МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Полоцкий государственный университет»

Факультет информационных технологий Кафедра технологий программирования

# Лабораторная работа №4

по дисциплине: «Программируемые мобильные системы»

на тему: «Вызов Активности с помощью явного намерения и получение результатов работы.

Использование неявных Намерений. Получение данных из Намерения»

Вариант №4

Выполнил Студент группы 21-ИТ-2

Зинькевич И.Ю.

Проверил Ассистент

Васильева Д.М.

Полоцк 2023

**Цель работы:** научиться вызывать Активность с использованием явного намерения и получать результаты её работы. Научиться использовать неявные Намерения и получать данные из Намерения.

Ход работы:

1. Продолжить разработку проекта из Л.р.2 и Л.р.3.

2. Добавить в приложение вызов явного и неявного намерения.

3. Модифицировать методы onCreate Активностей так, чтобы с помощью Toast они показывали действие вызвавшего их Намерения.

# Полный листинг программы:

# DatabaseHelper.java

# public class DatabaseHelper extends SQLiteOpenHelper {

# private static final String DATABASE\_NAME = "mydatabase.db";

# private static final int DATABASE\_VERSION = 1;

# //USERS/////////////////////////////////////////////////////////////////////////////////////////

# private static final String TABLE\_USERS = "users";

# private static final String USER\_COLUMN\_ID = "id";

# private static final String USER\_COLUMN\_LOGIN = "login";

# private static final String USER\_COLUMN\_PASSWORD = "password";

# //TICKETS///////////////////////////////////////////////////////////////////////////////////////

# private static final String TABLE\_TICKETS = "tickets";

# private static final String TICKET\_COLUMN\_ID = "id";

# private static final String TICKET\_COLUMN\_DIRECTION\_FROM = "direction\_from";

# private static final String TICKET\_COLUMN\_DIRECTION\_TO = "direction\_to";

# private static final String TICKET\_COLUMN\_PRICE = "price";

# public DatabaseHelper(Context context) {

# super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

# }

# @Override

# public void onCreate(SQLiteDatabase db) {

# String createUserTableQuery = "CREATE TABLE " + TABLE\_USERS +

# "(" + USER\_COLUMN\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

# USER\_COLUMN\_LOGIN + " TEXT, " +

# USER\_COLUMN\_PASSWORD + " TEXT)";

# db.execSQL(createUserTableQuery);

# String createTicketTableQuery = "CREATE TABLE " + TABLE\_TICKETS +

# "(" + TICKET\_COLUMN\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

# TICKET\_COLUMN\_DIRECTION\_FROM + " TEXT, " +

# TICKET\_COLUMN\_DIRECTION\_TO + " TEXT, " +

# TICKET\_COLUMN\_PRICE + " TEXT)";

# db.execSQL(createTicketTableQuery);

# createTicketsRecords(db);

# }

# @Override

# public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

# String dropTableUsersQuery = "DROP TABLE IF EXISTS " + TABLE\_USERS;

# db.execSQL(dropTableUsersQuery);

# String dropTableTicketsQuery = "DROP TABLE IF EXISTS " + TABLE\_TICKETS;

# db.execSQL(dropTableTicketsQuery);

# onCreate(db);

# }

# private void createTicketsRecords(SQLiteDatabase db) {

# ContentValues ticketValues = new ContentValues();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Miory");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Novopolotsk");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 9.30);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Verhnedvinsk");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Ropno");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 2.27);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Polotsk");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Miory");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 8.20);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Miory");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Verhnedvinsk");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 4.95);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Polotsk");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Minsk");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 15.00);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Glubokoe");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Naroch");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 12.35);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Smolevichi");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Kopachi");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 34.12);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Vitebsk");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Braslav");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 20.50);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Minsk");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Vitebsk");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 14.70);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Miory");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Odessa");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 250.40);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# ticketValues.clear();

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_FROM, "Moscow");

