**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент департамента программной инженерии факультета компьютерных наук, кандидат техн. Наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Брейман  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. |  | | **ПРИЛОЖЕНИЕ МУЗЫКАЛЬНЫЙ ПЛЕЕР ДЛЯ ANDROID  Текст программы   ЛИСТ УТВЕРЖДЕНИЯ** **RU.17701729.04.01-01 12 01-1-ЛУ**  Исполнитель  Студент группы БПИ172  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/  О.В. Омельчишина /  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. | | аыувакауаааааа |

**УТВЕРЖДЕН**

**RU.17701729.04.01-01 12 01-1-ЛУ**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. |  | | **ПРИЛОЖЕНИЕ МУЗЫКАЛЬНЫЙ ПЛЕЕР ДЛЯ ANDROID**  **Текст программы  RU.17701729.04.01-01 12 01-1  Листов 38** |  |

Оглавление

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc8842299)

[1. MediaPlayerService.java 3](#_Toc8842300)

[2. CustomTouchListener.java 13](#_Toc8842301)

[3. MainActivity.java 14](#_Toc8842302)

[4. onItemClickListener.java 27](#_Toc8842303)

[5. PlaybackStatus.java 27](#_Toc8842304)

[6. RecyclerView\_Adapter.java 27](#_Toc8842305)

[7. SettingsBottomSheet.java 31](#_Toc8842306)

[8. StorageUtil.java 33](#_Toc8842307)

[9. Track.java 36](#_Toc8842308)

# 1. ТЕКСТ ПРОГРАММЫ

# MediaPlayerService.java

package com.olga\_o.course\_work.musicplayer;

//import android.app.Notification;

import android.app.Service;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.media.AudioManager;

import android.media.MediaPlayer;

import android.os.Binder;

import android.os.Handler;

import android.os.IBinder;

import android.telephony.PhoneStateListener;

import android.telephony.TelephonyManager;

import android.util.Log;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.Random;

import java.util.Set;

public class MediaPlayerService extends Service implements MediaPlayer.OnCompletionListener,

MediaPlayer.OnPreparedListener, MediaPlayer.OnErrorListener, MediaPlayer.OnSeekCompleteListener,

MediaPlayer.OnInfoListener, MediaPlayer.OnBufferingUpdateListener,

AudioManager.OnAudioFocusChangeListener {

public static final String ACTION\_PLAY = "ru.olga.ACTION\_PLAY";

public static final String ACTION\_PAUSE = "ru.olga.ACTION\_PAUSE";

public static final String ACTION\_PREVIOUS = "ru.olga.ACTION\_PREVIOUS";

public static final String ACTION\_NEXT = "ru.olga.ACTION\_NEXT";

public static final String ACTION\_STOP = "ru.olga..ACTION\_STOP";

private MediaPlayer mediaPlayer;

private boolean isLoopOrder;

public boolean isPlaying() {

return mediaPlayer.isPlaying();

}

public boolean isNull() {

return mediaPlayer == null;

}

private int resumePosition;

private AudioManager audioManager;

private final IBinder iBinder = new LocalBinder();

private ArrayList<Track> trackList;

private ArrayList<Integer> trackOrder;

// отслеживает что новая песня появилась

public interface new\_trackListener {

public void trackChangedEvent();

}

public static class TrackState {

private static boolean new\_track = false;

private static final ArrayList<new\_trackListener> nextTrackListeners = new ArrayList<>();

public static void setNextTrack() {

new\_track = !new\_track;

for (new\_trackListener ping : nextTrackListeners) {

ping.trackChangedEvent();

}

}

public void addRotationListener(new\_trackListener toadd) {

nextTrackListeners.add(toadd);

}

}

public Track getCurrentTrack() {

if (trackList.size() == 0)

return null;

return trackList.get(trackOrder.get(storage.getTrackIndex()));

}

public int getRealIndex()

{

return trackOrder.get(storage.getTrackIndex());

}

private StorageUtil storage;

private boolean ongoingCall = false;

private PhoneStateListener phoneStateListener;

private TelephonyManager telephonyManager;

@Override

public IBinder onBind(Intent intent) {

return iBinder;

}

@Override

public void onCreate() {

super.onCreate();

callStateListener();

registerBecomingNoisyReceiver();

register\_playNewAudio();

storage = new StorageUtil(getApplicationContext());

updateStorage(null);

switch (storage.getOrderSettings()) {

case 0: {

this.linearTrackOrder(storage.getTrackIndex());

break;

}

case 1: {

this.shuffleTrackOrder(storage.getTrackIndex());

break;

}

case 2: {

this.loopTrackOrder(storage.getTrackIndex());

break;

}

default: {

this.linearTrackOrder(storage.getTrackIndex());

storage.setOrderSettings(0);

break;

}

}

}

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

try {

updateStorage(null);

} catch (NullPointerException e) {

stopSelf();

}

//Request audio focus

if (requestAudioFocus() == false) {

stopSelf();

}

return super.onStartCommand(intent, flags, startId);

}

@Override

public boolean onUnbind(Intent intent) {

return super.onUnbind(intent);

}

@Override

public void onDestroy() {

super.onDestroy();

if (mediaPlayer != null) {

stopMedia();

mediaPlayer.release();

}

removeAudioFocus();

if (phoneStateListener != null) {

telephonyManager.listen(phoneStateListener, PhoneStateListener.LISTEN\_NONE);

}

unregisterReceiver(becomingNoisyReceiver);

unregisterReceiver(playNewAudio);

new StorageUtil(getApplicationContext()).clearCachedTrackPlaylist();

}

public class LocalBinder extends Binder {

public MediaPlayerService getService() {

return MediaPlayerService.this;

}

}

@Override

public void onBufferingUpdate(MediaPlayer mp, int percent) {

}

@Override

public void onCompletion(MediaPlayer mp) {

if (trackList.size() >= 1)

skipToNext();

