ЗАНЯТИЕ 2.14

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Тема: Работа с файловой системой Android-устройства. Изучение хранилища Shared Preferences.

Упражнение №1

В данном упражнении будет изучена работа с файловой системой Android (внутренним хранилищем).

Создайте новое Android-приложение для телефона с использованием шаблона Empty Activity (или Empty View Activity, в зависимости от версии Android Studio).

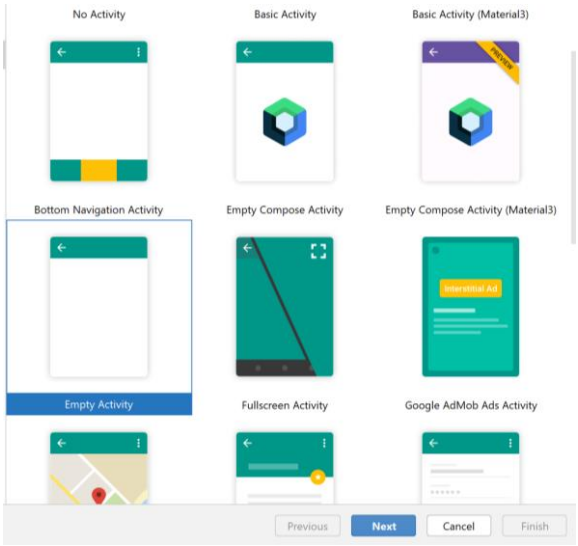

Рисунок 1

Установите минимальную версию API 21. **Язык разработки – Java (не Kotlin)**. Имя приложения – DataStorageApplication.

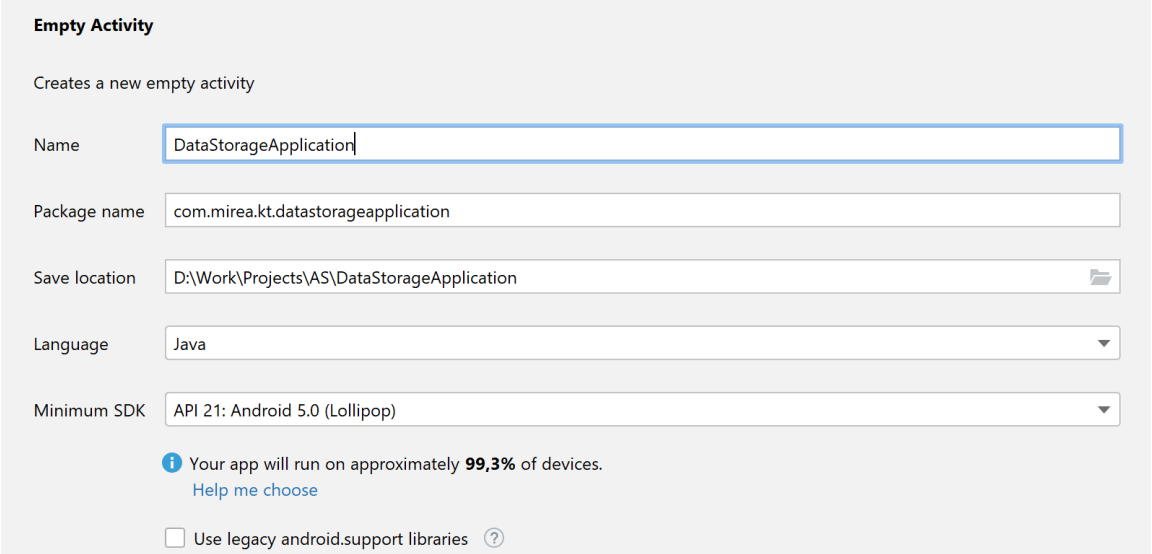

Рисунок 2

Дождитесь создания приложения и корректного обновления плагина Gradle. Перейдите в файл макета (в режим Code) **activity_main.xml**. Добавьте в макет поле ввода EditText и кнопку Button. Обратите внимание, что значения текста и цвета указываются из файлов ресурсов strings.xml и colors.xml соответственно:

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:id="@+id/etValue"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_margin="8dp"
        android:background="@color/grey"
        android:hint="@string/enter_text"
        android:gravity="start"
        android:padding="4dp"
        android:textColorHint="@color/white"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/save"
        android:textSize="24sp"
        android:layout_marginTop="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etValue" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 3

Макет в итоге должен выглядеть следующим образом:

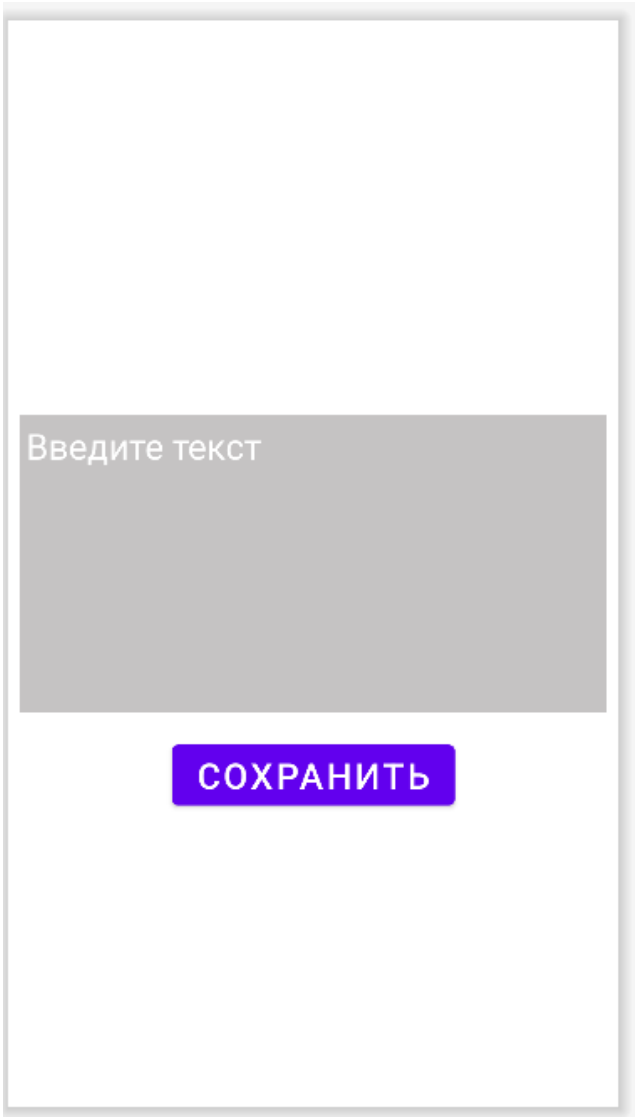

Рисунок 4

Для работы с файлами хорошей практикой является создание отдельного класса, который будет содержать необходимые методы. Создайте в проекте новый класс FilesUtils.

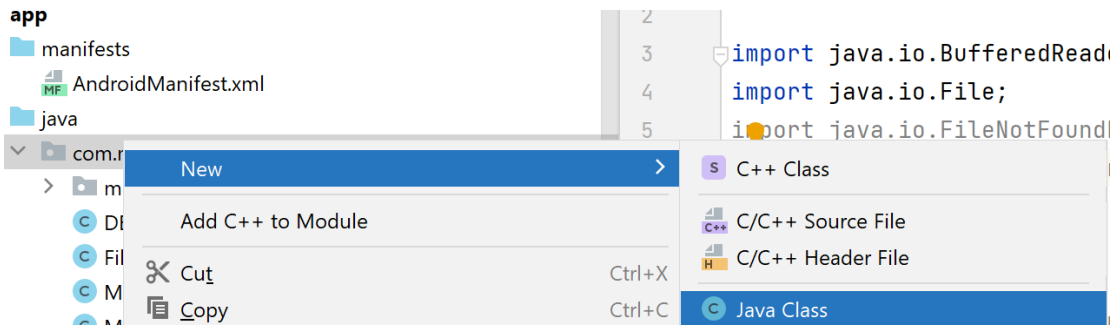

Рисунок 5

Создайте в этом классе два метода, первый из которых будет отвечать за сохранение текста в файл, второй – за загрузку текста из файла.

```
public class FilesUtils {

    public static void saveTextToFile(File file, String text) throws IOException {
        FileWriter fw = new FileWriter(file, append: false);
        fw.write(text);
        fw.close();
    }

    public static String loadTextFromFile(File file) throws IOException {
        FileReader fr = new FileReader(file);
        BufferedReader br = new BufferedReader(fr);
        StringBuilder sb = new StringBuilder();
        String line;
        while ((line = br.readLine()) != null) {
            sb.append(line);
        }
        return sb.toString();
    }

}
```

Рисунок 6

Примечание: если названия классов подчеркиваются красным (ошибка), убедитесь, что вы импортировали их.

