожного елемента вказується явна позиція на екрані в системі координат (x, y). Не рекомендується для використання, адже не можливо вирахувати точну позицію об'єктів для всіх розмірів екрана одночасно.

1.1.3 Після встановлення типу розмітки можна розміщувати об'єкти на layout. Для елементів, що містять простий текст, винесіть TextView з палітри на центр layout (рис. 1.5). Направляючі допоможуть спозиціонувати об'єкти.

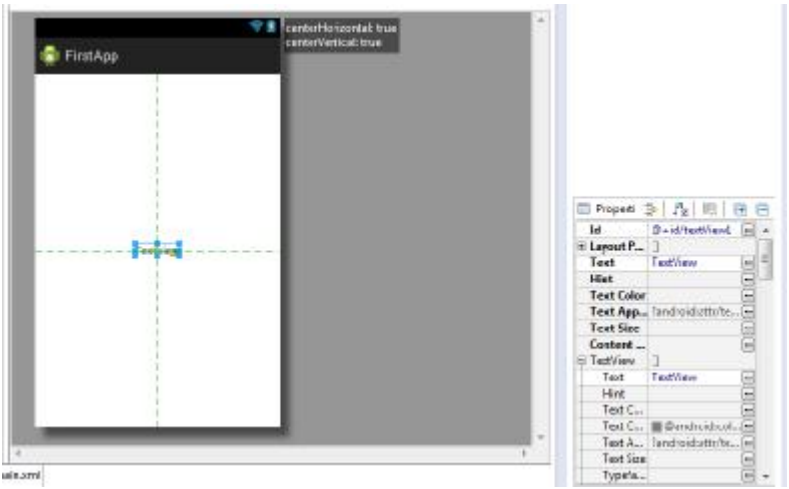


Рисунок 1.5 – Розміщення TextView у центрі layout додатку

Кожен елемент інтерфейсу містить id (унікальний ідентифікатор ресурса), що автоматично генерується у файлі R.java. Для зміни ідентифікатора відредагуйте поле Id вікна властивостей елемента TextView. Встановіть замість @+id/textView1 ідентифікатор @+id/tv1. Символ @ означає посилання на ресурси проекту, а + це команда додавання нового ресурсу. Підтвердіть зміну ідентифікатора.

Для зміни текстового вмісту елемента відредагуйте поле Text властивойстей. Впишіть туди "Hello World!!!" (без лапок).

1.1.4 Розмістіть на layout ще два елементи TextView. Один – вище центрального елемента, а другий – нижче. Змініть ідентифікатор

верхнього елемента на @+id/tvZNTU, нижнього – на @+id/tvName. Змініть вміст верхнього елемента на "ZNTU", нижнього – на Ваше ім'я. Змініть колір елементів на будь-який інший. Для цього у полі Text Color запишіть число, що відповідає #RRGGBB палітрі у 16-річному вигляді, наприклад зелений колір – #00FF00.

1.1.5 Налаштування layout простого додатку завершені, переходимо до коду. Відкрийте на редагування файл головного класу MainActivity.java. Eclipse згенерувала його з підтримкою меню та фрагментів, які в нашому додатку не потрібні, тому відредагуйте його вміст як показано нижче:

```
package edu.zntu.firstapp;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity
{
     @Override
     protected void onCreate(Bundle savedInstanceState)
     {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
     }
}
```

Середовище підкреслює слово Activity, тому що ми не імпортували необхідні пакети. Для цього наведіть стрілку миші на Activity та виберіть зі списку підказок "Import Activity". Для видалення зайвих імпортувань наведіть мишу на секцію import та виберіть "Remove unused import" та "Fix same problems".

Головний клас, названий `MainActivity`, розширює суперклас `Activity` з `Android SDK`, в якому містяться основні методи роботи з активністю (екраном).

Метод `onCreate` визивається на початку життєвого циклу активності, тобто його код виконується першим. Він містить визов відповідного методу суперкласу, який виконує фонові підготовчі дії, та метод `setContentView`, що встановлює layout для цього Activity. Зверніть увагу, як формуються посилання у файлі ресурсів R: `R.layout.activity_main`. Посилання на файл ресурсів, далі на

layout-ресурси і далі наконкретний layout. Аналогічно будуються інші посилання в Android додатках.

1.1.6 Додамо одному з TextView можливість змінювати текст і колір при натисканні по ньому (клік пальцем на елементі в сенсорних екранах). Для цього змініть вихідний код MainActivity.java наступним чином:

```
public class MainActivity extends Activity
{

   TextView tvZNTU;
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      tvZNTU = (TextView) findViewById(R.id.tvZNTU);
      tvZNTU.setOnClickListener(new OnClickListener()
      {
         @Override
         public void onClick(View v)
         {
            tvZNTU.setText("Kafedra KSN");
            tvZNTU.setTextColor(Color.rgb(150, 200, 50));
         }
      });
   }
}
```

Для імпортування класу, що працює з TextView наведіть курсор миші по підкресленому червоним кольором TextView та виберіть зі списку підказок import TextView. Також необхідно проімпортувати OnClickListener. Для цього наведіть курсор миші на new OnClickListener() та зі списку підказок виберіть import OnClickListener.

Ми створили екземпляр класу TextView, а саме tvZNTU. У методі onCreate прив'язуємо наш екземпляр до об'єкта на layout. Для цього використовуємо метод findViewById, в якому вказуємо посилання на id TextView у файлі ресурсів R.java.

Від екземпляра класа TextView викликаємо метод setOnClickListener, що прив'язує обробник події натискання. Аргументом цього метода має бути екземпляр класу OnClickListener. В Java існують анонімні (без імені) класи. В даному випадку зручно створити анонімний клас new OnClickListener. Ключове слово new створює екземпляр класа та повертає посилання на нього. Ця посилання і буде аргументом метода setOnClickListener. В тілі анонімного класу необхідно перевизначити метод onClick. Його тіло буде виконуватися тоді, коли користувач клікає по TextView, що має ідентифікатор tvZNTU. В тілі метода змінюємо текст TextView за допомогою метода setText, а також змінюємо колір тексту методом setTextColor.

1.1.7 Запустимо створений додаток для платформи Android. Для цього можна скористатися смартфоном, підключеним за допомогою USB, або емулятором такого пристрою. Ми використовуватимемо емулятор пристрою. Він емулює поведінку процесора ARM та виконує реальну прошивку, що зберігається у фалі-образі в Android SDK.

Створіть віртуальний пристрій, що знадобиться нам для виконання цієї та усіх наступних лабораторних робіт. В меню Eclipse виберіть Window|Android Virtual Device Manager. Виберіть пункт New. Виконайте налаштування згідно рис. 1.6.

Далі в опціях запуску додатку необхідно вибрати емулятор: меню Run|Run configurations… У полі Project виберіть FirstApp та опцію Launch Default Activity. На вкладці Target виберіть Automatically pick compatible device. У таблиці виберіть наш віртуальний пристрій "VirtualTelefon1". Натисніть кнопку Run. Запуск віртуального смартфону може відбуватися декілька хвилин, оскільки це пов'язане з емуляцією процесора ARM та виконанням прошивки. Після запуску віртуального смартфону розблокуйте його провівши мишею по блокувальнику зліва на право. На екрані повинен відобразитися наш додаток (рис. 1.7). Клацніть мишею по тексту ZNTU, він має змінити вміст та колір. Переверніть екран натиснувши Ctrl+F12. Як бачите, вміст TextView не зберігається при перевертанні екрана. Для його збереження необхідно програмувати передачу вмісту між екранами.