Handler handler = new Handler();

handler.postDelayed(new Runnable() {

public void run() {

}

}, 1000);

Intent broadcastIntent = new Intent(MainActivity.Broadcast\_PLAY\_NEW\_AUDIO);

sendBroadcast(broadcastIntent);

}

public void playTrackWithSameIndexInUpdatedList() {

if (isLoopOrder)

return;

if (storage.getTrackIndex() == trackList.size()) {

storage.setTrackIndex(0);

}

if (mediaPlayer != null) {

stopMedia();

mediaPlayer.reset();

}

initMediaPlayer();

Handler handler = new Handler();

handler.postDelayed(new Runnable() {

public void run() {

// Actions to do after 10 seconds

}

}, 1000);

Intent broadcastIntent = new Intent(MainActivity.Broadcast\_PLAY\_NEW\_AUDIO);

sendBroadcast(broadcastIntent);

}

public void playNextMainActivity() {

skipToNext();

Handler handler = new Handler();

handler.postDelayed(new Runnable() {

public void run() {

// Actions to do after 10 seconds

}

}, 1000);

Intent broadcastIntent = new Intent(MainActivity.Broadcast\_PLAY\_NEW\_AUDIO);

sendBroadcast(broadcastIntent);

}

public void playPrevMainActivity() {

skipToPrevious();

Handler handler = new Handler();

handler.postDelayed(new Runnable() {

public void run() {

// Actions to do after 10 seconds

}

}, 1000);

Intent broadcastIntent = new Intent(MainActivity.Broadcast\_PLAY\_NEW\_AUDIO);

sendBroadcast(broadcastIntent);

}

@Override

public boolean onError(MediaPlayer mp, int what, int extra) {

switch (what) {

case MediaPlayer.MEDIA\_ERROR\_NOT\_VALID\_FOR\_PROGRESSIVE\_PLAYBACK:

Log.d("MediaPlayer Error", "MEDIA ERROR NOT VALID FOR PROGRESSIVE PLAYBACK " + extra);

break;

case MediaPlayer.MEDIA\_ERROR\_SERVER\_DIED:

Log.d("MediaPlayer Error", "MEDIA ERROR SERVER DIED " + extra);

break;

case MediaPlayer.MEDIA\_ERROR\_UNKNOWN:

Log.d("MediaPlayer Error", "MEDIA ERROR UNKNOWN " + extra);

break;

}

return false;

}

@Override

public boolean onInfo(MediaPlayer mp, int what, int extra) {

return false;

}

@Override

public void onPrepared(MediaPlayer mp) {

playMedia();

}

@Override

public void onSeekComplete(MediaPlayer mp) {

}

@Override

public void onAudioFocusChange(int focusState) {

switch (focusState) {

case AudioManager.AUDIOFOCUS\_GAIN:

// resume playback

if (mediaPlayer == null) initMediaPlayer();

else if (!mediaPlayer.isPlaying()) {

mediaPlayer.start();

TrackState.setNextTrack();

}

mediaPlayer.setVolume(1.0f, 1.0f);

break;

case AudioManager.AUDIOFOCUS\_LOSS:

if (mediaPlayer.isPlaying()) mediaPlayer.stop();

mediaPlayer.release();

mediaPlayer = null;

break;

case AudioManager.AUDIOFOCUS\_LOSS\_TRANSIENT:

if (mediaPlayer.isPlaying()) mediaPlayer.pause();

break;

case AudioManager.AUDIOFOCUS\_LOSS\_TRANSIENT\_CAN\_DUCK:

if (mediaPlayer.isPlaying()) mediaPlayer.setVolume(0.1f, 0.1f);

break;

}

}

private boolean requestAudioFocus() {

audioManager = (AudioManager) getSystemService(Context.AUDIO\_SERVICE);

int result = audioManager.requestAudioFocus(this, AudioManager.STREAM\_MUSIC, AudioManager.AUDIOFOCUS\_GAIN);

if (result == AudioManager.AUDIOFOCUS\_REQUEST\_GRANTED) {

//Focus gained

return true;

}

return false;

}

private boolean removeAudioFocus() {

return AudioManager.AUDIOFOCUS\_REQUEST\_GRANTED ==

audioManager.abandonAudioFocus(this);

}

private void initMediaPlayer() {

if (mediaPlayer == null) {

mediaPlayer = new MediaPlayer();

}

mediaPlayer.setOnCompletionListener(this);

mediaPlayer.setOnErrorListener(this);

mediaPlayer.setOnPreparedListener(this);

mediaPlayer.setOnBufferingUpdateListener(this);

mediaPlayer.setOnSeekCompleteListener(this);

mediaPlayer.setOnInfoListener(this);

mediaPlayer.reset();

mediaPlayer.setAudioStreamType(AudioManager.STREAM\_MUSIC);

try {

if ((new File(trackList.get(trackOrder.get(storage.getTrackIndex())).getPath())).exists())

mediaPlayer.setDataSource(trackList.get(trackOrder.get(storage.getTrackIndex())).getPath());

else {

playNextMainActivity();

return;

}

} catch (IOException e) {

e.printStackTrace();

playNextMainActivity();

return;

}

mediaPlayer.prepareAsync();

}

public void playMedia() {

if (!mediaPlayer.isPlaying()) {

mediaPlayer.start();

TrackState.setNextTrack();

}

}

public void stopMedia() {

if (mediaPlayer == null) return;

if (mediaPlayer.isPlaying()) {

mediaPlayer.stop();

resumePosition = 0;

}

}

public void pauseMedia() {

if (mediaPlayer.isPlaying()) {

mediaPlayer.pause();

resumePosition = mediaPlayer.getCurrentPosition();

}

}

public void resumeMedia() {

if (!mediaPlayer.isPlaying()) {

mediaPlayer.seekTo(resumePosition);

mediaPlayer.start();