Вернитесь в класс вашей activity и реализуйте обработчик нажатия на кнопку «Сохранить» таким образом, чтобы введенный текст сохранялся во внутренней памяти устройства (internal storage) в папку files приватного каталога вашего приложения (/data/user/0/[PACKAGE_NAME]/files или аналогичный: data/data//[PACKAGE_NAME]/files/). Имя файла можно использовать любое. В данном примере имя файла – simple_text.txt.

Обратите внимание, что в качестве текстового значения тэга в методе логирования используется финализированная статическая переменная LOG_TAG (имя можно задать любое).

Текстовые значения используются из файла ресурсов strings.xml:

```
<string name="enter_text">Введите текст</string>
<string name="save">Сохранить</string>
<string name="no_empty_field">Поле ввода не может быть пустым</string>
```

Рисунок 7

```java
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private EditText et;
    public static final String LOG_TAG = "my_app_tag";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn = findViewById(R.id.btnSave);
        et = findViewById(R.id.etValue);
        btn.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        String text = et.getText().toString();
        if(!text.isEmpty()){
            File dirPath = getFilesDir(); // получили путь до папки files приватного каталога приложения
            File filePath = new File(dirPath, child: "simple_text.txt");
            try {
                FilesUtils.saveTextToFile(filePath, text);
                Log.d(LOG_TAG, msg: "File saved to: " + filePath.getAbsolutePath());
            } catch (IOException e) {
                Log.e(LOG_TAG, msg: "Error saving: " + e.getMessage());
            }
        }else{
            Toast.makeText( context: this, R.string.no_empty_field, Toast.LENGTH_LONG).show();
        }
    }
```

Рисунок 8

Запустите приложение, убедитесь, что в окне Logcat отображаются логи, оповещающие об успешном сохранении файла.

Перейдите в файл макета и добавьте в этот же контейнер ConstraintLayout ниже еще одну кнопку – «Загрузить»:

```xml
<Button
    android:id="@+id/btnLoad"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/load"
    android:textSize="24sp"
    android:layout_marginTop="16dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btnSave"/>
```

Рисунок 9

Убедитесь, что кнопка корректно отображается в режиме макета.

Перейдите в класс вашей activity и САМОСТОЯТЕЛЬНО напишите код загрузки содержимого сохраненного файла в поле EditText по нажатию на

кнопку «Загрузить». При этом метод **onClick** должен обрабатывать нажатия на обе кнопки (используйте конструкцию if-else if).

Упражнение №2

Продолжайте работу в текущем приложении.

В этом упражнении будут изучены особенности работы с внешней памятью устройства.

Перейдите в файл манифеста AndroidManifest.xml и внутри тега **manifest** добавьте в него записи о новых разрешениях (permission), которые будут использовать приложение.

```
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="29"/>
<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />
```

Рисунок 10

Разрешение WRITE_EXTERNAL_STORAGE будет корректно работать до Android 10. Для более новых версий необходим другой тип разрешения на доступ к external storage – MANAGE_EXTERNAL_STORAGE. Данное разрешение также позволяет получать доступ ко всей внешней памяти устройства. Однако, можно обратить внимание, что IDE выделяет его желтым цветом. В данном случае это означает, что разрешение входит в список «особо контролируемых» и все приложения, которые его используют должны пройти дополнительную верификацию при публикации в Google Play. В рамках образовательного проекта его можно использовать без проверки. Необходимо лишь реализовать запрос на использование этого разрешения у пользователя.

Разрешение WRITE_EXTERNAL_STORAGE будет запрашиваться стандартным способом (в основном все «опасные» разрешения запрашиваются таким способом), а разрешение MANAGE_EXTERNAL_STORAGE необходимо запросить с помощью неявного намерения.