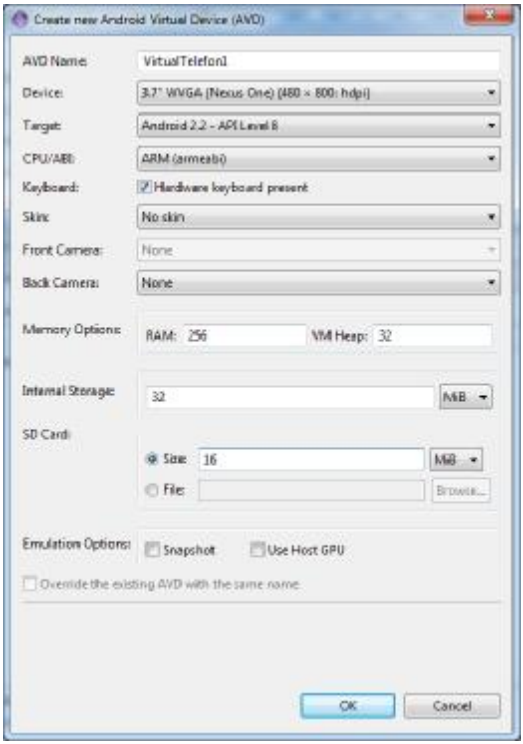


Рисунок 1.6 – Налаштування віртуального смартфона Android

Рисунок 1.7 – Перший додаток на емуляторі смартфона

1.1.8 Завдання для самостійного виконання:

Варіант 1: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.secondapp. Створіть два TextView: один повинен містити ваше ім'я, а другий – прізвище. По натисканні на будь-якому з них, колір тексту повинен змінюватися на випадковий.

Примітка: для генерації випадкового цілого числа використовуйте клас Random. Від екземпляра цього класу виконайте метод nextInt(256), що згенерує випакове число від нуля до 255.

Варіант 2: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.secondapp. Створіть TextView, в якому повинна відображатися поточна дата, а при натисканні на ньому – поточний час.

Примітка: для отримання строки з поточним часом можна використати клас SimpleDateFormat, наприклад:

```
SimpleDateFormat sdf;
sdf = new SimpleDateFormat("HH:mm:ss");
String time = sdf.format(
              new Date(System.currentTimeMillis()));
```

Поле time можна використати у якості аргумента метода setText TextView.

Для отримання поточної дати можна використати наступний код:

```
sdf = new SimpleDateFormat("dd.MM.yyyy");
```

1.2 Зміст звіту

1.2.1 Структура проекту додатку для платформи Android.

1.2.2 Зображення результатів виконання програми згідно індивідуального варіанту.

1.2.3 Текст програми згідно індивідуального варіанту.

1.2.4 Відповіді на контрольні питання.

1.3 Контрольні питання

1.4.1 Що собою являє мінімальний додаток для Android?

1.4.2 Які особливості мобільної платформи (які обмеження треба враховувати при написанні додатків для смартфонів)?

1.4.3 Поясніть призначення Activity, Layout, TextView.

1.4.4 Поясніть життєвий цикл Activity.

1.4.5 Як додати обробник кліків до будь-якого об'єкта layout?

2 ЛАБОРАТОРНА РОБОТА № 2. ІНТЕРФЕЙС ВЗАЄМОДІЇ МОБІЛЬНОГО ПРИСТРОЮ З КОРИСТУВАЧЕМ

Мета роботи: здобуття навичок програмування основних елементів інтерфейсу взаємодії мобільного пристрою з користувачем.

2.1 Порядок виконання роботи

Розробимо додаток, який по натисканню кнопки "+" буде рахувати суму двох чисел, взятих з двох полей для вводу. На основі такого додатку можна організувати будь-який типовий інтерфейс взаємодії з користувачем.

2.1.1 Створіть новий проект для платформи Android, назвіть його "CalcApp", назва пакету – "edu.zntu.calcapp".

2.1.2 Відкрийте activity_main для редагування у графічному режимі. Змініть тип layout на RelativeLayout. З палітри винесіть на layout два поля вводу (елементи Plain Text з групи Text Fields). Змініть id текстових полів на @+id/edit1 та @+id/edit2.

2.1.3 Між текстовими полями розмістіть кнопку (елемент Button з групи Form Widgets). Задайте їй ідентифікатор @+id/btnSum. Властивості "Text", що відповідає за напис на кнопці задайте значення "+".

2.1.4 Додайте TextView на layout, в якому будемо виводити результат обчислень. Змініть його ідентифікатор на @+id/tvRes. Властивість "Text" задайте пустою, щоб ніякий текст не відображувався у полі, доки не буде обчислено результат. У групі Layout Parameters властивостей TextView задайте поле width рівним "match_parent" для коректного вирівнювання поля результату на екрані. Збережіть зміни в activity_main.xml. Остаточний вигляд activity_main.xml має бути схожим на рис. 2.1.

2.1.5 Відкрийте вихідний код додатку у файлі MainActivity.java. Змініть його з урахуванням рекомендацій п.1.1.5. Додайте два об'єкти класу EditText, а саме edit1, edit2. Проімпортуйте клас EditText. Створіть екземпляр класу Button – btnSum. Проімпортуйте клас Button. Створіть об'єкт класу TextView – tvRes. Проімпортуйте відповідний клас.

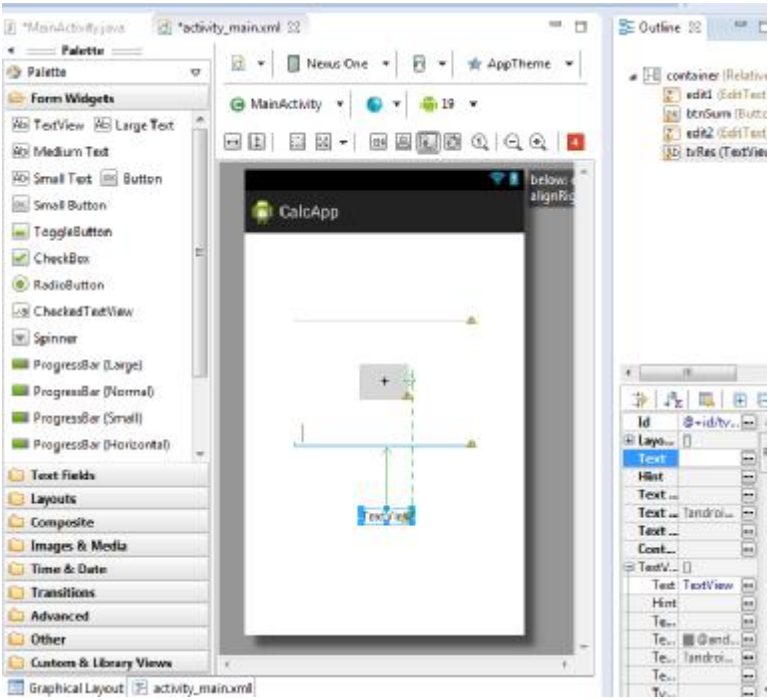


Рисунок 2.1 – Вигляд activity_main.xml

2.1.6 Далі необхідно зв'язати елементи в activity_main.xml та в коді MainActivity.java. Для цього в методі protected void onCreate(Bundle savedInstanceState) після виклику метода setContentView додайте код:

```
edit1 = (EditText) findViewById(R.id.edit1);
edit2 = (EditText) findViewById(R.id.edit2);
btnSum = (Button) findViewById(R.id.btnSum);
tvRes = (TextView) findViewById(R.id.tvRes);
```

Додайте обробник натискання кнопки, який буде читати з текстових полів, перетворювати дані на числа типу double, рахувати їх суму та виводити результат до tvRes. Для цього в методі onCreate після прив'язки елементів, прописаних в xml, до об'єктів в MainActivity.java додайте наступний код:

```
btnSum.setOnClickListener(new OnClickListener()
    {
      @Override
```

```
        public void onClick(View v)
        {
           try
           {
           String s1 = edit1.getText().toString();
           String s2 = edit2.getText().toString();
           double res =
           Double.parseDouble(s1)+Double.parseDouble(s2);
           String s3 = String.format(s1+" + "+s2+" =
                                     "+ "%.3f",res);
           tvRes.setText(s3);
           }
           catch(Exception e)
           {
             Toast.makeText(getBaseContext(),    "Введені
некоректні числа", Toast.LENGTH_LONG).show();
             tvRes.setText("");
           }

        }
           });
```

Поля вводу є джерелом типових проблем, пов'язаних з некоректним вводом значень. В даному випадку користувач повинен ввести числа (цілі, або з комою) інакше програма не зможе обчислити суму нечислових значень. Тому обробку натискання кнопки треба розміщувати у блоці перехоплення виключень try{}catch(){}.