TrackState.setNextTrack();

}

}

public void stop() {

if (mediaPlayer.isPlaying())

mediaPlayer.stop();

mediaPlayer.release();

mediaPlayer = null;

}

private void skipToNext() {

if (isLoopOrder)

return;

if (storage.getTrackIndex() == trackList.size() - 1) {

storage.setTrackIndex(0);

} else {

storage.setTrackIndex(storage.getTrackIndex() + 1);

}

if (mediaPlayer != null) {

stopMedia();

mediaPlayer.reset();

}

initMediaPlayer();

}

private void skipToPrevious() {

if (isLoopOrder)

return;

if (storage.getTrackIndex() == 0) {

storage.setTrackIndex(trackList.size() - 1);

} else {

storage.setTrackIndex(storage.getTrackIndex() - 1);

}

if (mediaPlayer != null) {

stopMedia();

mediaPlayer.reset();

}

initMediaPlayer();

}

/\*

TRACK ORDER

в очереди номер песни в дефолтном листе

0 1

1 3

2 5

3 0

4 2

5 4

\*/

public void linearTrackOrder(int start\_pos) {

trackOrder = new ArrayList<>(trackList.size());

for (int i = 0; i < trackList.size(); ++i)

trackOrder.add((i + start\_pos) % trackList.size());

isLoopOrder = false;

}

public void shuffleTrackOrder(int start\_pos) {

isLoopOrder = false;

trackOrder = new ArrayList<>(trackList.size());

Random rd = new Random();

Set<Integer> used = new HashSet<>();

used.add(-1);

trackOrder.add(start\_pos);

used.add(start\_pos);

for (int i = 1; i < trackList.size(); ++i) {

int next\_rd = -1;

while (used.contains(next\_rd)) {

next\_rd = rd.nextInt(trackList.size());

}

used.add(next\_rd);

trackOrder.add(next\_rd);

}

int i = 0;

i += 1;

}

public void loopTrackOrder(int index) {

trackOrder = new ArrayList<>();

for (int i = 0; i < trackList.size(); ++i)

trackOrder.add(index);

isLoopOrder = true;

}

public void updateOrder(int start\_index) {

if (start\_index == -1) // значит поменять порядок но первая песня = текущая песня

start\_index = trackOrder.get(storage.getTrackIndex());

switch (storage.getOrderSettings()) {

case 0: {

linearTrackOrder(start\_index);

break;

}

case 1: {

shuffleTrackOrder(start\_index);

break;

}

case 2: {

loopTrackOrder(start\_index);

break;

}

default: {

linearTrackOrder(start\_index);

break;

}

}

}

public void updateStorage(RecyclerView\_Adapter adapter) {

if (adapter == null)

trackList = storage.getTrackList();

else

trackList = adapter.getTrackList();

}

private BroadcastReceiver becomingNoisyReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

pauseMedia();

}

};

private void registerBecomingNoisyReceiver() {

//register after getting audio focus

IntentFilter intentFilter = new IntentFilter(AudioManager.ACTION\_AUDIO\_BECOMING\_NOISY);

registerReceiver(becomingNoisyReceiver, intentFilter);

}

private void callStateListener() {

// Get the telephony manager

telephonyManager = (TelephonyManager) getSystemService(Context.TELEPHONY\_SERVICE);

//Starting listening for PhoneState changes

phoneStateListener = new PhoneStateListener() {

@Override

public void onCallStateChanged(int state, String incomingNumber) {

switch (state) {

//if at least one call exists or the phone is ringing

//pause the MediaPlayer

case TelephonyManager.CALL\_STATE\_OFFHOOK:

case TelephonyManager.CALL\_STATE\_RINGING:

if (mediaPlayer != null) {

pauseMedia();

ongoingCall = true;

}

break;

case TelephonyManager.CALL\_STATE\_IDLE:

// Phone idle. Start playing.

if (mediaPlayer != null) {

if (ongoingCall) {

ongoingCall = false;

resumeMedia();

}

}

break;

}

}

};

telephonyManager.listen(phoneStateListener,

PhoneStateListener.LISTEN\_CALL\_STATE);

}

private BroadcastReceiver playNewAudio = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

int temp\_index = storage.getTrackIndex();

if (temp\_index != -1 && temp\_index < trackList.size()) {

} else {

stopSelf();

}

if (requestAudioFocus() == false) {

//Could not gain focus

stopSelf();

}

if (mediaPlayer == null)

initMediaPlayer();

stopMedia();

mediaPlayer.reset();

initMediaPlayer();

storage.setRealIndex(getRealIndex());

}

};

private void register\_playNewAudio() {

//Register playNewMedia receiver

IntentFilter filter = new IntentFilter(MainActivity.Broadcast\_PLAY\_NEW\_AUDIO);

registerReceiver(playNewAudio, filter);

}

}

# CustomTouchListener.java

package com.olga\_o.course\_work.musicplayer;

import android.content.Context;

import android.support.v7.widget.RecyclerView;

import android.view.GestureDetector;

import android.view.MotionEvent;

import android.view.View;

public class CustomTouchListener implements RecyclerView.OnItemTouchListener {

//Gesture detector to intercept the touch events

GestureDetector gestureDetector;

private onItemClickListener clickListener;

public CustomTouchListener(Context context, final onItemClickListener clickListener) {

this.clickListener = clickListener;

gestureDetector = new GestureDetector(context, new GestureDetector.SimpleOnGestureListener() {

@Override

public boolean onSingleTapUp(MotionEvent e) {

return true;

}

});

}

@Override

public boolean onInterceptTouchEvent(RecyclerView recyclerView, MotionEvent e) {

View child = recyclerView.findChildViewUnder(e.getX(), e.getY());

if (child != null && clickListener != null && gestureDetector.onTouchEvent(e)) {

clickListener.onClick(child, recyclerView.getChildLayoutPosition(child));

}

return false;