# ticketValues.put(TICKET\_COLUMN\_DIRECTION\_TO, "Miory");

# ticketValues.put(TICKET\_COLUMN\_PRICE, 0.99);

# db.insert(TABLE\_TICKETS, null, ticketValues);

# }

# //USERS/////////////////////////////////////////////////////////////////////////////////////////

# public long addUser(String login, String password) {

# SQLiteDatabase db = this.getWritableDatabase();

# ContentValues values = new ContentValues();

# values.put(USER\_COLUMN\_LOGIN, login);

# values.put(USER\_COLUMN\_PASSWORD, password);

# long id = db.insert(TABLE\_USERS, null, values);

# db.close();

# return id;

# }

# public void updateUser(int id, String login, String password) {

# SQLiteDatabase db = this.getWritableDatabase();

# ContentValues values = new ContentValues();

# values.put(USER\_COLUMN\_LOGIN, login);

# values.put(USER\_COLUMN\_PASSWORD, password);

# db.update(TABLE\_USERS, values, USER\_COLUMN\_ID + " = ?", new

# String[]{String.valueOf(id)});

# db.close();

# }

# public void deleteUser(String login) {

# SQLiteDatabase db = this.getWritableDatabase();

# db.delete(TABLE\_USERS, USER\_COLUMN\_LOGIN + " = ?", new

# String[]{String.valueOf(login)});

# db.close();

# }

# public boolean authorize(String login, String password) {

# String selectQuery = "SELECT \* FROM " + TABLE\_USERS + " WHERE " + USER\_COLUMN\_LOGIN + " = ?";

# SQLiteDatabase db = this.getWritableDatabase();

# Cursor cursor = db.rawQuery(selectQuery, new String[]{login});

# boolean authorized = false;

# if (cursor.moveToFirst()) {

# String storedPassword = cursor.getString(cursor.getColumnIndex(USER\_COLUMN\_PASSWORD));

# if (password.equals(storedPassword)) {

# authorized = true;

# }

# }

# cursor.close();

# db.close();

# return authorized;

# }

# public List<User> getAllUsers() {

# List<User> users = new ArrayList<>();

# String selectQuery = "SELECT \* FROM " + TABLE\_USERS;

# SQLiteDatabase db = this.getWritableDatabase();

# Cursor cursor = db.rawQuery(selectQuery, null);

# if (cursor.moveToFirst()) {

# do {

# int id = cursor.getInt(cursor.getColumnIndex(USER\_COLUMN\_ID));

# String login =

# cursor.getString(cursor.getColumnIndex(USER\_COLUMN\_LOGIN));

# String password =

# cursor.getString(cursor.getColumnIndex(USER\_COLUMN\_PASSWORD));

# User contact = new User(id, login, password);

# users.add(contact);

# } while (cursor.moveToNext());

# }

# cursor.close();

# db.close();

# return users;

# }

# //TICKETS/////////////////////////////////////////////////////////////////////////////////////////

# public List<Ticket> getAllTickets() {

# List<Ticket> tickets = new ArrayList<>();

# String selectQuery = "SELECT \* FROM " + TABLE\_TICKETS;

# SQLiteDatabase db = this.getWritableDatabase();

# Cursor cursor = db.rawQuery(selectQuery, null);

# if (cursor.moveToFirst()) {

# do {

# int id = cursor.getInt(cursor.getColumnIndex(TICKET\_COLUMN\_ID));

# String directionFrom = cursor.getString(cursor.getColumnIndex(TICKET\_COLUMN\_DIRECTION\_FROM));

# String directionTo = cursor.getString(cursor.getColumnIndex(TICKET\_COLUMN\_DIRECTION\_TO));

# double price = cursor.getDouble(cursor.getColumnIndex(TICKET\_COLUMN\_PRICE));

# Ticket ticket = new Ticket(id, directionFrom, directionTo, price);

# tickets.add(ticket);

# } while (cursor.moveToNext());

# }

# cursor.close();

# db.close();

# return tickets;

# }

# public long addTicket(String direction\_from, String direction\_to, double price) {

# SQLiteDatabase db = this.getWritableDatabase();

# ContentValues values = new ContentValues();

# values.put(TICKET\_COLUMN\_DIRECTION\_FROM, direction\_from);

# values.put(TICKET\_COLUMN\_DIRECTION\_TO, direction\_to);

# values.put(TICKET\_COLUMN\_PRICE, price);

# long id = db.insert(TABLE\_TICKETS, null, values);

# db.close();

# return id;

# }

# }

# MainActivity.java

# public class MainActivity extends AppCompatActivity {

# private ListView listViewTickets;

# private ListView listViewCart;

# private ArrayAdapter<Ticket> ticketAdapter;

# private ArrayAdapter<Ticket> cartAdapter;

# private List<Ticket> allTickets;

# private EditText editTextFrom;

# private EditText editTextTo;

# private TextView textViewTotalPrice;

# String userLogin;

# @Override

# protected void onCreate(Bundle savedInstanceState) {

# super.onCreate(savedInstanceState);

# setContentView(R.layout.activity\_main);

# getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT\_INPUT\_STATE\_ALWAYS\_HIDDEN);

# listViewTickets = findViewById(R.id.listViewTickets);

# listViewCart = findViewById(R.id.listViewСart);

# editTextFrom = findViewById(R.id.editTextFrom);

# editTextTo = findViewById(R.id.editTextTo);

# textViewTotalPrice = findViewById(R.id.textViewTotalPrice);

# userLogin = getIntent().getStringExtra("userLogin");

# setTitle("Glad to see you, " + userLogin + "!");

# TabHost tabHost = (TabHost) findViewById(R.id.tabHost);

# tabHost.setup();

# TabHost.TabSpec tabSpec = tabHost.newTabSpec("tag1");

# // ----------ВКЛАДКИ------------------------------------------------------------------------------------------------------------------

# tabSpec = tabHost.newTabSpec("tag1");

# tabSpec.setContent(R.id.tab1);

# tabSpec.setIndicator("Tickets");

# tabHost.addTab(tabSpec);

# tabSpec = tabHost.newTabSpec("tag2");

# tabSpec.setContent(R.id.tab2);

# tabSpec.setIndicator("Shop Cart");

# tabHost.addTab(tabSpec);

# tabHost.setCurrentTab(0);

# ticketsLoad();

# cartAdapter = new ArrayAdapter<Ticket>(this, android.R.layout.simple\_list\_item\_1) {

# @Override

# public View getView(int position, View convertView, ViewGroup parent) {

# if (convertView == null) {

# convertView = LayoutInflater.from(getContext()).inflate(android.R.layout.simple\_list\_item\_1, parent, false);

# }

# TextView textView = convertView.findViewById(android.R.id.text1);

# Ticket ticket = getItem(position);

# String ticketInfo = "Ticket № " + ticket.getId() + "\n" +

# "Direction: " + ticket.getDirectionFrom() + " - " + ticket.getDirectionTo() + "\n" +

# "Price: " + ticket.getPrice() + " BYN";

# SpannableStringBuilder spannableStringBuilder = new SpannableStringBuilder(ticketInfo);

# ForegroundColorSpan grayColorSpan = new ForegroundColorSpan(Color.GRAY);

# RelativeSizeSpan smallSizeSpan = new RelativeSizeSpan(0.8f);

# spannableStringBuilder.setSpan(grayColorSpan, 0, 6, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# spannableStringBuilder.setSpan(smallSizeSpan, 0, 6, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# ForegroundColorSpan purpleColorSpan = new ForegroundColorSpan(Color.parseColor("#ff33ff"));

# spannableStringBuilder.setSpan(purpleColorSpan, ticketInfo.indexOf("Direction:"), ticketInfo.indexOf("Direction:") + 10, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# ForegroundColorSpan goldColorSpan = new ForegroundColorSpan(Color.parseColor("#FFD700"));

# RelativeSizeSpan largeSizeSpan = new RelativeSizeSpan(1.2f);

# StyleSpan boldStyleSpan = new StyleSpan(Typeface.BOLD);

# spannableStringBuilder.setSpan(goldColorSpan, ticketInfo.indexOf("Price:"), ticketInfo.length(), Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# spannableStringBuilder.setSpan(largeSizeSpan, ticketInfo.indexOf("Price:"), ticketInfo.length(), Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# spannableStringBuilder.setSpan(boldStyleSpan, ticketInfo.indexOf("Price:"), ticketInfo.length(), Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# textView.setText(spannableStringBuilder);