Перейдите в класс activity и создайте там новый метод requestPermissions:

```
private void requestPermissions(){

}
```

Рисунок 11

Также создайте метод checkPermission, который будет возвращать true или false зависимости от того, получено разрешение или нет.

```
private boolean checkPermission(){
    if (VERSION.SDK_INT >= VERSION_CODES.R) {
        return Environment.isExternalStorageManager();
    } else if (VERSION.SDK_INT >= VERSION_CODES.M) {
        return ContextCompat.checkSelfPermission( context: this, WRITE_EXTERNAL_STORAGE)
                == PackageManager.PERMISSION_GRANTED;
    } else {
        return true;
    }
}
```

Рисунок 12

Добавьте код в метод requestPermissions. В зависимости от версии Android будет запрашиваться соответствующее разрешение:

```
private void requestPermissions(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
        try {
            Intent intent = new Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERMISSION);
            intent.addCategory("android.intent.category.DEFAULT");
            intent.setData(Uri.parse(String.format("package:%s", getPackageName())));
            startActivityForResult(intent, requestCode: 12345);
        } catch (Exception e) {
            Intent intent = new Intent();
            intent.setAction(Settings.ACTION_MANAGE_ALL_FILES_ACCESS_PERMISSION);
            startActivityForResult(intent, requestCode: 12345);
        }
    } else {
        ActivityCompat.requestPermissions( activity: this,
                new String[] {WRITE_EXTERNAL_STORAGE },
                requestCode: 12345);
    }
}
```

Рисунок 13

Теперь необходимо обработать ответ пользователя – разрешил он использование приложением внешней памяти или нет. Для обработки результата полученного (или не полученного permission) запрошенного методом ActivityCompat.requestPermissions необходимо переопределить в activity метод onRequestPermissionsResult. Так как для получения разрешения MANAGE_EXTERNAL_STORAGE использовался обычный запуск activity, методом startActivityForResult, то проверить вернувшийся результат можно, переопределив метод activity – onActivityResult.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if (requestCode == 12345) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            if (checkPermission()) {
                Toast.makeText( context: this, text: "Разрешение получено!",Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText( context: this, text: "Разрешение не получено!",Toast.LENGTH_LONG).show();
            }
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == 12345) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText( context: this, text: "Разрешение получено!",Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText( context: this, text: "Разрешение не получено!",Toast.LENGTH_LONG).show();
        }
    }
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
}
```

Рисунок 14

Реализуйте запрос по нажатию на любую из кнопок:

```
@Override
public void onClick(View v) {
    if(!checkPermission()){
        requestPermissions();
    }else{
        // здесь можно писать код
        // для работы с файлами в external storage
    }
//...
}
```

Рисунок 15

Запустите приложение, убедитесь, что запрос появляется при нажатии на кнопку. После получения разрешения вы можете обращаться к файловой системе внешней памяти устройства (см. лекцию №2.11). САМОСТОЯТЕЛЬНО напишите код сохранения содержимого поля ввода в файл в external storage (по любому пути) и обратную операцию загрузки текста из сохраненного файла в поле EditText.

Упражнение №3

Продолжайте работу в текущем приложении.

В этом упражнении будут изучены особенности работы с ресурсами Assets, а также хранилищем Shared Preferences.

Создайте новую activity – DataActivity. Используйте Empty шаблон. Назначьте ее **запускающей (launcher)**. То есть теперь приложение будет стартовать именно с этой activity.

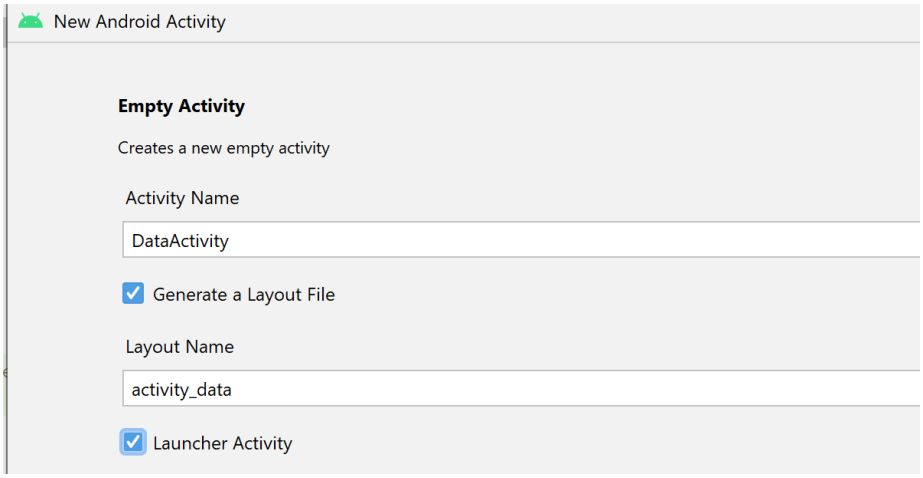

Рисунок 16

Следует обратить внимание, что теперь в файле манифеста две activity помечены в IntentFilter как MAIN и LAUNCHER. Запускаться первой будет та, которая расположена выше в файле AndroidManifest.xml.

Перейдите в файл макета созданной activity (activity_data.xml) и поместите туда поле ввода и две кнопки «Сохранить» и «Загрузить» (можно скопировать из предыдущего упражнения).

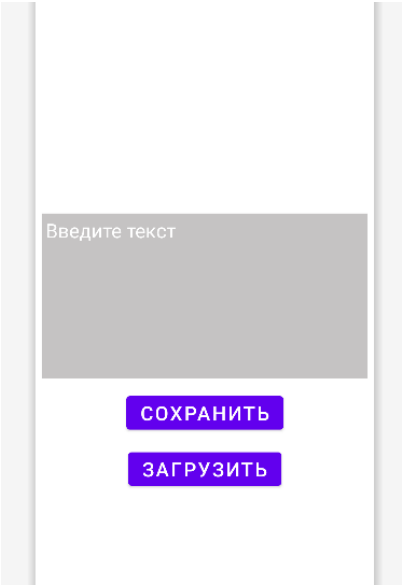

Рисунок 17

Перейдите в папку вашего приложения на локальном диске. И в каталоге ~\APP_NAME\app\src\main\ создайте папку assets. В этой папке создайте текстовый файл (имя файла – любое, в данном случае example.txt) и напишите в нем любой текст. Этот файл отобразится в вашем проекте:

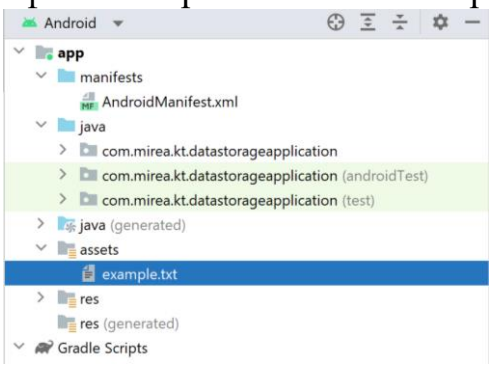

Рисунок 18

Чтобы получить доступ к файлам, хранящимся в assets, необходимо использовать специальный класс – **AssetManager**. Получить объект этого класса можно с помощью метода getAssets из любого Context-класса. Напишите код, который по нажатию кнопки «Загрузить» в DataActivity отобразит содержимое файла, хранящегося в assets в поле EditText.

```
public class DataActivity extends AppCompatActivity implements View.OnClickListener {

    private EditText et;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_data);
        et = findViewById(R.id.etValue);
        Button btnLoad = findViewById(R.id.btnLoad);
        btnLoad.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        if(v.getId() == R.id.btnLoad){
            AssetManager am = getApplicationContext().getAssets();
            InputStream is;
            try {
                is = am.open( fileName: "example.txt");
                ByteArrayOutputStream result = new ByteArrayOutputStream();
                byte[] buffer = new byte[1024];
                int length;
                while ((length = is.read(buffer)) != -1) {
                    result.write(buffer,  off: 0, length);
                }
                String data = result.toString();
                is.close();
                et.setText(data);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Рисунок 19

Измените текст в поле ввода. Как реализовать сохранение измененного значения, чтобы при следующем открытии приложения он сразу отобразился в EditText? Для этого можно воспользоваться простым хранилищем **Shared Preferences**. Данное хранилище используется для хранения простых значений типов: int, String, boolean, float, long, StringSet. Обычно Shared Preferences используется для быстрого сохранения и быстрой загрузки каких-либо параметров приложения, настроек.

