Aula 08 – Armazenamento Persistente de Dados

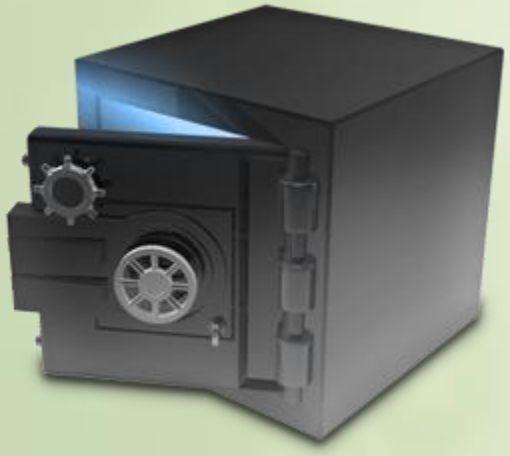
Programação em Java para a Plataforma Android

Agenda

- Como armazenar as opções de preferência de uma aplicação?
- Como tocar músicas em uma atividade Android?
- Como armazenar dados usando Bundles?
- Como usar o sistema de arquivos do aparelho celular?
- Como manipular arquivos de texto em Java?
- Como armazenar objetos diretamente em arquivos?

Armazenamento permanente

- Dados armazenados permanentemente não desaparecem quando a aplicação termina sua execução.



Em que situações precisamos de armazenamento persistente?

Pense em aplicações Android que usam persistência.

Armazenamento permanente

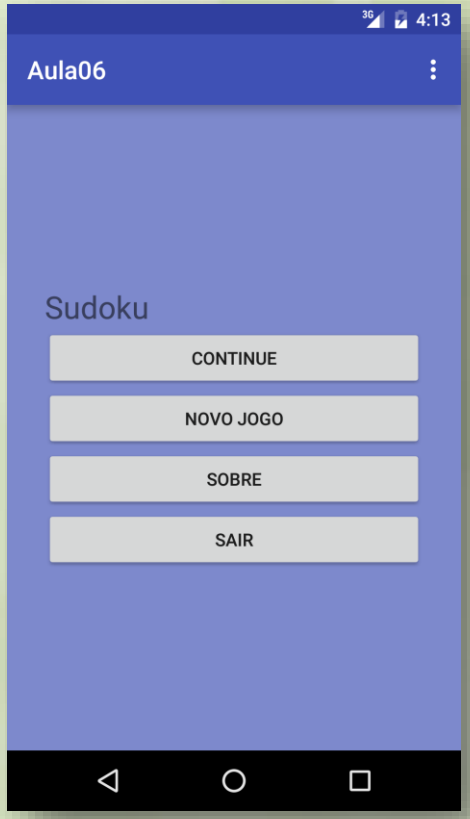
- Dados armazenados permanentemente não desaparecem quando a aplicação termina sua execução.
  - Guardar a lista de compras;
  - Armazenar a lista de endereços;
  - Lembrar a posição das peças de xadrez no tabuleiro;
  - Armazenar as opções escolhidas para uma aplicação;
  - etc

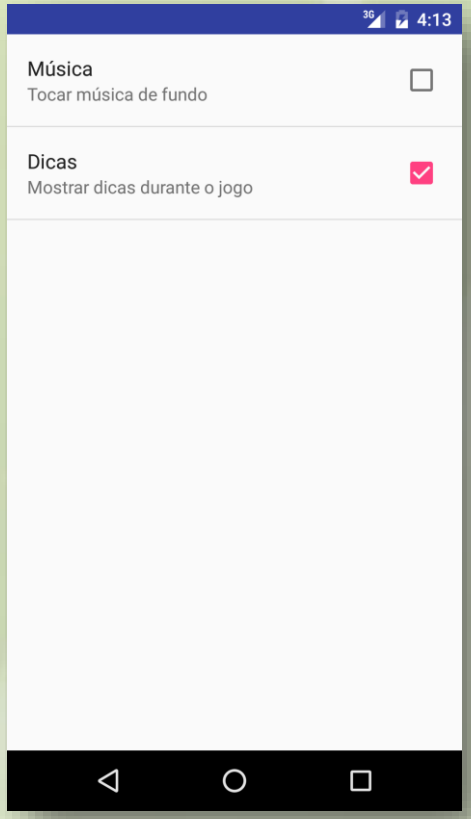
Muitas formas de Armazenamento

- API de preferências
- Estados de “*Bundles*”
- Arquivos de armazenamento em memória *flash*
- etc

Preferências

- Nosso Sudoku possui um menu de opções
  - Você lembra como isso foi implementado?





