Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Programação de Dispositivos Móveis
Guia para Aula Laboratorial 3
Licenciatura em Engenharia Informática

Programming of Mobile Devices
Guide for Laboratory Class 3
Degree in Computer Science and Engineering

Sumário
Introdução à programação e desenvolvimento de interfaces de utilizador orientadas por eventos para aplicações Android™. Implementação de uma aplicação que simula uma calculadora com as operações mais básicas como forma de estudar a criação de consumidores para objetos interativos e o ficheiro *eXtended Markup Language* (XML) que define o desenho da interface de utilizador.

Summary
Introduction to event based programming and development of user interfaces for Android applications. Implementation of an application that simulates a calculator with the most basic operations, with the excuse to introduce the creation of listeners for widgets and the eXtended Markup Language (XML) file that defines the design of the user interface.

Pré-requisitos:
Algumas das tarefas enunciadas a seguir requerem o acesso a um sistema com o Android Studio e com o SDK Android™, bem como com a Gradle™ instalados ou, alternativamente, com permissões para instalação e configuração do IDE, *kit* e ferramenta. Serão suficientes permissões para criar diretorias e ficheiros num disco local e para configurar variáveis de sistema, nomeadamente a *path*. É necessário ter acesso a uma versão e imagem da plataforma Android™ ou a um dispositivo físico com o sistema operativo e com a opção de *debug* ativa. É igualmente necessário ter um compilador Java instalado.

1 Preliminares
Preliminaries

O guia laboratorial 2 elabora nos passos necessários a criação e compilação (*build*) de projetos de aplicações para a plataforma Android™ via linha de comandos. Esta abordagem, apesar de não comportar algumas das facilidades oferecidas por ambientes de desenvolvimento integrados, nomeadamente ambientes de edição da interface de utilizador *What You See Is What You Get* (WYSIWYG), permite conhecer em maior profundidade os detalhes de implementação de uma aplicação Android™, mas requer que, após instalação do Android Studio, se atualize e instalem as várias ferramentas do *Software Development Kit* (SDK) Android™[1]. Depois do sistema estar devidamente configurado, 4 passos são suficientes para criar um projeto Android™, gerar o ficheiro `.apk` e instalar a aplicação num dispositivo (virtual ou real):

1. Inicializar o dispositivo móvel virtual ou ligar um real ao computador[2];
2. Gerar o projeto através do Android Studio;
3. Compilar o projeto com a ferramenta Gradle™, emitindo o comando `$ gradlew assembleDebug` na raiz do projeto;
4. Instalar a aplicação com um comando semelhante a `$ adb install -r path\NomeApp-debug.apk`.

[1] Ver `https://developer.android.com/studio/intro/update`.
[2] Se o dispositivo for real, tem de ter a opção de depuração ativada.

Tarefa 1 *Task 1*

Como já vem sendo habitual, a primeira tarefa consiste em iniciar um *Android Virtual Device* (AVD). Para isso, pode emitir o comando `$ emulator`, incluído na pasta `emulator` do SDK, e lançar um AVD. Caso não exista nenhum AVD configurado, crie um[3]. O ideal será um emulador de uma versão superior à 6.0 do Sistema Operativo (SO).

Q1.: Qual das seguintes combinações lista todos os AVDs disponíveis com o comando `$ emulador`?

☑ `$ emulator -list-avds`
☐ `$ emulator -avd list`

Tarefa 2 *Task 2*

Verifique se as variáveis `ADB_HOME` e `PATH` estão devidamente definidas com comandos parecidos com:
`$ echo %ANDROID_HOME%`

[3] O guia laboratorial 1 contém uma breve discussão acerca deste assunto.

The “PATH” is a special variable that saves

```
$ echo %PATH%
```

Caso as variáveis já estejam devidamente definidas, passe para a secção seguinte. Caso contrário, precisa de as definir antes de avançar, com comandos semelhantes a:

```
$ set ANDROID_HOME=C:\installation location\sdk
```

```
$ set PATH=%PATH%; %ANDROID_HOME%\tools;
%ANDROID_HOME%\emulator;
%ANDROID_HOME%\platform-tools
```

Nota: Para mais detalhes consultar o guia laboratorial 2.