В блоці try{} розміщуємо основний код читання текстових полів, конвертування їх значень у числа типу double, обчислення суми, форматування результату та його виводу у tvRes. Метод getText(), викликаний від екземпляру текстового поля, читає його значення та повертає результат у вигляді, який потрібно конвертувати в текст за допомогою метода toString(). Статичний метод format класу String форматує строку. Метод setText(String s) може бути викликаний від текстового поля, або від поля вводу. Він встановлює значення поля рівним своєму аргументу.

В блоці catch(Exception e) {} перехоплюємо можливі виключення. При цьому скидаємо поле tvRes, а також виводимо просте повідомлення про некоректні введені дані. Для цього використовується статичний метод *makeText* класу Toast. Він отримує 3 аргументи:

– контекст класу, що можна отримати методом getBaseContext();

– строку, що буде виведено у повідомленні;

– тривалість повідомлення, що керується одним з двох можливих значень: Toast.*LENGTH_LONG* або *Toast.LENGTH_SHORT*.

Для виводу повідомлення необхідно також викликати метод show(). В Java дозволено викликати методи ланцюжком один від одного, тому можна скористатися коротким записом

```
Toast.makeText(getBaseContext(), "Введені некоректні числа", Toast.LENGTH_LONG).show();
```

2.1.7 Запустіть додаток на виконання згідно рекомендацій п.1.1.7. Віртуальний пристрій створювати вже не потрібно. Його було налаштовано у минулій роботі. Крім того він містить усі Ваші додатки, що на ньому запускалися. Зараз же потрібно тільки прописати налаштування запуску проекта. Виконайте тестування додатку з коректними та некоректними даними у текстових полях. Результат має бути схожим на рис. 2.2.

Рисунок 2.2 – Результат виконання додатку

2.1.8 Завдання для самостійного виконання:

Варіант 1: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.calc. Додаток має містити функціонал обчислення

площі трикутника за трьома сторонами. Сторони задаються у трьох полях вводу. Результат з'являється за натисканням кнопки у короткотривалому повідомленні за допомогою класа Toast.

Примітка: для обчислення площі трикутника за трьома сторонами зручно використати формулу Герона:

$$S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)},$$

де p – напівпериметр трикутника; a, b, c – сторони трикутника.

Варіант 2: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.calc. Додаток має містити функціонал обчислення суми, різниці, добутку, ділення, квадратного кореня. Аргументи вводяться у двох полях. На кожну операцію окрема кнопка. Результати відображаються в об'єкті TextView.

2.2 Зміст звіту

2.2.1 Вміст activity_main.xml у текстовому вигляді.

2.2.2 Зображення результатів виконання програми згідно індивідуального варіанту.

2.2.3 Текст програми згідно індивідуального варіанту.

2.2.4 Відповіді на контрольні питання.

2.3 Контрольні питання

2.3.1 Поясніть налаштування activity_main.xml.

2.3.2 Як виконується обробка натискання кнопки?

2.3.3 Як прочитати вміст поля вводу, як записати туди значення?

2.3.4 Поясніть призначення класів Button та EditText.

2.3.5 Як вивести просте повідомлення на екран за допомогою класу Toast?

2.3.6 Поясніть сенс перехоплення виключень у додатках Java.

3 ЛАБОРАТОРНА РОБОТА № 3. РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЦЬКОГО МЕНЮ

Мета роботи: ознайомлення з основними способами програмування головного та контекстного меню додатку для мобільної платформи.

3.1 Порядок виконання роботи

Розробимо додаток, який буде містити головне меню, що викликається по натисканню функціональної клавіши смартфона, та контекстне меню, яке з'являється при утриманні пальця на елементі керування. Головне меню буде містити пункти "Про додаток" та "Вихід". Контекстне меню запропонує користувачу перейти на одну з двох сторінок, що присвячені інформації про кафедру комп'ютерних систем та мереж та про факультет комп'ютерних наук та технологій.

3.1.1 Створіть новий проект для платформи Android, назвіть його "MenuApp", назва пакету – "edu.zntu.menuapp".

3.1.2 Відкрийте файл строкових ресурсів /res/values/strings.xml. Додайте до нього строкові константи. Для цього натисніть кнопку Add… та виберіть елемент String. У полі Name впишіть службову назву пункту меню "about", а у полі Value – текстове значення "Про додаток" (рис. 3.1). Натисніть Ctrl+s для збереження ресурсів.

Рисунок 3.1 – Додавання строкового ресурсу

Додайте всі строкові константи, що знадобляться для пунктів меню: "exit" – "Вихід", "csn" – "Про кафедру КСМ", "fknt" – "Про

факультет КНТ". Додайте строкову константу "info" – "Інформація", що буде відображатися на головному екрані та викликати контекстне меню. Збережіть зміни у ресурсах.

3.1.3 Відкрийте activity_main.xml, змініть тип layout на RelativeLayout та додайте елемент TextView з id @+id/tvInfo та властивістю Text @string/info, що містить посилання на ресурс. Зверніть увагу, вміст TextView одразу змінився на "Інформація".

3.1.4 Відкрийте ресурси меню /res/menu/main.xml. Видаліть пункт "action_settings" та додайте пункт меню за допомогою Add… Item. Змініть Title, для цього натисніть кнопку Browse та виберіть пункт about, який підключиться зі строкових ресурсів. Змініть id на @+id/mainitem1. Встановіть властивості Visible та Enabled в true. Властивості Order in category (номер пункту меню) задайте значенням "1". За аналогією додайте ще один пункт для виходу з додатка (для цього у пункті Add виберіть Create a new element at the top level in menu). Номеру пункту меню привласніть значення "2". Змініть id на @+id/mainitem2. Збережіть xml файл.

3.1.5 Створіть пункти для контекстного меню. Для цього клацніть правою кнопкою миші по теці /res/menu та виберіть пункт New|Android XML File. Назвіть файл context1. За аналогією з п. 3.1.4 додайте два пункти контекстного меню для переходу на інформацію про кафедру або факультет (пункти зі строкових ресурсів "csn" та "fknt"). Встановіть пунктам ідентифікатори @+id/contextitem1 та @+id/contextitem2. Збережіть зміни в xml файлі.

3.1.6 Відкрийте MainActivity.java та відредагуйте його згідно з п. 1.1.5. Для програмування виклику меню та обробки реакції на вибір його пунктів необхідно у класі MainActivity після методу onCreate реалізувати методи onCreateOptionsMenu, що призначений для завантаження пунктів меню, та onOptionsItemSelected, який дозволяє запрограмувати реакцію на вибір пункту меню. Реалізацію цих методів зручно зробити наступним чином: поставте курсор після закінчення методу onCreate, у меню редактора Eclipse виберіть Source|Override/Implement Methods. Далі поставте прапорці біля методів onCreateOptionsMenu(Menu) та onOptionsItemSelected(MenuItem), що успадковані від класу Activity. Eclipse додасть реалізації цих методів в наш клас.