# return convertView;

# }

# };

# listViewCart.setAdapter(cartAdapter);

# listViewTickets.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {

# @Override

# public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {

# try {

# Ticket selectedTicket = (Ticket) parent.getItemAtPosition(position);

# ArrayAdapter<Ticket> cartAdapter = (ArrayAdapter<Ticket>) listViewCart.getAdapter();

# cartAdapter.add(selectedTicket);

# updateTotalPrice();

# cartAdapter.notifyDataSetChanged();

# return true;

# } catch (Exception e) {

# e.printStackTrace();

# Toast.makeText(MainActivity.this, "Ex: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

# return false;

# }

# }

# });

# }

# private void ticketsLoad() {

# try {

# DatabaseHelper dbHelper = new DatabaseHelper(this);

# allTickets = dbHelper.getAllTickets();

# ticketAdapter = new ArrayAdapter<Ticket>(this, android.R.layout.simple\_list\_item\_1, allTickets) {

# @Override

# public View getView(int position, View convertView, ViewGroup parent) {

# if (convertView == null) {

# convertView = LayoutInflater.from(getContext()).inflate(android.R.layout.simple\_list\_item\_1, parent, false);

# }

# TextView textView = convertView.findViewById(android.R.id.text1);

# Ticket ticket = getItem(position);

# String ticketInfo = "Ticket № " + ticket.getId() + "\n" +

# "Direction: " + ticket.getDirectionFrom() + " - " + ticket.getDirectionTo() + "\n" +

# "Price: " + ticket.getPrice() + " BYN";

# SpannableStringBuilder spannableStringBuilder = new SpannableStringBuilder(ticketInfo);

# ForegroundColorSpan grayColorSpan = new ForegroundColorSpan(Color.GRAY);

# RelativeSizeSpan smallSizeSpan = new RelativeSizeSpan(0.8f);

# spannableStringBuilder.setSpan(grayColorSpan, 0, 6, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# spannableStringBuilder.setSpan(smallSizeSpan, 0, 6, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# ForegroundColorSpan purpleColorSpan = new ForegroundColorSpan(Color.parseColor("#ff33ff"));

# spannableStringBuilder.setSpan(purpleColorSpan, ticketInfo.indexOf("Direction:"), ticketInfo.indexOf("Direction:") + 10, Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# ForegroundColorSpan goldColorSpan = new ForegroundColorSpan(Color.parseColor("#FFD700"));

# RelativeSizeSpan largeSizeSpan = new RelativeSizeSpan(1.2f);

# StyleSpan boldStyleSpan = new StyleSpan(Typeface.BOLD);

# spannableStringBuilder.setSpan(goldColorSpan, ticketInfo.indexOf("Price:"), ticketInfo.length(), Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# spannableStringBuilder.setSpan(largeSizeSpan, ticketInfo.indexOf("Price:"), ticketInfo.length(), Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# spannableStringBuilder.setSpan(boldStyleSpan, ticketInfo.indexOf("Price:"), ticketInfo.length(), Spanned.SPAN\_EXCLUSIVE\_EXCLUSIVE);

# textView.setText(spannableStringBuilder);

# return convertView;

# }

# };

# listViewTickets.setAdapter(ticketAdapter);

# } catch (Exception e) {

# Toast.makeText(this, "Ex: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

# e.printStackTrace();

# }

# }

# public void filterTickets(View view) {

# String from = editTextFrom.getText().toString().trim();

# String to = editTextTo.getText().toString().trim();

# if (from.isEmpty() && to.isEmpty()) {

# ticketsLoad();

# } else {

# List<Ticket> filteredTickets = new ArrayList<>();

# for (Ticket ticket : allTickets) {

# boolean matchFrom = from.isEmpty() || ticket.getDirectionFrom().equalsIgnoreCase(from);

# boolean matchTo = to.isEmpty() || ticket.getDirectionTo().equalsIgnoreCase(to);

# if (matchFrom && matchTo) {

# filteredTickets.add(ticket);

# }

# }

# ticketAdapter.clear();

# ticketAdapter.addAll(filteredTickets);

# ticketAdapter.notifyDataSetChanged();

# }

# }

# public void clearFilters(View view) {

# editTextFrom.getText().clear();

# editTextTo.getText().clear();

# ticketsLoad();

# }

# public void clearCart(View view) {

# cartAdapter.clear();

# updateTotalPrice();

# }

# private void updateTotalPrice() {

# try {

# double totalPrice = 0;

# for (int i = 0; i < cartAdapter.getCount(); i++) {

# Ticket ticket = cartAdapter.getItem(i);

# totalPrice += ticket.getPrice();

# }

# if (totalPrice == 0) {

# textViewTotalPrice.setText("There is no tickets yet!");

# } else {

# DecimalFormat decimalFormat = new DecimalFormat("#0.00"); // Форматирование до двух знаков после запятой

# String formattedPrice = decimalFormat.format(totalPrice);

# textViewTotalPrice.setText("Total Price: " + formattedPrice + " BYN");

# }

# } catch (Exception e) {

# Toast.makeText(this, "Ex: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

# e.printStackTrace();

# }

# }

# public void buyCart(View view) {

# if(textViewTotalPrice.getText() != "There is no tickets yet!")

# {

# Toast.makeText(MainActivity.this, textViewTotalPrice.getText() + " were withdrawn from the card!", Toast.LENGTH\_LONG).show();

# clearCart(view);

# DatabaseHelper dbHelper = new DatabaseHelper(this);

# dbHelper.deleteUser(userLogin);

# redirectToSignIn();

# Toast.makeText(MainActivity.this, "Also your account was deleted :3 byyeeeeeee", Toast.LENGTH\_LONG).show();

# }

# }

# public void redirectToSignIn() {

# Intent intent = new Intent(this, SignInActivity.class);

# startActivity(intent);

# }

# }

# SignInActivity.java package com.example.pms2;

# public class SignInActivity extends AppCompatActivity {

# @Override

# protected void onCreate(Bundle savedInstanceState) {

# super.onCreate(savedInstanceState);

# setContentView(R.layout.sign\_in);

# }

# public void onSignUpButtonClick(View view) {

# Intent intent = new Intent(this, SignUpActivity.class);

# startActivity(intent);

# }

# public void onSignInClick(View view) {

# try {

# EditText editTextLogin = findViewById(R.id.editTextLogin);

# EditText editTextPassword = findViewById(R.id.editTextPassword);

# String login = editTextLogin.getText().toString();

# String password = editTextPassword.getText().toString();

# DatabaseHelper dbHelper = new DatabaseHelper(this);

# boolean authorized = dbHelper.authorize(login, password);

# if (authorized) {

# Toast.makeText(this, "Vse cool!!!!", Toast.LENGTH\_SHORT).show();

# redirectToMain(login);

# } else {

# Toast.makeText(this, "Ne =)))", Toast.LENGTH\_SHORT).show();

# }

# } catch (Exception e) {

# Toast.makeText(this, "Exception: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

# e.printStackTrace();

# }

# }

# public void redirectToSignUp(View view) {

# Intent intent = new Intent(this, SignUpActivity.class);

# startActivity(intent);

# }

# public void redirectToMain(String userLogin) {

# Intent intent = new Intent(this, MainActivity.class);

# intent.putExtra("userLogin", userLogin);

# startActivity(intent);

# }

# }

# SignUpActivity.java

# public class SignUpActivity extends AppCompatActivity {

# @Override

# protected void onCreate(Bundle savedInstanceState) {

# super.onCreate(savedInstanceState);

# setContentView(R.layout.sign\_up);

# }

# public void onSignUpClick(View view) {

# try {

# EditText editTextLogin = findViewById(R.id.editTextLogin);

# EditText editTextPassword = findViewById(R.id.editTextPassword);

# String login = editTextLogin.getText().toString();

# String password = editTextPassword.getText().toString();

# DatabaseHelper dbHelper = new DatabaseHelper(this);

# long userId = dbHelper.addUser(login, password);

# Toast.makeText(this, "Nice password, " + login + "! Your id is " + userId, Toast.LENGTH\_SHORT).show();

# Toast.makeText(this, "Come and sign in now!!!!!!" + userId, Toast.LENGTH\_SHORT).show();

# } catch (Exception e) {

# Toast.makeText(this, "Exception: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

# e.printStackTrace();

# }

# }

# public void redirectToSignIn(View view) {

# Intent intent = new Intent(this, SignInActivity.class);

# startActivity(intent);

# }

# }

**Вывод:** в результате выполнения лабораторной работы научился вызывать Активность с использованием явного намерения и получать результаты её работы. Научился использовать неявные Намерения и получать данные из Намерения.