}

@Override

public void onTouchEvent(RecyclerView rv, MotionEvent e) {

}

@Override

public void onRequestDisallowInterceptTouchEvent(boolean disallowIntercept) {

}

}

# MainActivity.java

package com.olga\_o.course\_work.musicplayer;

import android.Manifest;

import android.content.ComponentName;

import android.content.ContentResolver;

import android.content.Context;

import android.content.Intent;

import android.content.ServiceConnection;

import android.content.pm.PackageManager;

import android.database.Cursor;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

import android.graphics.Color;

import android.net.Uri;

import android.os.Bundle;

import android.os.IBinder;

import android.os.PersistableBundle;

import android.provider.MediaStore;

import android.support.design.widget.BottomSheetBehavior;

import android.support.v4.widget.DrawerLayout;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.support.v7.widget.SearchView;

import android.view.MenuInflater;

import android.view.View;

import android.view.Menu;

import android.view.MenuItem;

import android.view.inputmethod.EditorInfo;

import android.widget.ImageView;

//import android.widget.Toolbar;

import android.support.v7.widget.Toolbar;

import android.widget.LinearLayout;

import android.widget.TextView;

import java.io.File;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity implements SettingsBottomSheet.BottomSheetListener {

public static final String Broadcast\_PLAY\_NEW\_AUDIO = "ru.olga.PlayNewAudio";

private static final String LOG\_TAG = "log\_tag";

private MediaPlayerService player;

boolean serviceBound = false;

ArrayList<Track> defaultTrackList;

StorageUtil storage;

ImageView fav;

boolean sort\_less\_to\_bigger = true;

private ArrayList<Track> favTrackList;

boolean favPlaying;

RecyclerView\_Adapter adapter;

RecyclerView\_Adapter adapter\_fav;

DrawerLayout right\_side\_drawer;

ImageView play\_pause;

private Bitmap defaultCover;

public void updateCurrentTrackView() {

View current\_track\_view = findViewById(R.id.current\_track);

Track current\_track = player.getCurrentTrack();

ImageView album\_cover = (ImageView) current\_track\_view.findViewById(R.id.track\_logo);

TextView artist = (TextView) current\_track\_view.findViewById(R.id.track\_artist);

TextView title = (TextView) current\_track\_view.findViewById(R.id.track\_title);

artist.setText(current\_track.getArtist());

title.setText(current\_track.getTitle());

for (Track track : favTrackList)

if (track.getPath().equals(current\_track.getPath())) // если такая песня уже есть в списке любимых убираем ее отуда и сменяем сердечко

{

fav.setImageResource(R.drawable.ic\_favorite\_true);

return;

}

fav.setImageResource(R.drawable.ic\_favorite\_false);

Bitmap track\_cover = current\_track.getCoverImage(this.getContentResolver());

if (track\_cover != null)

album\_cover.setImageBitmap(Bitmap.createScaledBitmap(track\_cover, 360, 360, true));

else

album\_cover.setImageBitmap(Bitmap.createScaledBitmap(defaultCover, 360, 360, true));

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

storage = new StorageUtil(getApplicationContext());

String[] permissions = {android.Manifest.permission.READ\_EXTERNAL\_STORAGE, Manifest.permission.READ\_PHONE\_STATE};

this.requestPermissions(permissions, LOCATION\_PERMISSION);

}

int LOCATION\_PERMISSION = 2;

@Override

public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {

if (requestCode == LOCATION\_PERMISSION && grantResults[0] == PackageManager.PERMISSION\_GRANTED)

start\_app();

else {

this.requestPermissions(permissions, LOCATION\_PERMISSION);

}

}

private void start\_app() {

setContentView(R.layout.activity\_main);

loadTracks();

if (defaultTrackList.size() == 0)

stop();

loadFavTracks();

bounService();

initFavTracksRecycleView();

initRecyclerView();

initButtons(); // так же подгружаются старые настройки

setSupportActionBar((Toolbar) findViewById(R.id.toolbar));

getSupportActionBar().setDisplayShowTitleEnabled(false);

setFavListBottomSheet();

setTrackChangedListener();

right\_side\_drawer = (DrawerLayout) findViewById(R.id.drawer\_layout);

right\_side\_drawer.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_LOCKED\_CLOSED);

right\_side\_drawer.setScrimColor(Color.TRANSPARENT);

defaultCover = BitmapFactory.decodeResource(getResources(), R.drawable.default\_cover\_image);

}

private void stop() {

android.os.Process.killProcess(android.os.Process.myPid());

System.exit(1);

}

private ServiceConnection serviceConnection = new ServiceConnection() {

@Override

public void onServiceConnected(ComponentName name, IBinder service) {

MediaPlayerService.LocalBinder binder = (MediaPlayerService.LocalBinder) service;

player = binder.getService();

storage.setTrackIndex(0);

player.updateOrder(storage.getRealIndex());

serviceBound = true;

}

@Override

public void onServiceDisconnected(ComponentName name) {

serviceBound = false;

}

};

private void bounService() {

Intent playerIntent = new Intent(this, MediaPlayerService.class);

startService(playerIntent);

bindService(playerIntent, serviceConnection, Context.BIND\_AUTO\_CREATE);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.menu\_and\_search, menu);

MenuItem searchItem = menu.findItem(R.id.search\_view\_track);

SearchView searchView = (SearchView) searchItem.getActionView();

searchView.setImeOptions(EditorInfo.IME\_ACTION\_DONE);

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

@Override

public boolean onQueryTextSubmit(String query) {

return false;

}

@Override

public boolean onQueryTextChange(String newText) {

adapter.getFilter().filter(newText);

return false;

}

});

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.settings:

SettingsBottomSheet bottomSheet = new SettingsBottomSheet();

bottomSheet.setAdapter(adapter);

bottomSheet.show(getSupportFragmentManager(), "exampleBottomSheet");

return true;

default:

return super.onOptionsItemSelected(item);