2 Introdução às Interfaces de Utilizador Orientadas por Eventos

Introduction to Event Oriented User Interfaces

O grande objetivo deste guia laboratorial é o de introduzir a programação e o desenvolvimento de interfaces de utilizador orientadas por eventos para aplicações móveis. Antes de prosseguir, procure a definição de *interface* (no contexto da programação) e use o espaço seguinte para incluir essa definição:

> In computing, an interface is a boundary that separates 2 sides (computer, user, etc...). Through this virtual barrier, information can be exchanged and shared.

Q2.: Consegue aperceber-se da diferença entre uma interface de utilizador e uma interface entre duas componentes de um programa?
☐ Não. :_(
☑ Agora que penso nisso... sim, consigo.

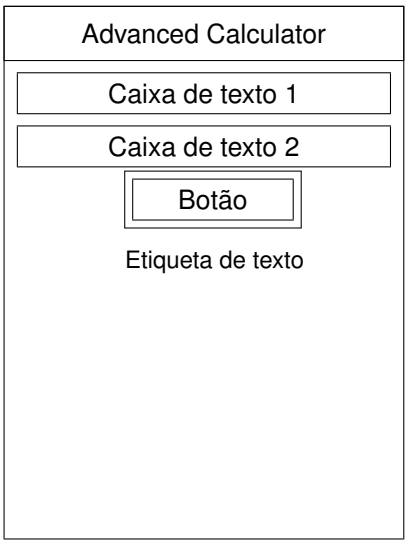
Tarefa 3 *Task 3*

Usando o Android Studio, crie e teste um projeto padrão de uma aplicação Android™ sobre o qual vai elaborar nas próximas tarefas. Considere as seguintes sugestões aquando da preparação do projeto:

- O nome do projeto deve ser `AdvancedCalculator`;
- A raiz do projeto deve ser uma pasta com permissões de escrita para o utilizador atual, idealmente na área de aluno (e.g., `C:\Users\aluno\workspace\ACalculator`)
- O nome do pacote deve ser semelhante a `pt.ubi.di.pmd.acalculator`;
- O projeto deve possuir apenas uma *Empty Activity*;
- O nome da atividade principal pode ser `ACalculator`.

Tarefa 4 *Task 4*

Depois de se certificar que o esqueleto base não apresenta problemas, mude o visual da aplicação. A tarefa consiste em adicionar duas caixas de texto editáveis seguidas, um botão e uma etiqueta de texto. Considere que vai criar uma calculadora que apenas sabe somar dois números reais. As duas caixas de texto servirão para a introdução dos dois números a somar, enquanto que o resultado aparecerá na etiqueta de texto após o botão ser pressionado. A figura seguinte esquematiza o visual pretendido para a aplicação.



Para conseguir mudar o *layout* da aplicação, abra e edite, com o bloco de notas (`$ notepad.exe nome-do-ficheiro.xml`), o ficheiro XML que contém a definição desse *layout* para a aplicação principal. **Q3.: Em que local pode o ficheiro mencionado ser encontrado?**
☑ `src\main\res\layout` ☐ `libs\layout`
☐ `gen\layout` ☐ `src\main\res`
☐ Em casa, a descansar.

Quando abrir o ficheiro, vai notar que já existe pelo menos uma etiqueta de texto definida. Faça 4 cópias do elemento que a define no XML e altere o nome de três desses elementos para `EditText` (dois desses elementos) e `Button`. Mude o valor da propriedade `android:text` do primeiro elemento para `"This is a very Advanced Calculator!"`. No final deve ter 5 elementos definidos com a seguinte ordem:

- 1 `TextView`;
- 2 `EditText`;
- 1 `Button` (já agora, deve ajustar o texto do botão para o símbolo +) e
- 1 `TextView`.

Depois de fazer alterações mencionadas antes, compile e teste a aplicação. **Q4.: Funcionou?**
☑ Sim senhor. Está tudo a rolar.
☐ Não... deve ter havido algum problema.

Deve verificar que os vários objetos se encontram sobrepostos. Isto deve-se a dois fatores: (i) os diferentes objetos não têm identificadores (IDs) diferentes; (ii) não são definidas corretamente as **restrições** de posicionamento dos objetos. Note que cada um dos objetos devia estar por baixo do anterior.