Modifique a classe Prefs para que ela armazene e retorne as preferências escolhidas pelo usuário.

Armazenamento de preferências

```
private static final String OPT_MUSIC = "music";
private static final boolean OPT_MUSIC_DEF = true;
private static final String OPT_HINTS = "hints";
private static final boolean OPT_HINTS_DEF = true;

public static boolean getMusic(Context context) {
    return PreferenceManager.getDefaultSharedPreferences
            (context).getBoolean(OPT_MUSIC, OPT_MUSIC_DEF);
}
public static boolean getHints(Context context) {
    return PreferenceManager.getDefaultSharedPreferences
            (context).getBoolean(OPT_HINTS, OPT_HINTS_DEF);
}
```

Prefs.java

Como "aplicar" essas referências?

Tocar ou não tocar uma música

```
public static void play(Context context, int resource){
    stop(context);
    mediaPlayer = MediaPlayer.create(context, resource);
    mediaPlayer.setLooping(true);
    mediaPlayer.start();
}
```

A classe Prefs se comporta como um singleton.

Music.java

```
public static void play(Context context, int resource){
    stop(context);
    if(Prefs.getMusic(context)) {
        mediaPlayer = MediaPlayer.create(context, resource);
        mediaPlayer.setLooping(true);
        mediaPlayer.start();
    }
}
```

E como habilitar ou desabilitar as "dicas"?

Removendo as dicas

PuzzleView.java

```
if(Prefs.getHints(getContext())) {
    // A cor da dica é baseada no número de opções restantes
    Paint hint = new Paint();
    int c[] = {ContextCompat.getColor(getContext(), R.color.puzzle_hint_0),
            ContextCompat.getColor(getContext(), R.color.puzzle_hint_1),
            ContextCompat.getColor(getContext(), R.color.puzzle_hint_2),};
    Rect r = new Rect();
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            int movesleft = 9 - game.getUsedTiles(i, j).length;
            if (movesleft < c.length) {
                getRect(i, j, r);
                hint.setColor(c[movesleft]);
                canvas.drawRect(r, hint);
            }
        }
    }
}
```

Continuando um jogo

A opção continue
Modifique a nossa implementação de Sudoku para que seja possível interromper e continuar um jogo interrompido

Seria possível utilizar a API de preferências?

A API de preferências no fundo é uma tabela hash persistente!

Como um jogo novo é obtido

```
private int[] getPuzzle(int diff) {
    String puz;
    switch (diff) {
        case DIFFICULT_HARD:
            puz = hardPuzzle;
            break;
        case DIFFICULT_MEDIUM:
            puz = mediumPuzzle;
            break;
        case DIFFICULT_EASY:
        default:
            puz = easyPuzzle;
            break;
    }
    currentPuzzle = puz;
    return fromPuzzleString(puz);
}
```

Game.java

Podemos adicionar uma nova opção para que seja possível escolhermos o puzzle a partir da tabela de preferências!

```
private int[] getPuzzle(int diff) {
    String puz;
    switch (diff) {
        case DIFFICULTY_CONTINUE:
            puz = getPreferences(MODE_PRIVATE)
              .getString(PREF_PUZZLE, easyPuzzle);
            break;
            ...
    }
    currentPuzzle = puz;
    return fromPuzzleString(puz);
}
```

Game.java

E como colocar o estado do tabuleiro na tabela de preferências?

Quando o puzzle deve ser salvo?

onPause

Game.java

```
protected void onPause() {
    super.onPause();
    Music.stop(this);
    getPreferences(MODE_PRIVATE).edit().putString(PREF_PUZZLE,
            toPuzzleString(puzzle)).commit();
}
```

E como podemos começar um jogo que havia sido interrompido?

Em outras palavras, como tratar o botão continue?