}

}

@Override

public void onSaveInstanceState(Bundle outState, PersistableBundle outPersistentState) {

super.onSaveInstanceState(outState, outPersistentState);

outState.putBoolean("serviceStatus", serviceBound);

}

@Override

protected void onRestoreInstanceState(Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

serviceBound = savedInstanceState.getBoolean("serviceStatus");

}

@Override

protected void onResume()

{

super.onResume();

if (player != null && !player.isNull())

if (player.isPlaying())

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_pause));

else

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_play));

}

private void loadFavTracks() {

if (storage.getFavTrackList() != null) {

favTrackList = storage.getFavTrackList();

for (Track track : favTrackList)

if (!(new File(track.getPath())).exists())

favTrackList.remove(track);

} else

favTrackList = new ArrayList<>();

}

private void loadTracks() {

ContentResolver contentResolver = getContentResolver();

Uri uri = MediaStore.Audio.Media.EXTERNAL\_CONTENT\_URI;

String selection = MediaStore.Audio.Media.IS\_MUSIC + "!= 0";

String sortOrder = MediaStore.Audio.Media.TITLE + " ASC";

Cursor cursor = contentResolver.query(uri, null, selection, null, sortOrder);

defaultTrackList = new ArrayList<>();

if (cursor != null && cursor.getCount() > 0) {

while (cursor.moveToNext()) {

String path = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DATA));

String title = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.TITLE));

String album = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM));

String artist = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST));

String duration = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DURATION));

String file\_name = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DISPLAY\_NAME));

String creationDate = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DATE\_ADDED));

Long albumId = cursor.getLong(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.ALBUM\_ID));

defaultTrackList.add(new Track(path, title, album, artist, duration, file\_name, creationDate, albumId));

}

}

cursor.close();

}

private void updateFavList() {

adapter\_fav.trackList = favTrackList;

adapter\_fav.notifyDataSetChanged();

storage.setFavTrackList(favTrackList);

}

private void initRecyclerView() {

if (defaultTrackList.size() > 0) {

RecyclerView recyclerView = (RecyclerView) findViewById(R.id.track\_RecyclerView);

adapter = new RecyclerView\_Adapter(defaultTrackList, getApplication());

recyclerView.setAdapter(adapter);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

recyclerView.addOnItemTouchListener(new CustomTouchListener(this, new onItemClickListener() {

// При нажатии на элемент списка устанавливается текущий порядок и размер проигрывания

@Override

public void onClick(View view, int index) {

favPlaying = false;

storage.setTrackIndex(0);

player.updateStorage(adapter);

player.updateOrder(index);

playAudio();

}

}));

}

}

private void initFavTracksRecycleView() {

android.support.v7.widget.RecyclerView fav\_RecyclerView = (android.support.v7.widget.RecyclerView) findViewById(R.id.fav\_track\_RecyclerView);

adapter\_fav = new RecyclerView\_Adapter(favTrackList, getApplication());

fav\_RecyclerView.setAdapter(adapter\_fav);

fav\_RecyclerView.setLayoutManager(new LinearLayoutManager(this));

fav\_RecyclerView.addOnItemTouchListener(new CustomTouchListener(this, new onItemClickListener() {

// При нажатии на элемент списка устанавливается текущий порядок и размер проигрывания

@Override

public void onClick(View view, int index) {

favPlaying = true;

storage.setTrackIndex(0);

player.updateStorage(adapter\_fav);

player.updateOrder(index);

playAudio(); // index - порядок в текущем листе и 0

}

}));

}

BottomSheetBehavior bottomSheetBehavior;

private void setFavListBottomSheet() {

// получение вью нижнего экрана

LinearLayout llBottomSheet = (LinearLayout) findViewById(R.id.include\_fav);

bottomSheetBehavior = BottomSheetBehavior.from(llBottomSheet);

bottomSheetBehavior.setHideable(false);

}

private void initButtons() {

/\*

FAV Button

\*/

fav = (ImageView) findViewById(R.id.do\_fav\_current\_track);

fav.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Track current\_track = player.getCurrentTrack();

for (Track track : favTrackList)

if (track.getPath().equals(current\_track.getPath())) // если такая песня уже есть в списке любимых убираем ее отуда и сменяем сердечко

{

favTrackList.remove(track);

updateFavList();

if (favPlaying) {

player.stop();

if (favTrackList.size() == 0) { // если это был последний трек меняем источник воспроизведения на текущий отображаемый обычный список

player.updateStorage(adapter);

storage.setTrackIndex(0);

player.updateOrder(0);

updateCurrentTrackView();

favPlaying = false;

} else {

player.playTrackWithSameIndexInUpdatedList();

updateCurrentTrackView();

player.updateStorage(adapter\_fav);

}

}

fav.setImageResource(R.drawable.ic\_favorite\_false);

//todo сменить сердечко

return;

}

for (Track track : defaultTrackList)

if (track.getPath().equals(current\_track.getPath())) {

favTrackList.add(track);

fav.setImageResource(R.drawable.ic\_favorite\_true);

updateFavList();

return;

}

}

});

/\*

sort\_less\_to\_bigger swap

\*/

final ImageView sort\_less\_to\_bigger\_button = (ImageView) findViewById(R.id.less\_to\_bigger);

sort\_less\_to\_bigger\_button.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

sort\_less\_to\_bigger = !sort\_less\_to\_bigger;

adapter.setSort(storage.getOrderSettings(), sort\_less\_to\_bigger);

storage.setTrackList(adapter.getTrackList());

}

});

/\*

PLAY PAUSE

\*/

//меняется с play/pause

play\_pause = (ImageView) findViewById(R.id.play\_pause);

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_play));

play\_pause.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (player == null) {

bounService();

return;

}

if (player.isNull()) {

player.playTrackWithSameIndexInUpdatedList();