Para resolver a questão dos IDs, volte a abrir para edição o referido ficheiro de *layout* e adicione a propriedade `android:id="@+id/nome_do_recurso"` a cada um dos elementos que adicionou anteriormente. Os IDs a atribuir às duas caixas de texto, ao botão e à etiqueta de texto devem ser, respetivamente:

- `number1`;
- `number2`;
- `sum`;
- `result`.

Agora, é necessário escolher a **restrição** *(constraint)* a usar em cada um dos objetos. **Q5.: Qual das seguintes deverá usar?**
☐ `app:layout_constraintBottom_toBottomOf`
☐ `app:layout_constraintLeft_toLeftOf`
☐ `app:layout_constraintRight_toRightOf`
☑ `app:layout_constraintTop_toBottomOf`

Independentemente do que respondeu antes, note que cada objeto definido no XML deve, no final deste exercício, ter uma definição semelhante à seguinte (com as devidas adaptações):

```
<Button
  android:id="@+id/sum"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text=" + "
  app:layout_constraintLeft_toLeftOf="parent"
  app:layout_constraintRight_toRightOf="parent"
  app:layout_constraintTop_toBottomOf="@+id/number2"
  />
```

Q6.: Não pergunta ao Prof. pelo significado do `@` e do `+` na definição da propriedade `android:id`?
☐ Não, porque já sei tudo.
Use o espaço em baixo para dizer o que significam os dois símbolos:

@

somewhere in specific location

+

if, while being compiled, the ID Class doesn´t exist, it will be generated to fill in the gap

Tarefa 5 ***Task 5***

Se tudo correu bem até este passo, deve ter conseguido desenvolver uma aplicação minimalista (e com uma só atividade), já com um design semelhante ao que foi pedido, e sem ter de escrever ou manipular uma única linha de código Java. **Q7.: Estes factos vão de encontro ao que foi referido em relação à arquitetura de software MVC?**
☐ Isto não tem nada a ver com a arquitetura MVC.
☐ De facto, isto parece ir de encontro àquela arquitetura de desenvolvimento de software, já que o código está perfeitamente separado das vistas.
☐ De facto, isto parece ir de encontro àquela arquitetura de desenvolvimento de software na medida em que as vistas sobre os dados são definidas em XML.
☐ De facto, isto parece ir de encontro àquela arquitetura de desenvolvimento de software na medida em que as vistas sobre os dados são definidas em Java.
☐ De facto, isto parece ir de encontro àquela arquitetura de desenvolvimento de software na medida em que o controlo está separado do código da aplicação.

Procure e abra para edição o ficheiro contendo o código fonte da única atividade da aplicação que criou. **Q8.: Já se sabe que estará dentro da diretoria `src`, mas qual é o caminho completo do ficheiro a partir da raiz da aplicação?**
☐ `src/ACalculator.java`
☐ É um comboio desgraçado.
☐ `app/src/main/java/pmd/di/ubi/pt/acalculator/...`

Depois de abrir o ficheiro para edição no notepad, adicione os dois `imports` seguintes:

```
import android.view.*;
import android.widget.*;
```

Q9.: Consegue encontrar a linha de código que *carrega* o *layout* da aplicação a partir do ficheiro?
☐ Sim, é a linha com a instrução

```
super.onCreate(savedInstanceState);
```

☑ Sim, é a linha com a instrução

```
setContentView(R.layout.activity_acalculator);
```

☐ Não, não consigo.

Note que, para poder manipular, alterar ou aceder a valores contidos nos objetos interativos (*widgets*) mencionados na tarefa anterior, precisa de os instanciar na forma de objetos (Java) no código. Para isso, considere declarar os 4 objetos seguintes **antes** do método `onCreate()`:

```
EditText oTEdit1;
EditText oTEdit2;
Button oButton;
TextView oTView1;
```

Dentro do método `onCreate()`, pode instanciar estes objetos através de instruções semelhantes a:

```
oTEdit1 = (EditText) findViewById(R.id.number1);
oTEdit2 = (EditText) findViewById(R.id.number2);
oButton = (Button)   findViewById(R.id.SUM);
oTView1 = (TextView) findViewById(R.id.result);
```

Q10.: Por que é que é que cada vez que declara a função `findViewById(...)` necessita de colocar o nome de uma classe de um objeto entre parêntesis antes?