3.1.7 В методі onCreateOptionsMenu(Menu) необхідно прописати код перенесення пунктів меню з ресурсів. Для цього відредагуйте код метода наступним чином:

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
  getMenuInflater().inflate(R.menu.main, menu);
  return true;
}
```

Метод `getMenuInflater()` повертає екземпляр класу Inflater, що дає змогу викликати метод `inflate(R.menu.main, menu)` який призначено для створення пунктів меню з xml файлу /res/menu/main.xml. Ми прописали створення власного меню, тому необхідно повернути істину з метода onCreateOptionsMenu.

3.1.8 Для обробки вибору пунктів меню необхідно відредагувати метод onOptionsItemSelected наступним чином:

```
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
  switch(item.getItemId())
  {
    case R.id.mainitem1:
      Toast.makeText(this, "Додаток створено на кафедрі КСМ", Toast.LENGTH_SHORT).show();
      break;
    case R.id.mainitem2:
      finish();
      break;
  }
  return true;
}
```

За допомогою `item.getItemId()` отримуємо ідентифікатор пункту меню. За допомогою оператора switch перевіряємо ідентифікатор на `R.id.mainitem1`, що відповідає пункту "Про додаток" та `R.id.mainitem2` – пункт "Вихід". Метод `finish()` завершує поточний Activity. Оскільки наш Activity є головним, то його завершення призведе до завершення всього додатку.

3.1.9 Створимо ще два класи Activity, в яких буде записана коротка довідка про кафедру та факультет. Перехід на ці Activity буде відбуватися за вибором відповідного пункту контекстного меню

TextView "Інформація". Отже, клацніть правкою кнопкою миші на пакеті edu.zntu.menuapp New|Class. У полі Name вкажіть ім'я класу Csn. У полі Superclass натисніть кнопку Browse… та впишіть Activity, виберіть зі списку android.app.Activity. Натисніть кнопку Finish. В результаті буде створено новий клас, що спадкує Activity. Для нього потрібно створити власний layout. Для цього клацніть правою кнопкою миші по теці /res/layout та виберіть пункт New|Android XML File. Виберіть зі списку RelativeLayout. У полі File впишіть activity_csn. Натисніть кнопку Finish. В результаті буде створено новий layout. Розмістіть на ньому TextView. Текст, що буде в ньому відображатися краще прописати в строкових константах. Для цього переходимо до /res/values/strings.xml. Створюємо нову строку "text_csn", її значенню привласніть наступний текст: «Кафедра комп\'ютерних систем та мереж створена у 2002 р. на базі кафедри радіотехніки\nКафедра здійснює підготовку фахівців зі спеціальностей «Комп\'ютерні системи та мережі» та «Спеціалізовані комп\'ютерні системи»». Повертаємося до activity_csn, обираємо TextView та встановлюємо властивість Text у @string/text_csn. Змініть колір та розмір шрифта на свій смак.

3.1.10 Переходимо до коду класа Csn.java. Змініть його як показано далі:

```
public class Csn extends Activity
{
  @Override
  protected void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_csn);
  }
}
```

Зверніть увагу на метод `setContentView`, він прив'язує відповідний layout до класу.

3.1.10 За аналогією створіть клас Fknt.java. Створіть для нього layout activity_fknt. Запишіть строкову константу "text_fknt": "Факультет комп\'ютерних наук і технологій започатковано у 2001 році.\nВін об\'єднує кафедри комп\'ютерних систем та мереж, програмних засобів і системного аналізу та обчислювальної

математики". Пропишіть код класу Fknt.java. У методі `setContentView` прив'яжіть до класу відповідний layout.

3.1.11 Далі необхідно прописати новостворені класи у файлі маніфесту, інакше додаток не буде знати про існуючі класи. Відкрийте файл AndroidManifest.xml. Перейдіть на вкладку Application. Знайдіть внизу пункт Application Nodes. Натисніть Add… Виберіть Activity. Виберіть доданий Activity та справа у пункті Name додайте Csn за допомогою кнопки Browse… Змініть тему оформлення, для цього клацніть на кнопку Browse пункту Theme та виберіть @android:style/Theme.Black. За допомогою пункта Add… додайте до Csn Intent Filter. Збережіть зміни. За аналогією додайте в маніфест клас Fknt. Встановіть йому тему @style/Theme.Base.Light. Не забудьте додати Intent Filter.

3.1.12 Отже, ми створили всі необхідні класи та закодували їх вміст. Залишилось запрограмувати появу контекстного меню. Переходимо до MainActivity.java. Ставимо курсор після останнього метода та викликаємо Source|Override/Implement Methods. Зі списку методів вибираємо onCreateContextMenu та onContextItemSelected. Код onCreateContextMenu прописуємо аналогічно методу onCreateOptionsMenu, не забуваючи підтягнути R.menu.*context1*. Код onContextItemSelected пропишіть в наступному вигляді:

```
@Override
  public boolean onContextItemSelected(MenuItem item)
  {
    Intent anIntent;
    switch(item.getItemId())
    {
      case R.id.contextitem1:
        anIntent = new Intent(this,Csn.class);
        startActivity(anIntent);
        break;
      case R.id.contextitem2:
        anIntent = new Intent(this,Fknt.class);
        startActivity(anIntent);
        break;
    }
    return true;
  }
```

Intent – це намір змінити Activity. Екземпляр Intent використовується у якості аргумента методу startActivity, що змінює поточний Activity. Конструктор Intent(this,Csn.class) приймає два аргумента: поточний контекст та назву класу, що містить Activity, до якого буде здійснено перехід.

3.1.13 В класі MainActivity створіть екземпляр TextView tvInfo; В методі onCreate додайте код:

```
tvInfo = (TextView) findViewById(R.id.tvInfo);
tvInfo.setOnCreateContextMenuListener(this);
```

Таким чином ми прив'язали появу контекстного меню до нашого текстового елементу.

3.1.14 Запустіть розроблений додаток на виконання. Для виклику головного меню натисніть клавішу F2, що на емуляторі імітує натискання функціональної клавіші смартфона. Для виклику контекстного меню натисніть та потримайте ліву клавішу миші на текстовому елементі. Здійсніть переходи між екранами. Для повертання до попередньої активності натисніть Escape. Результат виконання додатку наведено на рис. 3.2.

Рисунок 3.2 – Результати виконання додатку

3.1.15 Завдання для самостійного виконання:

Варіант 1: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.time. Додаток має містити головне меню з чотирма пунктами: "Дата", "Час", "День тижня", "Вихід". Вибір пункта меню повинен переносити користувача на інший екран з відповідним вмістом. Вибір пункту "Вихід" завершує додаток.

Примітка: для отримання поточної дати, часу або номера дня тижня зручно скористатися класом `SimpleDateFormat`, про який згадувалося у першій лабораторній роботі. Зверніть увагу, що час, який відобразить методи цього класу, буде співпадати з часом на емуляторі.

Варіант 2: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.complectation. Головний екран додатку має містити два текстові елементи: "Процесори" та "Жорсткі диски". При виборі елементу "Процесори" має з'явитися контекстне меню з пунктами "Intel" та "AMD". Контекстне меню елементу "Жорсткі диски" має містити пункти "Seagate" та "WD". Вибір будь-якого пункту контекстного меню повинен показати користувачу відповідний екран з інформацією про виробника комплектуючих.

Примітка: для перевірки який саме елемент викликав контекстне меню, та відповідно, для правильної побудови пунктів контекстного меню, у методі onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) перевірте ідентифікатор елемента View v, наприклад, `if(v.getId() == R.id.tvInfo) getMenuInflater().inflate(R.menu.context1, menu);`

3.2 Зміст звіту

3.2.1 Код вихідної програми.

3.2.2 Код програми та зображення результатів її роботи згідно індивідуального завдання.

3.2.3 Маніфест програми згідно індивідуального завдання.

3.2.4 Відповіді на контрольні питання.

3.3 Контрольні питання

3.3.1 Як створити меню додатку, як його викликати, як прописати обробку вибору пунктів меню?

3.3.2 Як створити контекстне меню, як прописати обробку вибору пунктів контекстного меню, як розрізнити від якого елемента воно викликане?

3.3.3 Що таке Intent та для чого він потрібний?

3.3.4 Які налаштування прописуються у файлі маніфесту?

3.3.5 Як додати строкову константу?

3.3.6 Що таке стилі додатку та як вони використовуються?

4 ЛАБОРАТОРНА РОБОТА № 4. РОЗРОБКА ЗАСОБІВ ВЕДЕННЯ ДІАЛОГУ

Мета роботи: здобуття навичок програмування основних діалогів для мобільних пристроїв.

4.1 Порядок виконання роботи

Розробимо додаток для мобільної платформи Android, в якому продемонструємо організацію інтерфейсу з користувачем засобами діалогів. В Android існує три основні типи діалогів:

– TimePickerDialog, призначений для вводу часу;

– DatePickerDialog, за допомогою якого можна ввести дату;

– AlertDialog, що відображає довільний вміст з кнопками позитивного, нейтрального та негативного вибору.

На екрані додатку розмістимо три текстові елементи: "Time", "Date", "University". Натискання на елемент "Time" викличе TimePickerDialog, який за результатом вибору часу замінить напис на елементі Time на вибраний час. Аналогічно буде функціонувати елемент "Date", призначений для вибору дати. Елемент "University" викличе AlertDialog з питанням. Реакція на відповідь змінить напис на елементі на назву університета.

4.1.1 Створіть новий проект для платформи Android, назвіть його "Dialogs", назва пакету – "edu.zntu.dialogs".

4.1.2 Відкрийте activity_main.xml та конвертуйте layout в RelativeLayout. Розмістіть на ньому три елементи TextView з ідентифікаторами та написами (згори до низу): @+id/tvTime – "Time", @+id/tvDate – "Date", @+id/tvUniver – "University". Збережіть зміни.

4.1.3 Відредагуйте код MainActivity.java згідно рекомендацій п. 1.1.5. Створіть три екземпляри TextView та прив'яжіть до них відповідні елементи layout. Додайте обробку кліку до цих елементів. Оскільки у нас три елементи, що мають прослуховуватися OnClickListener, то зручно реалізувати інтерфейс OnClickListener від класу MainActivity, тоді ми зможемо виконати метод setOnClickListener від текстових елементів з аргументом this (клас MainActivity обробляє кліки). Результуючий код має бути схожим на такий:

```
public class MainActivity extends Activity
implements OnClickListener
{
  TextView tvTime, tvDate, tvUniver;
  @Override
  protected void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tvTime = (TextView) findViewById(R.id.tvTime);
    tvDate = (TextView) findViewById(R.id.tvDate);
    tvUniver = (TextView) findViewById(R.id.tvUniver);
    tvTime.setOnClickListener(this);
    tvDate.setOnClickListener(this);
    tvUniver.setOnClickListener(this);
  }
  @Override
  public void onClick(View v)
  {

  }
}
```

Метод onClick тепер реалізується прямо в класі MainActivity за рахунок реалізації інтерфейсу OnClickListener.

4.1.4 Створіть поля класу MainActivity, що будуть зберігати вибрані користувачем дату та час:

```
private int H, Min, Y, Mon, D;
private Calendar calendar = Calendar.getInstance();
```

Проімпортуйте java.util.Calendar. Його зручно використати для отримання поточної дати та часу.

Додайте в метод onClick обробку кліків текстових елементів. Вибір текстового елемента здійснюється за допомогою v.getId():

```
switch(v.getId())
     {
     case R.id.tvTime:
       break;
     case R.id.tvDate:
       break;
     case R.id.tvUniver:
       break;
     }
```

4.1.5 Пропишіть виклик TimePickerDialog в першій секції switch:

```
TimePickerDialog tpd = new TimePickerDialog(
MainActivity.this,
new OnTimeSetListener()
     {
          @Override
          public void onTimeSet(TimePicker view,
                                int hourOfDay, int minute)
          {
            H = hourOfDay;
            Min = minute;
            tvTime.setText("Time is: "+H+":"+Min);
          }
   }, calendar.get(Calendar.HOUR_OF_DAY),
      calendar.get(Calendar.MINUTE), true);
tpd.show();
```

Конструктор класу TimePickerDialog приймає 5 параметрів:

– контекст класу, в якому він викликаний;

– обробник OnTimeSetListener, в якому треба перевизначити метод onTimeSet(TimePicker view, int hourOfDay, int minute). Коли користувач вибрав час, викликається цей метод та в аргументи hourOfDay і minute передається вибраний час;

– час за замовчанням;

– хвилини за замовчанням;

– булеве значення 24 години чи AM/PM.

Метод від екземпляра TimePickerDialog викликає показ діалога.

4.1.6 Аналогічно запрограмуйте DatePickerDialog у другій секції switch. Його код наведено нижче:

```
DatePickerDialog dpd = new DatePickerDialog(
MainActivity.this,
new OnDateSetListener()
        {
          @Override
          public void onDateSet(DatePicker view,
              int year, int monthOfYear, int dayOfMonth)
          {
             Y=year;
             Mon=monthOfYear;
             D=dayOfMonth;
             tvDate.setText(D+"."+(Mon+1)+"."+Y);
          }
        },
calendar.get(Calendar.YEAR),
calendar.get(Calendar.MONTH),
calendar.get(Calendar.DAY_OF_MONTH));
dpd.show();
```

4.1.7 Для програмування AlertDialog додайте в третю секцію switch наступний код:

```
AlertDialog.Builder aDb = new AlertDialog.Builder(
MainActivity.this);
aDb.setMessage("Ви навчаєтеся в ЗНТУ?");
aDb.setCancelable(false);
aDb.setPositiveButton("Так", new
DialogInterface.OnClickListener()
        {
          @Override
          public void onClick(DialogInterface dialog,
                              int which)
          {
             Toast.makeText(MainActivity.this, "ЗНТУ",
Toast.LENGTH_LONG).show();
          }
        });
aDb.setNeutralButton("Не навчаюся", new
DialogInterface.OnClickListener()
        {
          @Override
          public void onClick(DialogInterface dialog,
                              int which)
          {
```