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_pause)); return;

}

if (player.isPlaying()) {

player.pauseMedia();

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_play));

} else {

player.resumeMedia();

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_pause));

}

}

});

/\*

STOP BUTTON

final Button stop = (Button) findViewById(R.id.stop);

stop.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (player == null) {

playAudio();

return;

}

player.stopMedia();

}

});

\*/

/\*

play next

\*/

final ImageView play\_next = (ImageView) findViewById(R.id.skip\_to\_next);

play\_next.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (player == null) {

bounService();

return;

}

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_pause));

player.playNextMainActivity();

}

});

/\*

play prev

\*/

final ImageView play\_prev = (ImageView) findViewById(R.id.skip\_to\_previous);

play\_prev.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (player == null) {

bounService();

return;

}

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_pause));

player.playPrevMainActivity();

}

});

/\*

ORDER

\*/

final ImageView fab\_order = (ImageView) findViewById(R.id.order);

//set text from cashed settings

int old\_order = storage.getOrderSettings();

switch (old\_order) {

case 0: {

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_linear));

break;

}

case 1: {

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_shuffle));

break;

}

case 2: {

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_loop));

break;

}

default: {

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_linear));

storage.setOrderSettings(0);

break;

}

} // setOldTextSettings

fab\_order.setOnClickListener(new View.OnClickListener() {

int next\_param = storage.getOrderSettings() + 1;

// 0 - linear

// 1- shuffle

// 2 - loop

@Override

public void onClick(View view) {

switch (next\_param) {

case 0: {

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_linear));

storage.setOrderSettings(0);

break;

}

case 1: {

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_shuffle));

storage.setOrderSettings(1);

break;

//next\_param = -1;

}

case 2: {

next\_param = -1;

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_loop));

storage.setOrderSettings(2);

break;

}

default: {

fab\_order.setImageDrawable(getDrawable(R.drawable.ic\_linear));

storage.setOrderSettings(0);

break;

}

}

storage.setTrackIndex(0);

++next\_param;

player.updateOrder(-1); // все равно внутри надо проверить что порядок тот

}

});

/\*

SORT by

\*/

/\*

Sort

0 - by file name

1 - by track title

2 - by artist

3 - by date add

\*/

final ImageView botton\_sort = (ImageView) findViewById(R.id.sort\_by);

int old\_sort = storage.loadSortSettings();

switch (old\_sort) {

case 0: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.file\_sort));

adapter.setSort(0, sort\_less\_to\_bigger);

storage.setTrackList(adapter.getTrackList());

break;

}

case 1: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.track\_sort));

adapter.setSort(1, sort\_less\_to\_bigger);

storage.setTrackList(adapter.getTrackList());

break;

}

case 2: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.artist\_sort));

adapter.setSort(2, sort\_less\_to\_bigger);

storage.setTrackList(adapter.getTrackList());

break;

}

case 3: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.date\_sort));

adapter.setSort(3, sort\_less\_to\_bigger);

storage.setTrackList(adapter.getTrackList());

break;

}

default: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.file\_sort));

adapter.setSort(0, sort\_less\_to\_bigger);

storage.storeSortSettings(0);

storage.setTrackList(adapter.getTrackList());

break;

}

}

// setOldTextSettings

botton\_sort.setOnClickListener(new View.OnClickListener() {

int next\_param = storage.loadSortSettings() + 1;

/\*

Sort

0 - by file name

1 - by track title

2 - by artist

3 - by date add

\*/

@Override

public void onClick(View view) {

switch (next\_param) {

case 0: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.file\_sort));

storage.storeSortSettings(0);

adapter.setSort(0, sort\_less\_to\_bigger);

break;

}

case 1: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.track\_sort));

storage.storeSortSettings(1);

adapter.setSort(1, sort\_less\_to\_bigger);

break;

}

case 2: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.artist\_sort));

storage.storeSortSettings(2);

adapter.setSort(2, sort\_less\_to\_bigger);

break;

}

case 3: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.date\_sort));

storage.storeSortSettings(3);

adapter.setSort(3, sort\_less\_to\_bigger);

next\_param = -1;

break;

}

default: {

botton\_sort.setImageDrawable(getDrawable(R.drawable.file\_sort));

storage.storeSortSettings(0);

adapter.setSort(0, sort\_less\_to\_bigger);

break;

}

}

++next\_param;

storage.setTrackList(adapter.getTrackList());

//player.updateOrder();

}

});

// drag\_fav click

final ImageView drag\_fav = (ImageView) findViewById(R.id.drag\_fav\_list);

drag\_fav.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if (bottomSheetBehavior.getState() == BottomSheetBehavior.STATE\_EXPANDED)

bottomSheetBehavior.setState(BottomSheetBehavior.STATE\_COLLAPSED);

else

bottomSheetBehavior.setState(BottomSheetBehavior.STATE\_EXPANDED);

}

});

}

private void setTrackChangedListener() {

MediaPlayerService.TrackState ts = new MediaPlayerService.TrackState();

MediaPlayerService.new\_trackListener new\_trackListener = new MediaPlayerService.new\_trackListener() {

@Override

public void trackChangedEvent() {

updateCurrentTrackView();

right\_side\_drawer.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_UNLOCKED);

}

};

ts.addRotationListener(new\_trackListener);

}

private void playAudio() {

play\_pause.setImageDrawable(getDrawable(R.drawable.image\_pause));

Intent broadcastIntent = new Intent(Broadcast\_PLAY\_NEW\_AUDIO);

sendBroadcast(broadcastIntent);

right\_side\_drawer.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_UNLOCKED);

}

@Override

protected void onDestroy() {

super.onDestroy();

if (serviceBound) {

unbindService(serviceConnection);

player.stopSelf();

}

storage.setRealIndex(player.getRealIndex());

storage.setTrackIndex(0);