- ☐ Só para ter a certeza de que o objeto devolvido é da classe desejada.
- ☑ O protótipo da função mencionada obriga a que seja chamada desta forma.
- ☐ Se não se colocar nada entre parêntesis antes de chamar a função, o Java é como que fica chateado comigo... e debita erros, e assim.
- ☐ A função devolve sempre um objeto mais geral, da classe `View`, que precisa ser convertido para a subclasse mais específica através de um *cast*.

Tarefa 6 *Task 6*

Note que o método `findViewById( ID )` aceita o `ID` (de IDentificador) do objeto interativo definido no ficheiro XML de *layout*. Estes IDs já foram anteriormente definidos.

Repare que se está a usar código semelhante a `R.algo`, é porque existe uma classe chamada `R` algures no seu projeto. A sua implementação deve estar num ficheiro chamado `R.java` (porque o Java costuma obrigar a que o nome dos ficheiros tenha o mesmo nome das classes neles implementadas). **Q11.: Agora sem olhar (!), consegue encontrar este ficheiro?**

- ☐ Sim, algures em `build/generated/source ...`
- ☐ Sim, em `build/generated/source/r/debug ...`
- ☐ Sim, algures dentro da pasta `src`.
- ☐ Sim, algures dentro da pasta `gradle`.

Aproveite o facto de estar a editar o ficheiro XML para explorar algumas particularidades do *layout* através da alteração das propriedades dos elementos. Por exemplo, experimente mudar a propriedade `android:layout_width="wrap_content"` do **botão** para `android:layout_width="fill_parent"`. **Q12.: Qual o efeito desta alteração na interface de utilizador da aplicação?**

- ☐ O botão passa a estar na horizontal, em vez de estar na vertical.
- ☐ O botão passa a estar na vertical, em vez de estar na horizontal.
- ☐ O botão passa a estar obliquo, grande maluco!
- ☐ O botão passa a ter um tamanho que se coaduna com o tamanho do texto que contém.
- ☑ O botão passa a ocupar o máximo do ecrã na horizontal.

Tarefa 7 *Task 7*

Adicione um consumidor (*Listener*) ao objeto botão de forma a definir que, após este ser clicado, a operação por ele representada é efetuada sobre os dois inputs nas caixas de texto (`oTEdit1` e `oTEdit2`) e o resultado é mostrado na etiqueta de texto `oTView1`. O excerto de código seguinte, onde falta uma instrução, pode ser útil na resolução desta tarefa.

```
oButton.setOnClickListener(
  new View.OnClickListener()
  {
    public void onClick(View oView)
    {
      double d1 = (new Double(oTEdit1.getText().
          toString())).doubleValue();
      double d2 = (new Double(oTEdit2.getText().
          toString())).doubleValue();
      double dSum = d1 + d2;
      // FALTA INSTRUCAO
    }
  });
```

Q13.: Que nome se dá à classe criada pela instrução `new View.onClickListener()`?

- ☐ Classe objeto.
- ☐ Classe local.
- ☐ Pseudo classe.
- ☑ Classe anónima.

Tarefa 8 *Task 8*

Implemente as 3 operações que faltam para obter uma calculadora básica (i.e., subtração, multiplicação e divisão).

Tarefa 9 *Task 9*

Aproveite o facto da aplicação já aceitar 2 valores para sobre eles efetuar um cálculo para implementar as funções de logaritmo do `number1` na base `number2` e a de potência (`number1` elevado a `number2`).

Tarefa 10 *Task 10*

Adicione um botão que permita copiar o resultado de uma operação para o *clipboard*. O conteúdo do URL `http://developer.android.com/guide/topics/text/copy-paste.html` pode ser-lhe útil para a execução desta tarefa.

Q14.: Já agora, só por curiosidade, o que significa URL?

U niform R esource L ocator

Tarefa 11 *Task 11*

Esta tarefa é só para os mais corajosos: arranje forma

de atualizar a etiqueta de texto com o resultado **da soma** dos dois números sempre que for introduzido um número na segunda caixa de texto. Note que a etiqueta deve ser atualizada à medida que o número (potencialmente constituído por vários digitos) é inserido **Sugestão:** os procedimentos enunciados a seguir devem ser-lhe úteis na execução desta tarefa.

```
oTEdit2.addTextChangedListener(
  new TextWatcher(){
    public void afterTextChanged(Editable s) {
      ...
  }
```