Для получения объекта класса **SharedPreferences** необходимо вызвать метод getSharedPreferences любого объекта типа Context. Метод принимает на вход два параметра: имя файла и режим (значение типа int). Для современных приложений необходимо использовать приватный режим (константа MODE_PRIVATE, значение 0).

Хорошей практикой считается выделение работы с Shared Preferences в отдельный класс (как это было сделано с базами данных на предыдущих занятиях).

Создайте в приложении новый класс – PrefManager. Создайте в классе переменную типа **SharedPreferences** и конструктор, принимающий в качестве аргумента переменную типа Context. В этом же конструкторе можно реализовать инициализацию переменной типа **SharedPreferences**. Имя файла хранилища может быть любое. У приложения может быть не один такой файл.

```
public class PrefManager {

    private SharedPreferences pref;

    public PrefManager(Context context) {
        this.pref = context.getSharedPreferences( name: "my_app_prefs",Context.MODE_PRIVATE);
    }
}
```

Рисунок 20

В этом классе можно реализовывать методы по сохранению и загрузке переменных из хранилища SharedPreferences.

Напишите код для сохранения и загрузки текстового значения. Это текстовое значение будет сохранятся и загружаться по уникальному ключу, который задает разработчик. То есть, если потом необходимо сохранить уже какой-нибудь другой параметр (например, дату последнего старта приложения), необходимо использовать другой ключ.

```
public PrefManager(Context context) {
    this.pref = context.getSharedPreferences( name: "my_app_prefs",Context.MODE_PRIVATE);
}

public void saveDefaultTextValue(String text){
    SharedPreferences.Editor editor = this.pref.edit();
    editor.putString("default_text",text);
    editor.apply();
}

private String getDefaultTextValue(){
    return this.pref.getString( key: "default_text", defValue: "");
}
```

Рисунок 21

Создайте в классе DataActivity переменную типа PrefManager и в методе onCreate инициализируйте её.

```
public class DataActivity extends AppCompatActivity implements View.

    private EditText et;
    private PrefManager prefManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_data);
        et = findViewById(R.id.etValue);
        Button btnLoad = findViewById(R.id.btnLoad);
        Button btnSave = findViewById(R.id.btnSave);
        btnLoad.setOnClickListener(this);
        btnSave.setOnClickListener(this);
        prefManager = new PrefManager( context: this);
    }
```

Рисунок 22

Вызовите метод сохранения значения из поля ввода в хранилище Shared Preferences (старый код можно закомментировать):

```
@Override
public void onClick(View v) {
    if(v.getId() == R.id.btnSave){
        String text = et.getText().toString();
        if(!text.isEmpty()){
            prefManager.saveDefaultTextValue(text);
        }else{
            Toast.makeText( context: this, R.string.no_empty_field, Toast.LENGTH_LONG).show();
        }
    }
    ...
}
```

Рисунок 23

В методе onCreate реализуйте вызов метода загрузки текстового значения из хранилища. И поместите это значение в EditText. Тем самым поле ввода будет заполнено значением по умолчанию, которое было сохранено при последнем использовании приложения:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_data);
    et = findViewById(R.id.etValue);
    Button btnLoad = findViewById(R.id.btnLoad);
    Button btnSave = findViewById(R.id.btnSave);
    btnLoad.setOnClickListener(this);
    btnSave.setOnClickListener(this);
    prefManager = new PrefManager( context: this);
    et.setText(prefManager.getDefaultTextValue());
}
```

Рисунок 24

Запустите приложение, напишите что-нибудь в поле ввода и нажмите на кнопку «Сохранить». Закройте приложение и запустите его заново. Убедитесь, что значение из Shared Preferences корректно загрузилось.

Рисунок 25

Содержимое файла хранилища после сохранения значения:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="default_text">bla-bla-bla</string>
</map>
```

Рисунок 26