}

@Override

public void onButtonClicked(String text) {

}

}

# onItemClickListener.java

package com.olga\_o.course\_work.musicplayer;

import android.view.View;

public interface onItemClickListener {

public void onClick(View view, int index);

}

# PlaybackStatus.java

package com.olga\_o.course\_work.musicplayer;

public enum PlaybackStatus {

PLAYING,

PAUSED

}

# RecyclerView\_Adapter.java

package com.olga\_o.course\_work.musicplayer;

import android.content.Context;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Filter;

import android.widget.Filterable;

import android.widget.TextView;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

public class RecyclerView\_Adapter extends RecyclerView.Adapter<ViewHolder> implements Filterable {

ArrayList<Track> trackList;

ArrayList<Track> fullTrackList;

Context context;

public ArrayList<Track> getTrackList() {

return trackList;

}

public RecyclerView\_Adapter(ArrayList<Track> list, Context context) {

this.fullTrackList = list;

this.trackList = new ArrayList<>(list);

this.context = context;

}

@Override

public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

//Inflate the layout, initialize the View Holder

View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.item\_layout, parent, false);

ViewHolder holder = new ViewHolder(v);

return holder;

}

@Override

public void onBindViewHolder(ViewHolder holder, int position) {

//Use the provided View Holder on the onCreateViewHolder method to populate the current row on the RecyclerView

holder.title.setText(trackList.get(position).getTitle());

holder.artist.setText(trackList.get(position).getArtist());

}

@Override

public int getItemCount() {

//returns the number of elements the RecyclerView will display

return trackList.size();

}

@Override

public void onAttachedToRecyclerView(RecyclerView recyclerView) {

super.onAttachedToRecyclerView(recyclerView);

}

@Override

public Filter getFilter() {

return filter;

}

public boolean file\_name = true;

public boolean title = true;

public boolean artist = true;

public boolean album = true;

private boolean filterOptions(Track track, String filterPattern) {

boolean return\_value = false;

if (file\_name)

return\_value = return\_value || track.getFile\_name().toLowerCase().contains(filterPattern.toLowerCase());

if (title)

return\_value = return\_value || track.getTitle().toLowerCase().contains(filterPattern.toLowerCase());

if (artist)

return\_value = return\_value || track.getArtist().toLowerCase().contains(filterPattern.toLowerCase());

if (album)

return\_value = return\_value || track.getAlbum().toLowerCase().contains(filterPattern.toLowerCase());

return return\_value;

}

private Filter filter = new Filter() {

@Override

protected FilterResults performFiltering(CharSequence constraint) {

List<Track> newList = new ArrayList<>();

if (constraint == null || constraint.length() == 0) {

newList.addAll(fullTrackList);

} else {

String filterPattern = constraint.toString().toLowerCase().trim();

for (Track item : fullTrackList) {

if (filterOptions(item, filterPattern)) {

newList.add(item);

}

}

}

FilterResults results = new FilterResults();

results.values = newList;

return results;

}

@Override

protected void publishResults(CharSequence constraint, FilterResults results) {

trackList.clear();

trackList.addAll((List) results.values);

notifyDataSetChanged();

}

};

/\*

Sort

0 - by file name

1 - by track title

2 - by artist

3 - by date add

\*/

public void setSort(final int sort\_type, final boolean less\_to\_bigger) {

trackList.sort(new Comparator<Track>() {

@Override

public int compare(Track t1, Track t2) {

switch (sort\_type) {

case 0: {

if (less\_to\_bigger)

return t1.getFile\_name().compareTo(t2.getFile\_name());

return -t1.getFile\_name().compareTo(t2.getFile\_name());

}

case 1: {

if (less\_to\_bigger)

return t1.getTitle().compareTo(t2.getTitle());

return -t1.getTitle().compareTo(t2.getTitle());

}

case 2: {

if (less\_to\_bigger)

return t1.getArtist().compareTo(t2.getArtist());

return -t1.getArtist().compareTo(t2.getArtist());

}

case 3: {

if (less\_to\_bigger)

return t1.getCreationDate() - t2.getCreationDate();

return -(t1.getCreationDate() - t2.getCreationDate());

}

default: {

return t1.getTitle().compareTo(t2.getTitle());

}

}

}

});

this.fullTrackList = new ArrayList<>(trackList);

notifyDataSetChanged();

}

}

class ViewHolder extends RecyclerView.ViewHolder implements View.OnLongClickListener {

TextView title;

TextView artist;

ViewHolder(View itemView) {

super(itemView);

title = (TextView) itemView.findViewById(R.id.item\_title);

artist = (TextView) itemView.findViewById(R.id.item\_artist);

}

@Override

public boolean onLongClick(View view) {

// Handle long click

// Return true to indicate the click was handled

return true;

}

}

# SettingsBottomSheet.java

package com.olga\_o.course\_work.musicplayer;

import android.content.Context;

import android.os.Bundle;

import android.support.annotation.Nullable;

import android.support.design.widget.BottomSheetDialogFragment;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.RadioButton;

import android.widget.RadioGroup;

public class SettingsBottomSheet extends BottomSheetDialogFragment {

private BottomSheetListener mListener;

RadioGroup radioGroup;

RecyclerView\_Adapter adapter;

public void setAdapter(RecyclerView\_Adapter adapter) {

this.adapter = adapter;

}

@Nullable

@Override

public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

View v = inflater.inflate(R.layout.settings\_bottom\_sheet, container, false);

RadioButton file\_nameRadioButton = (RadioButton) v.findViewById(R.id.radioButtonFileName);

file\_nameRadioButton.setOnClickListener(radioButtonClickListener);

file\_nameRadioButton.setChecked(adapter.file\_name);

RadioButton titleRadioButton = (RadioButton) v.findViewById(R.id.radioButtonTitle);

titleRadioButton.setOnClickListener(radioButtonClickListener);

titleRadioButton.setChecked(adapter.title);