```
            Toast.makeText(MainActivity.this, "Не
навчаюся", Toast.LENGTH_LONG).show();
          }
       });
aDb.setNegativeButton("Інший ВНЗ", new
DialogInterface.OnClickListener()
       {
          @Override
          public void onClick(DialogInterface dialog,
                              int which)
          {
            Toast.makeText(MainActivity.this, "Інший
ВНЗ", Toast.LENGTH_LONG).show();
          }
       });
       AlertDialog aD = aDb.create(); aD.show();
```

AlertDialog створюється за допомогою внутрішнього класу AlertDialog.Builder методом create(). Спочатку будуємо вкладений клас. Метод setMessage програмує текст діалога. Метод setCancelable дозволяє відміняти діалог. За допомогою методів setPositiveButton, setNeutralButton та setNegativeButton програмуються кнопки вибору в діалозі.

4.1.8 Запустіть додаток. Він повинен бути схожим на той, що наведено на рис. 4.1.

Рисунок 4.1 – Зображення результатів роботи додатку

4.1.9 Завдання для самостійного виконання:

Варіант 1: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.guessing_game. Функціонал додатку наступний: користувач загадує число від нуля до 20. Додаток пропонує випадкове число у цьому діапазоні та показує його користувачу у діалозі. Кнопки діалогу: "<", "=", ">". Якщо число вгадане – виводиться відповідне повідомлення, якщо ні – пропонуються наступні діалоги, поки число не буде вгадане.

Варіант 2: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.time_span. Функціонал додатку наступний: користувач вводить дату та час за допомогою відповідних діалогів, додаток обчислює проміжок часу між поточною датою/часом та уведеним в діалозі і виводить результат у вигляді повідомлення.

4.2 Зміст звіту

4.2.1 Код вихідної програми.

4.2.2 Код програми та зображення результатів її роботи згідно індивідуального завдання.

4.2.3 Відповіді на контрольні питання.

4.3 Контрольні питання

4.3.1 Опишіть основні типи діалогів в додатках Android.

4.3.2 Як створити довільний діалог?

4.3.3 Що таке анонімний клас?

4.3.4 Для чого використовують вкладені класи, наприклад, AlertDialog.Builder?

4.3.5 Опишіть можливості методів класу Calendar.

5 ЛАБОРАТОРНА РОБОТА № 5. РОЗРОБКА МІКРОПРОГРАМ ДЛЯ РОБОТИ З НАКОПИЧУВАЧЕМ

Мета роботи: ознайомлення з принципами роботи зі внутрішнім накопичувачем та зовнішньою картою пам'яті мобільних пристроїв на платформі Android.

5.1 Порядок виконання роботи

Розробимо додаток, що буде записувати дані в файл до внутрішньої пам'яті пристрою та у пам'ять карти накопичувача. Додаток повинен також зчитувати ці дані. Головний екран додатку буде містити чотири кнопки: дві з них працюють з пам'яттю пристрою, інші дві – з пам'яттю карти накопичувача. Кнопка "write_file" буде записувати бренд та модель пристрою у файл в пам'ять смартфона. Кнопка "read_file", відповідно, буде зчитувати цю інформацію та виводити на екран у вигляді повідомлення. Кнопка "write_file_sd" буде записувати версію операційної системи у файл на карту SD. Відповідна їй кнопка "read_file_sd" буде зчитувати ці дані та виводити їх на екран у вигляді повідомлення.

5.1.1 Створіть новий проект для платформи Android, назвіть його "Files", назва пакету – "edu.zntu.files".

5.1.2 Відкрийте activity_main.xml та конвертуйте layout в RelativeLayout. Розмістіть на ньому чотири кнопки "write_file" (id @+id/btnwrite), "read_file" (id @+id/btnread), "write_file_sd" (id @+id/btnwritesd), "read_file_sd" (id @+id/btnreadsd) як показано на рис. 5.1.

Рисунок 5.1 – Вміст activity_main.xml

5.1.3 Відкрийте MainActivity.java та відредагуйте його згідно з п. 1.1.5. Створіть 4 поля – екземпляра класса Button: "btnWrite", "btnRead", "btnWriteSD", "btnReadSD". Також створіть екземпляри

класа String: "filePath", "SDPath", вони знадобляться для отримання шляхів по файловій системі.

5.1.4 В методі onCreate прив'яжіть кнопки до елементів Button в layout, додайте для них обробника:

```
btnWrite = (Button)findViewById(R.id.btnwrite);
btnRead = (Button)findViewById(R.id.btnread);
btnWriteSD = (Button)findViewById(R.id.btnwritesd);
btnReadSD = (Button)findViewById(R.id.btnreadsd);
btnRead.setOnClickListener(this);
btnReadSD.setOnClickListener(this);
btnWrite.setOnClickListener(this);
btnWriteSD.setOnClickListener(this);
```

Не забудьте, що наш клас MainActivity не тільки наслідує Activity, а ще й реалізує (implements) OnClickListener.

5.1.5 Пропишіть код обробника кнопок:

```
@Override
  public void onClick(View v)
  {
    filePath =
context.getFilesDir().getPath().toString() + "/MyFile";
    SDPath =
Environment.getExternalStorageDirectory().getPath().
                          toString() + "/MySDFile";
    switch(v.getId())
    {
      case R.id.btnwrite:
           break;
      case R.id.btnread:
           break;
      case R.id.btnwritesd:
           break;
      case R.id.btnreadsd:
           break;
    }
  }
```

При натисканні на будь-яку кнопку на екрані смартфона заповнюємо строкові змінні "filePath" та "SDPath". В коді показано як отримати шляхи до сховища телефона та до файлової системи карти пам'яті. Далі за допомогою оператора switch опрацьовуємо натиснуту кнопку.

В секції *btnwrite* створюємо файл в сховищі телефону, записуємо туди бренд та модель телефона. Це можна зробити наступним чином:

```
     try{
          PrintWriter fOut = new PrintWriter(
new BufferedOutputStream(
new FileOutputStream(filePath)));
          fOut.write(android.os.Build.BRAND +" "+
                     android.os.Build.MODEL);
          fOut.flush();
          fOut.close();
        }
       catch (FileNotFoundException e)
        {
          Toast.makeText(this, e.getMessage(),
                         Toast.LENGTH_LONG).show();
        }
```

Код пишемо в блоці try – catch, оскільки при роботі з файловою системою можуть виникнути виключення, наприклад, не вдається створити файл, чи записати, чи прочитати і т.і.

Для відкриття потоку, напрямленого у файл, в який будемо записувати текстові дані, зручно скористатися класом PrintWriter. Класи-потоки в Java мають конструктори, що приймають екземпляри інших класів потоків, за допомогою чого досягається фільтрування властивостей потоків. Наприклад,

```
PrintWriter fOut = new PrintWriter(
new BufferedOutputStream(
new FileOutputStream(filePath)));
```

Створюємо екземпляр класу PrintWriter, в його конструктор передаємо анонімний екземпляр класу BufferedOutputStream, що додає властивість буферизації потоку, а в його конструктор передаємо анонімний екземпляр класу FileOutputStream, що направляє потік у файл, визначений строкою filePath. Такі потоки-обгортки можна вкладувати один в одний скільки завгодно, додаючи потоку нові властивості.

Запис даних у потік досягається методом write від екземпляра класу PrintWriter. Аргумент цього метода – строка з даними, що записується у потік. Оскільки ми зробили потік буферизованим, для

очистки буфера та запису в файл необхідно викликати метод flush. Для закривання потоку викличте метод close.

В секції *btnread* читаємо вміст файла зі сховища телефона:

```
try{
     Scanner fIn = new Scanner(
new BufferedInputStream(
new FileInputStream(filePath)));
      while(fIn.hasNextLine())
        Toast.makeText(this,
        fIn.nextLine(), Toast.LENGTH_LONG).show();
      fIn.close();
    }
catch (FileNotFoundException e)
    {
      Toast.makeText(this, e.getMessage(),
                     Toast.LENGTH_LONG).show();
    }
```

Відкриваємо потік для читання Scanner, надаємо йому властивість буферизації, та направляємо в файл filePath. Доки в файлі ще є строки для читання `while(fIn.hasNextLine())`, читаємо файл построково `fIn.nextLine()` та виводимо вміст у повідомленнях Toast на екран.

В секції *btnwritesd* створюємо файл на карті пам'яті та записуємо туди версію ОС смартфона. Але перед тим перевіряємо наявність карти пам'яті. Це можна зробити наступним чином:

```
if(!Environment.getExternalStorageState().
    equals(Environment.MEDIA_MOUNTED))
      {
        Toast.makeText(this, "No SD card mouned",
                       Toast.LENGTH_LONG).show();
        break;
      }
try  {
     PrintWriter fOut = new PrintWriter(
new BufferedOutputStream(
new  FileOutputStream(SDPath)));
      fOut.write(android.os.Build.VERSION.RELEASE);
      fOut.flush();
      fOut.close();
      }
      catch (FileNotFoundException e)
```

```
        {
           Toast.makeText(this, e.getMessage(),
                          Toast.LENGTH_LONG).show();
        }
```

Робота з потоком PrintWriter аналогічна описаній в секції *btnwrite* за виключенням необхідності попередньої перевірки наявності карти пам'яті. В потік записуємо версію ОС.

В секції *btnreadsd* кодуємо читання з файла на карті пам'яті:

```
    if(!Environment.getExternalStorageState().
        equals(Environment.MEDIA_MOUNTED))
        {
           Toast.makeText(this, "No SD card mouned",
                          Toast.LENGTH_LONG).show();
           break;
        }
    try  {
           Scanner fIn = new Scanner(
new BufferedInputStream(
new FileInputStream(SDPath)));
           while(fIn.hasNextLine())
                 Toast.makeText(this, fIn.nextLine(),
                                Toast.LENGTH_LONG).show();
           fIn.close();
        }
        catch (FileNotFoundException e)
        {
           Toast.makeText(this, e.getMessage(),
                          Toast.LENGTH_LONG).show();
        }
```

Код аналогічний секції *btnread.*

Звертаємо увагу, що класи PrintWriter та Scanner добре підходять для запису та читання текстових даних у потік. Для запису/читання даних в бінарному вигляді зручно використовувати класи DataInputStream та DataOutputStream.

Відкриття потоку на запис файла зтирає його вміст. Для можливості до запису (append) у файл додайте до PrintWritter фільтр:

```
File aFile = new File(SDPath);
PrintWriter pw = new PrintWriter(
new FileWriter(aFile, true));
```

Флаг true в конструкторі фільтра FileWriter дозволяє додавання вмісту до файла.

Додайте в маніфест дозвіл на роботу з файловою системою смартфона: android.permission.WRITE_EXTERNAL_STORAGE.

5.1.6 Запустіть додаток. Його функціонал має бути схожим на рис. 5.2. Натискаємо кнопку "write_file", що записує бренд та модель пристрою у файл в пам'ять смартфону. Кнопка "read_file" зчитує ці дані та відображає їх у вигляді повідомлення. Тиснемо "write_file_sd" – версія ОС записується у файл на карту SD. Кнопка "read_file_sd" зчитує ці дані та виводить їх на екран.

Рисунок 5.2 – Функціонал додатку

5.1.7 Завдання для самостійного виконання:

Варіант 1: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.notes. Функціонал додатку наступний: користувач записує в поле текстові примітки. Натискання кнопки "write" записує примітку в файл внутрішньої пам'яті смартфона. Кнопка "read" читає всі примітки та показує їх разом на новому екрані. Кнопка "reset" знищує вміст файла.

Варіант 2: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.budget. Функціонал додатку наступний: користувач записує в поле статтю витрат, а в інше поле – її вартість у гривнях. Натискання кнопки "write" записує дані в файл карти пам'яті. Кнопка "read" читає всі статті витрат, та виводить їх у одному повідомленні Toast. Друге повідомлення має містити загальну суму витрат. Кнопка "reset" знищує вміст файла.

Примітка: для читання наступного слова з файла можна використати метод next() класа Scanner.

5.2 Зміст звіту

5.2.1 Код програми вихідного завдання.

5.2.2 Код програми та зображення результатів її роботи згідно індивідуального завдання.

5.2.3 Порівняльна таблиця потоків PrintWritter, Scanner, DataOutputStream, DataInputStream (призначення, методи).

5.2.4 Відповіді на контрольні питання.

5.3 Контрольні питання

5.3.1 Як створити файл у внутрішній пам'яті смартфона?

5.3.2 Як прочитати вміст файла з карти пам'яті смартфона?

5.3.3 Яким чином програмно отримати інформацію про пристрій Android?

5.3.4 Що таке буферизація потоку та навіщо вона потрібна?

6 ЛАБОРАТОРНА РОБОТА № 6. РОЗРОБКА СЕРВІСІВ ДЛЯ МОБІЛЬНИХ ПЛАТФОРМ

Мета роботи: здобуття навичок програмування простих сервісів для мобільних платформ.

6.1 Порядок виконання роботи

Сервіс (служба) – це компонент програми, який може виконувати тривалі операції у фоновому режимі, при цьому він не забезпечує користувацький інтерфейс. Додаток може запустити сервіс, і він продовжуватиме працювати у фоновому режимі, навіть якщо користувач перемикається в іншу програму. Додаток може зв'язуватися з сервісом, щоб, наприклад, обробляти мережеві транзакції, програвати музику, виконувати потоки вводу/виводу та інше.

Розробимо простий додаток, який буде імітувати довготривалі операції, наприклад підключення к базі даних чи передачу даних за допомогою WI-FI. Спочатку спробуємо виконати такі операції в основному класі додатку. Як переконаємося, довготривалі операції блокують інтерфейс користувача, тому їх краще винести в окремий сервіс. Далі створимо клас – сервіс, в якому розмістимо код, що імітує довготривалі операції. В основному класі запускатимемо сервіс та переконаємося, що він не блокує основний потік з графічним інтерфейсом.

6.1.1 Створіть новий проект для платформи Android, назвіть його "Services", назва пакету – "edu.zntu.services".

6.1.2 Відкрийте activity_main.xml та конвертуйте layout в RelativeLayout. Розмістіть на ньому дві кнопки "Start Service" (id @+id/btnStart) та "Stop Service" (id @+id/btnStop). Клік по кнопці "start" запустить довготривалі операції, "stop" – зупинить їх. Додайте елемент ProgressBar(Normal). Це індикатор прогресу – коло, що обертається коли основний потік виконання не заблокований. Індикатор допоможе нам зрозуміти коли інтерфейс користувача заблокований, а коли ні.

6.1.3 Відкрийте MainActivity.java та відредагуйте його згідно з п. 1.1.5. Створіть 2 поля – екземпляра класса Button: "btnStartService", "btnStopService".

6.1.4 В методі onCreate прив'яжіть кнопки до елементів Button в layout, додайте для них обробника:

```
btnStartService = (Button) findViewById(R.id.btnStart);
btnStopService = (Button) findViewById(R.id.btnStop);
btnStartService.setOnClickListener(this);
btnStopService.setOnClickListener(this);
```

Клас MainActivity повинен реалізувати OnClickListener та метод onClick:

```
@Override
  public void onClick(View v)
{
  switch(v.getId())
    {
      case R.id.btnStart:
        int counter = 10;
        for(int i=0;i<counter;i++)
        {
          try
          { Thread.sleep(1000);
            Log.d("MyService", String.valueOf(i));
          }
          catch (Exception e){}
        }
        break;
      case R.id.btnStop:
        Log.d("MyService", "Almost stoped");
        break;
    }
}
```

По натисканню кнопки "Start Service" відкриваємо цикл на 10 ітерацій, а в ньому імітуємо довготривалі операції за допомогою метода `Thread.sleep(1000)`, що заставляє потік виконання заснути на одну секунду. Після цієї секунди пишемо в логи значення номеру ітерації. Логування виконується за допомогою статичного метода `Log.d`, що отримує два параметри: перший – мітка логу, другий – текст, що логується. Не забудьте імпортувати android.util.Log. Натискання кнопки "Stop Service" запише в лог фразу "Almost stoped".

6.1.5 Для перегляду логів в Eclipse додайте вікно LogCat. Для цього активуйте пункт меню Window|Show View|Other…|LogCat. В нижній частині інтефрейса Eclipse з'явиться вкладка LogCat. Виберіть її та натисніть кнопку "+" в категорії "Saved Filters". У полі "Filter Name" впишіть назву фільтра "Service", у полі "By Log Tag" впишіть тег повідомлень, що ми логуємо, а саме, "MyService" (це перший аргумент метода *d* класу Log).

Виберіть новостворений фільтр, його вміст поки що пустий. Запустіть додаток на виконання. Він має виглядати як показано на рис. 6.1.

Рисунок 6.1 – Інтерфейс додатку

Після запуску додатка очистіть повідомлення логів кнопкою з червоним крестиком на вкладці LogCat. Натисніть кнопку "Start Service" та зверніть увагу на логи та на індикатор прогреса, що обертається. Логи заповнюються з інтервалом в одну секунду (як ми програмували), але користувацький інтерфейс зависає доки весь цикл, що імітує довготривалі операції, не буде завершено. Крім того, якщо натиснути кнопку "Stop Service" доки цикл не завершено, то додаток, скоріш за все, перестане відповідати, про що буде виведено окреме повідомлення. Саме тому оціфійна документація к Android рекомендує довготривалі операції виносити в окремі потоки виконання, які зручно використовувати разом з сервісами.

6.1.6 Створимо клас – сервіс, в який перенесемо код, що імітує довготривалі фонові операції. Для цього клацніть правою кнопкою миші по пакету edu.zntu.services та виберіть пункт New|Class. У полі Name введіть MyService, у полі Superclass – android.app.Service та натисніть Finish. Відкрийте код цього сервіса та виберіть пункт меню

Source|Override|Implement Methods. Зі списку методів, що спадкуються від суперкласу Service виберіть onCreate, onStartCommand, onDestroy, onBind. Життєвий цикл сервісу дещо відрізняється від циклу Activity. Метод onCreate виконується коли сервіс вперше створено, onStartCommand – коли сервіс запущено методом startService, onDestroy – при знищенні сервіса, onBind – повертає комунікаційний канал сервісу (в простому випадку повертає null).

6.1.7 В класі MyService створіть поле Thread aThread = null. Це поле буде використано для створення окремого потоку виконання. Створіть змінну volatile boolean stoped = false. Вона послужить індикатором завершення потоку. Модифікатор volatile робить поле єдиним для всіх потоків виконання. В методі onCreate розмістіть код:

```
super.onCreate();
Log.d("Service", "Created");
```

В ньому ми викликаємо метод суперкласу, що виконує підготовчу роботу по створенню сервіса, а також логуємо фразу – індикатор створеного сервіса.

В методі onDestroy розмістіть наступний код:

```
Log.d("MyService", "Almost destroyed");
if(aThread!=null) stoped = true;
super.onDestroy();
```

Логуємо фразу про те, що сервіс майже знищено. Перевіряємо чи був сервіс запущений коректно, якщо так, то ссилка aThread не повинна бути null, тоді встановлюємо маркер зупинки потоку виконання. Потім викликаємо метод суперкласа, що знищує сервіс.

Основний код сервіса розміщуємо в методі onStartCommand:

```
final int counter = 10;
aThread = new Thread(new Runnable()
     {
        @Override
        public void run()
        {
           for(int i=0;i<counter;i++)
           {
              try
              {
                if(stoped)
                {
                   stopSelf(); // зупинити службу
                   return;
```

```
                }
                Thread.sleep(1000);
                Log.d("MyService", String.valueOf(i));
            }
              catch (Exception e){}
            }
        }
    });
aThread.start();
return super.onStartCommand(intent, flags, startId);
```

Створюємо новий потік виконання – екземпляр класа Thread. Йому передається один аргумент – анонімний екземпляр інтерфейса Runnable, в методі run якого розміщуємо код потока виконання сервісу. В коді потоку перевіряємо мітку stoped: якщо вона встановлена (в методі onDestroy) – зупиняємо цикл, що імітує довготривалі операції. На кожній ітерації циклу, аналогічно попередньому прикладу, логуємо номер поточної ітерації. Запуск потока виконання здійснюється методом start, що викликається від екземпляра класу Thread. По завершенні метода onStartCommand викликаємо метод суперкласа: super.onStartCommand.

6.1.8 Після створення сервіса модифікуємо клас MainActivity: в секції case R.id.btnStart:

```
Intent toService = new Intent(getBaseContext(),
                              MyService.class);
startService(toService);
```

Створюємо намір запуску сервіса та запускаємо сервіс з класу MyService. В секції case R.id.btnStop:

```
stopService(new Intent(getBaseContext(),
                       MyService.class));
```

Метод stopService з MainActivity викликає метод onDestroy в MyService.

6.1.9 Оскільки ми додали клас в проект, його треба прописати у маніфесті. Відкрийте файл маніфесту. Перейдіть на вкладку Application. Знайдіть внизу пункт Application Nodes. Натисніть Add… Виберіть Service. Виберіть доданий сервіс та справа у пункті Name додайте MyService за допомогою кнопки Browse… Збережіть манфіест.

6.1.10 Очистіть логи та запустіть додаток. Зверніть увагу, що запуск сервіса не блокує основний потік виконання та графічний інтерфейс (індикатор прогресу завжди обертається). В будь-який момент можна зупинити сервіс (рис. 6.2).

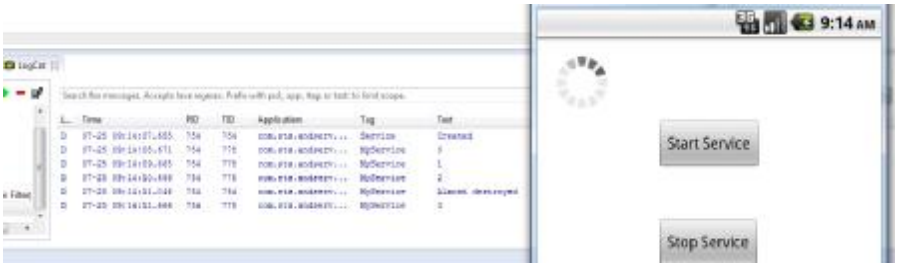


Рисунок 6.2 – Логи додатку

6.1.11 Завдання для самостійного виконання:

Варіант 1: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.writter. Функціонал додатку наступний: користувач натискає кнопку "start", яка запускає сервіс. Кнопка "stop" – зупиняє його. Сервіс повинен створити файл на карті пам'яті та записати туди масив з тисячі випадкових чисел. Далі сервіс рахує середнє арифметичне значення цих чисел та записує його в кінець файла та в лог.

Варіант 2: Створіть новий додаток. Пакет додатку назвіть com.ВАШЕ_ІМ'Я.tokenizer. Функціонал додатку наступний: користувач натискає кнопку "start", яка запускає сервіс. Кнопка "stop" – зупиняє його. Сервіс повинен створити файл у внутрішній пам'яті смартфона та записати туди прізвища, ім'я та по-батькові трьох ваших одногрупників. Далі сервіс читає файл і рахує яка буква найчастіше зустрічається та записує її в лог.

6.2 Зміст звіту

6.2.1 Код програми вихідного завдання.

6.2.2 Код програми та зображення результатів її роботи згідно індивідуального завдання.

6.2.3 Відповіді на контрольні питання.

6.3 Контрольні питання

6.3.1 Для чого призначені сервіси та в чому їх відмінність від Activity?

6.3.2 Як створити сервіс? В яких методах програмується його робота?

6.3.3 Що таке потік виконання? Як його створити, запустити та зупинити?

6.3.4 Для чого використовують логування? Як записати в лог довільне повідомлення?

ЛІТЕРАТУРА

1 Таненбаум Э. Архитектура компьютера. Шестое издание / Э. Таненбаум, Т. Остин. – СПб.: Питер, 2013. – 816 с.
2 Noel Markham-Java Programming Interviews Exposed / Wrox, 2014. – 386 с.
3 С. Хашими, С. Коматинени, Д. Маклин Разработка приложений для Android / Питер, 2011. – 280 с.
4 Рето Майер Android. Программирование приложений для планшетных компьютеров и смартфонов / Эксмо, 2011. – 310 с.
5 Холленд Р. Микропроцессоры и операционные системы: Краткое справочное пособие: / Пер. с англ.-М.: Энергоатомиздат,1991.– 192 с.
6 Google Android… это несложно [Электронный ресурс]. – Режим доступа: http://startandroid.ru/ru/