Eventos de clique

```
public void onClick(View v) {
    switch (v.getId()){
        case R.id.about_button:
            Intent i = new Intent(this, AboutActivity.class);
            startActivity(i);
            break;
        case R.id.new_button:
            iniciarNovoJogo();
            break;
        case R.id.exit_button:
            finish();
            break;
    }
}
```

Sudoku.java

Altere essa classe para que o botão continue possa funcionar

Eventos de clique

```
public void onClick(View v) {
    switch (v.getId()){
        case R.id.about_button:
            Intent i = new Intent(this, AboutActivity.class);
            startActivity(i);
            break;
        case R.id.new_button:
            iniciarNovoJogo();
            break;
        case R.id.exit_button:
            finish();
            break;
        case R.id.continue_button:
            novoJogo(Game.DIFFICULTY_CONTINUE);
            break;
    }
}
```

Sudoku.java

Ei, mas está faltando associar o evento ao botão!

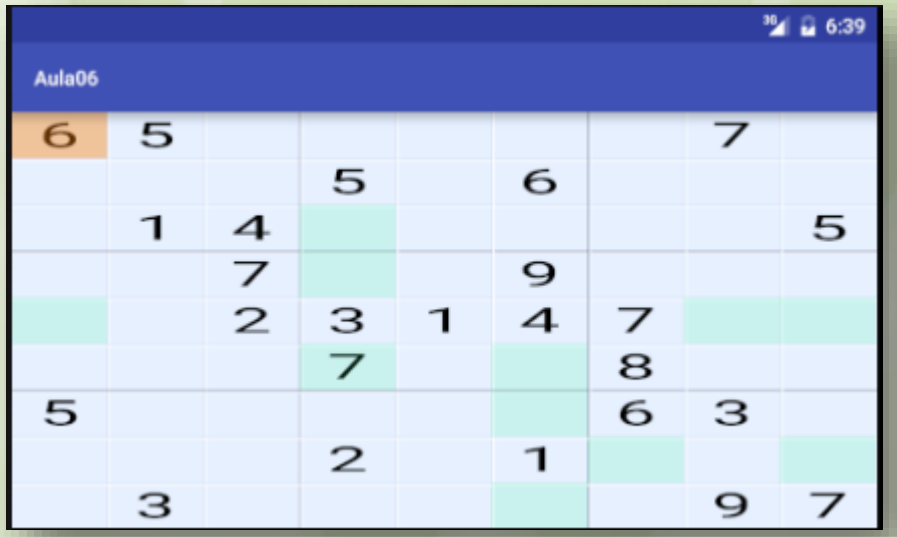
Botões e eventos

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sudoku);

    View aboutButton = findViewById(R.id.about_button);
    aboutButton.setOnClickListener(this);
    View newButton = findViewById(R.id.new_button);
    newButton.setOnClickListener(this);
    View exitButton = findViewById(R.id.exit_button);
    exitButton.setOnClickListener(this);
    View continueButton = findViewById(R.id.continue_button);
    continueButton.setOnClickListener(this);
}
```

Orientação da tela

- Se virarmos a tela em modo paisagem, nossa aplicação perde a informação sobre onde estava o cursor
  - Como podemos resolver este problema?



Estado da visão

- Toda visão possui um estado: posição do cursor, números, etc
- Quando trocamos a visão, o método onDraw da visão é invocado

E como podemos agora salvar a posição do cursor?

Bundles

- “Bundles” são tabelas usadas para passar dados entre atividades
  - Podem ser atividades diferentes
  - Ou a mesma atividade, no passado e no futuro
    - Neste caso, vamos utilizar dois métodos

```
public void onSaveInstanceState(Bundle outState)
```

```
protected void onCreate(Bundle savedInstanceState)
```

Salvando os dados

Game.java

```
public void onSaveInstanceState(Bundle outState) {
    outState.putString(JOGO_ATUAL, toPuzzleString(puzzle));
    outState.putString(CURRENT_PUZZLE, currentPuzzle);
    outState.putInt(SELX, puzzleView.getSelX());
    outState.putInt(SELY, puzzleView.getSelY());
    super.onSaveInstanceState(outState);
}
```

PuzzleView.java

```
public int getSelX() { return selX; }
public void setSelX(int selX) { this.selX = selX; }
public int getSelY() { return selY; }
public void setSelY(int selY) { this.selY = selY; }
```

Restaurando os dados

Game.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    puzzleView = new PuzzleView(this);
    if(savedInstanceState != null){
        String jogoAtual = savedInstanceState.getString(JOGO_ATUAL);
        puzzle = fromPuzzleString(jogoAtual);
        currentPuzzle = savedInstanceState.getString(CURRENT_PUZZLE);
        puzzleView.setSelX(savedInstanceState.getInt(SELX));
        puzzleView.setSelY(savedInstanceState.getInt(SELY));
    }
    else {
        int difficulty = getIntent().getIntExtra(KEY_DIFFICULTY,
              DIFFICULTY_EASY);
        puzzle = getPuzzle(difficulty);
    }
    calculateUsedTiles();
    setContentView(puzzleView);
    puzzleView.requestFocus();
}
```

O sistema de arquivos

- O Android OS é Linux
  - E portanto, possui um sistema de arquivos
- Arquivos podem ser manipulados pelas classes na biblioteca `java.io`
- Cada aplicação possui seu próprio espaço
  - Normalmente `data/data/nome_pacote`
- E a classe `Context` possui métodos para manipular arquivos armazenados nesse espaço

Manipulação de arquivos

- A classe `Context` possui diversos métodos para manipular arquivos
  - `deleteFile`: apaga um arquivo e retorna verdadeiro caso a deleção tenha acontecido
  - `fileList`: retorna um arranjo de strings com o nome dos arquivos no espaço da aplicação
  - `openFileInput`: abre um arquivo para leitura
  - `openFileOutput`: abre um arquivo para escrita

Arquivos de texto

- Podemos armazenar qualquer tipo de arquivo como “recursos crus”, em `res/raw/`

Exibição de texto
Implemente uma classe que abra, leia e exiba o conteúdo de um arquivo de texto armazenado em `/res/raw/file1.txt`

Layout da atividade

texto_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</ScrollView>
```

Lendo um arquivo

```
public class LeitorDeTextoActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.texto_layout);
        TextView text = (TextView) findViewById(R.id.textView);
        text.setText(getTextFromFile());
    }
    private String getTextFromFile() {
        StringBuffer contents = new StringBuffer();
        try {
            InputStream rawRes = getResources().openRawResource(R.raw.file1);
            BufferedReader input =
                    new BufferedReader(new InputStreamReader(rawRes));
            String line = null;
            while ((line = input.readLine()) != null) {
                contents.append(line + '\n');
            }
        } catch (IOException ex) { ex.printStackTrace(); }
        return contents.toString();
    }
}
```

Exercícios: Contraste

- Modifique a aplicação Sudoku, para que a visão do jogo tenha dois modos: “contraste” e “sem contraste”. A visão com contraste deve usar um fundo branco, e linhas maiores negras.
    - Essa opção também deve ser definida no layout de preferências, via um botão de seleção.

Prefs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <CheckBoxPreference
        android:key="music"
        android:title="@string/music_title"
        android:summary="@string/music_summary"
        android:defaultValue="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBoxPreference
        android:key="hints"
        android:title="@string/hints_title"
        android:summary="@string/hints_summary"
        android:defaultValue="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBoxPreference
        android:key="contrast"
        android:title="@string/contrast_title"
        android:summary="@string/contrast_summary"
        android:defaultValue="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</PreferenceScreen>
```

Strings.xml

```
<resources>
    ...

    <string name="contrast_title">Contraste</string>
    <string name="contrast_summary">Aumentar o contraste da tela</string>

    ...
</resources>
```

Prefs.java

```
...
private static final String OPT_CONTRAST = "contrast";
private static final boolean OPT_CONTRAST_DEF = true;

...

public static boolean getContrast(Context context) {
    return PreferenceManager.getDefaultSharedPreferences
            (context).getBoolean(OPT_CONTRAST, OPT_CONTRAST_DEF);
}
```

Precisamos agora aplicar o contraste

Escolhendo novas cores

- O modo de contraste deve usar linhas grandes negras e fundo branco

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    ...
    <color name="puzzle_big_lines">#ff000000</color>
    <color name="puzzle_light_background">#ffffffff</color>
</resources>
```

E como, agora usar as preferências?

PuzzleView.java

```
Paint backgroundPaint = new Paint();
backgroundPaint.setStyle(Paint.Style.FILL);
if(Prefs.getContrast(getContext())) {
    backgroundPaint.setColor(ContextCompat.getColor(getContext(),
          R.color.puzzle_light_background));
}
else{
    backgroundPaint.setColor(ContextCompat.getColor(getContext(),
          R.color.puzzle_background));
}
...
if(Prefs.getContrast(getContext())) {
    darkPaint.setColor(ContextCompat.getColor(getContext(),
          R.color.puzzle_big_lines));
}
else {
    darkPaint.setColor(ContextCompat.getColor(getContext(),
          R.color.puzzle_dark));
}
```

Dúvidas