RadioButton artistRadioButton = (RadioButton) v.findViewById(R.id.radioButtonArtist);

artistRadioButton.setOnClickListener(radioButtonClickListener);

artistRadioButton.setChecked(adapter.artist);

RadioButton albumRadioButton = (RadioButton) v.findViewById(R.id.radioButtonAlbum);

albumRadioButton.setOnClickListener(radioButtonClickListener);

albumRadioButton.setChecked(adapter.album);

Button button1 = v.findViewById(R.id.accept\_settings);

button1.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

mListener.onButtonClicked("Button 1 clicked");

dismiss();

}

});

return v;

}

public interface BottomSheetListener {

void onButtonClicked(String text);

}

@Override

public void onAttach(Context context) {

super.onAttach(context);

try {

mListener = (BottomSheetListener) context;

} catch (ClassCastException e) {

throw new ClassCastException(context.toString()

+ " must implement BottomSheetListener");

}

}

View.OnClickListener radioButtonClickListener = new View.OnClickListener() {

@Override

public void onClick(View v) {

RadioButton rb = (RadioButton) v;

switch (rb.getId()) {

case R.id.radioButtonFileName:

adapter.file\_name = !adapter.file\_name;

rb.setChecked(adapter.file\_name);

break;

case R.id.radioButtonTitle:

adapter.title = !adapter.title;

rb.setChecked(adapter.title);

break;

case R.id.radioButtonArtist:

adapter.artist = !adapter.artist;

rb.setChecked(adapter.artist);

break;

case R.id.radioButtonAlbum:

adapter.album = !adapter.album;

rb.setChecked(adapter.album);

break;

default:

break;

}

}

};

}

# StorageUtil.java

package com.olga\_o.course\_work.musicplayer;

import android.content.Context;

import android.content.SharedPreferences;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import java.util.ArrayList;

import java.lang.reflect.Type;

public class StorageUtil {

private final String STORAGE = "ru.olga.player.STORAGE";

private SharedPreferences preferences;

private Context context;

public StorageUtil(Context context) {

this.context = context;

}

public void setTrackList(ArrayList<Track> arrayList) {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

Gson gson = new Gson();

String json = gson.toJson(arrayList);

editor.putString("trackArrayList", json);

editor.apply();

}

public ArrayList<Track> getTrackList() {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

Gson gson = new Gson();

String json = preferences.getString("trackArrayList", null);

Type type = new TypeToken<ArrayList<Track>>() {

}.getType();

return gson.fromJson(json, type);

}

public void setTrackIndex(int index) {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

editor.putInt("audioOrderIndex", index);

editor.apply();

}

public int getTrackIndex() {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

return preferences.getInt("audioOrderIndex", 0);//return -1 if no data found

}

public void setLastSongPath(String path) {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

editor.putString("last\_song\_path", path);

editor.apply();

}

public String getLastSongPath()

{

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

return preferences.getString("last\_song\_path", null);

}

public void clearCachedTrackPlaylist() {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

editor.clear();

editor.commit();

}

/\*

order

0-linear

1-shuffle

2-loop

\*/

public void setOrderSettings(int order) {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

editor.putInt("orderSettings", order);

editor.apply();

}

public int getOrderSettings() {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

return preferences.getInt("orderSettings", -1);//return -1 if no data found

}

/\*

Sort

0 - alphabet

1 - by track title

2 - by artist

3 - by date add

\*/

public void storeSortSettings(int order) {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

editor.putInt("SortSettings", order);

editor.apply();

}

public int loadSortSettings() {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

return preferences.getInt("SortSettings", -1);//return -1 if no data found

}

public void setFavTrackList(ArrayList<Track> arrayList) {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = preferences.edit();

Gson gson = new Gson();

String json = gson.toJson(arrayList);

editor.putString("favTrackArrayList", json);

editor.apply();

}

public ArrayList<Track> getFavTrackList() {

preferences = context.getSharedPreferences(STORAGE, Context.MODE\_PRIVATE);

Gson gson = new Gson();

String json = preferences.getString("favTrackArrayList", null);

Type type = new TypeToken<ArrayList<Track>>() {

}.getType();

return gson.fromJson(json, type);

}

}

# Track.java

package com.olga\_o.course\_work.musicplayer;

import android.content.ContentResolver;

import android.content.ContentUris;

import android.graphics.Bitmap;

import android.net.Uri;

import android.provider.MediaStore;

import java.io.IOException;

import java.io.Serializable;

public class Track implements Serializable {

private String path;

private String title;

private String album;

private String artist;

private int duration;

private String file\_name;

private int creationDate;

private Long albumIdcover;

public Track(String path, String title, String album, String artist, String duration, String file\_name, String creationDate, Long albumId) {

this.path = path;

this.title = title;

this.album = album;

this.artist = artist;

this.duration = Integer.parseInt(duration);

this.file\_name = file\_name;

this.creationDate = Integer.parseInt(creationDate);

this.albumIdcover = albumId;

}

public String getPath() {

return path;

}

public void setData(String path) {

this.path = path;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getAlbum() {

return album;

}

public void setAlbum(String album) {

this.album = album;

}

public String getArtist() {

return artist;

}

public void setArtist(String artist) {

this.artist = artist;

}

public String getFile\_name() {

return file\_name;

}

public void setFile\_name(String file\_name) {

this.file\_name = file\_name;

}

public int getCreationDate() {

return creationDate;

}

public void setCreationDate(int creationDate) {

this.creationDate = creationDate;

}

public Bitmap getCoverImage(ContentResolver cr) {

try {

Uri sArtworkUri = Uri.parse("content://media/external/audio/albumart");

Uri cover = ContentUris.withAppendedId(sArtworkUri, albumIdcover);

return MediaStore.Images.Media.getBitmap(cr, cover);

} catch (IOException e) {

e.printStackTrace();

return null;

}

}

}

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| изменённых | заменённых | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |