Roteiro de Laboratório 03

José Cassiano Gunji

1. Codificação do Aplicativo Calculadora de Gorjeta

No ponto em que estamos, você já pode compilar e executar o aplicativo para testar o *layout* e verificar se ele está funcionando corretamente. Entretanto, o aplicativo ainda não consegue realizar qualquer tarefa. Para começar, vamos criar uma classe Java convencional para implementar os métodos que farão os cálculos das gorjetas.

> Lembre-se, a *activity* é uma classe de estereótipo fronteira e não deve realizar cálculos relacionados ao negócio. Tais cálculos devem ser realizados por outra classe, uma com o estereótipo controle.

No navegador de projeto, clique com o botão direito o pacote de seu aplicativo e selecione [new] -> [Java class].

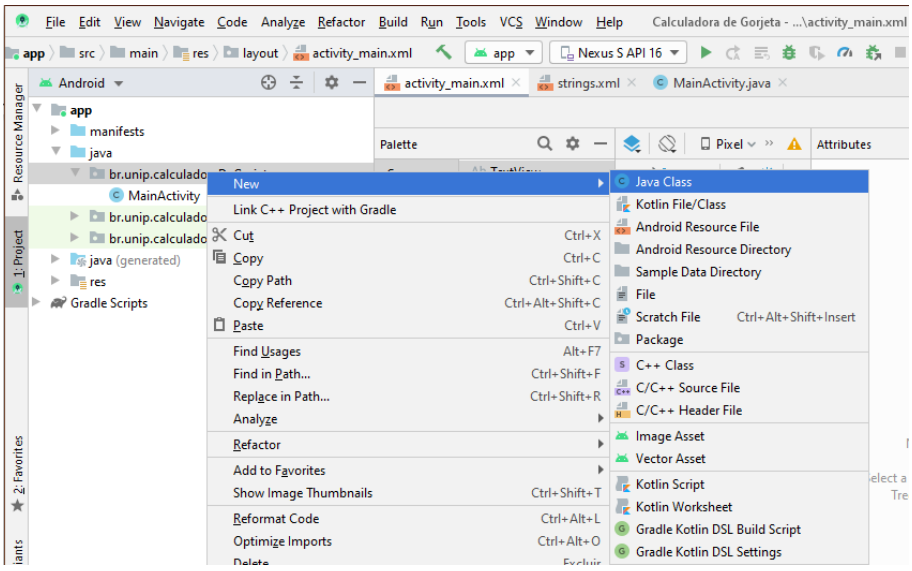


Figura 1 – Crie uma nova classe Java

A seguir escreva o código da classe Calculadora, como mostrado na figura a seguir:

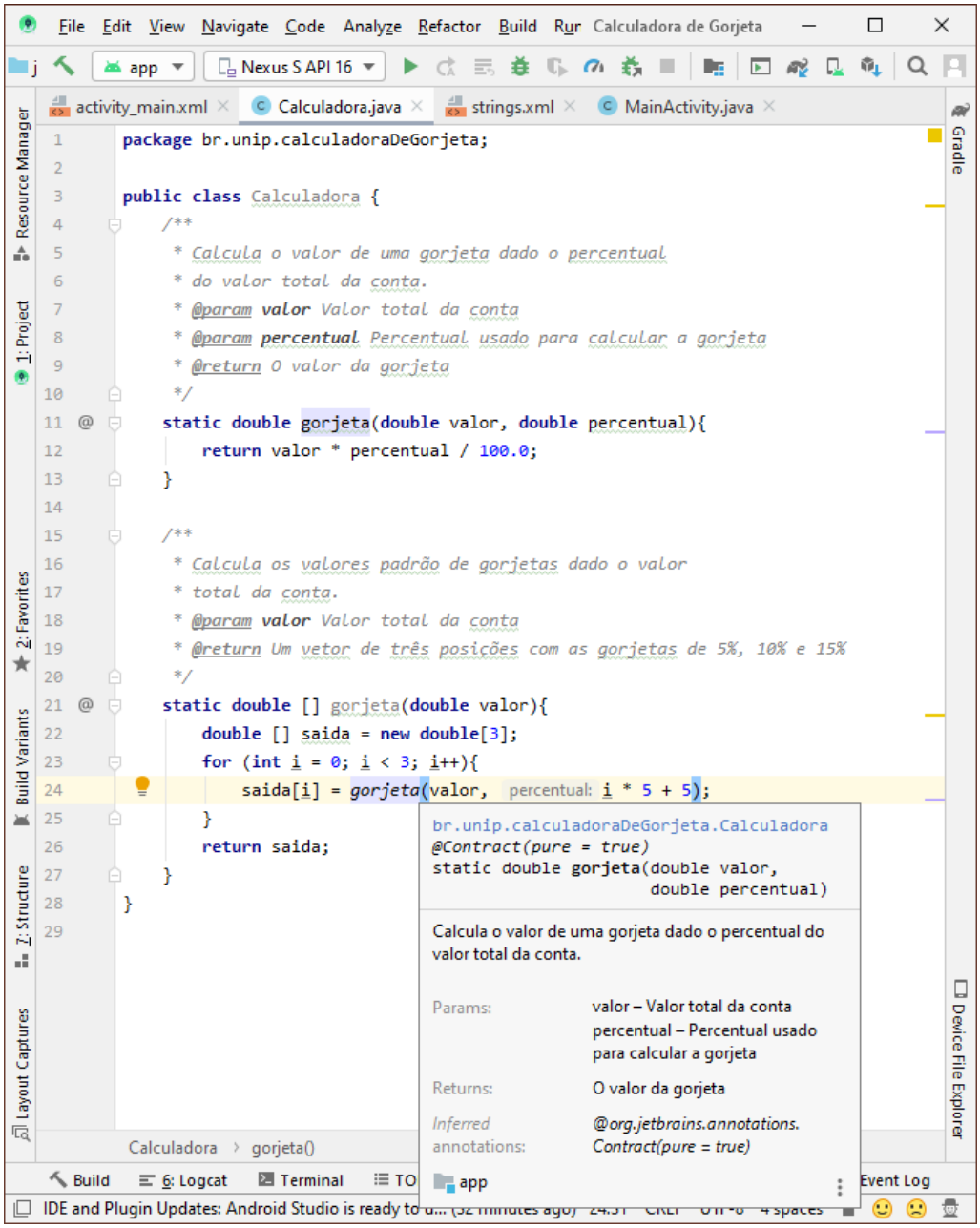


Figura 2 – Código da classe Calculadora

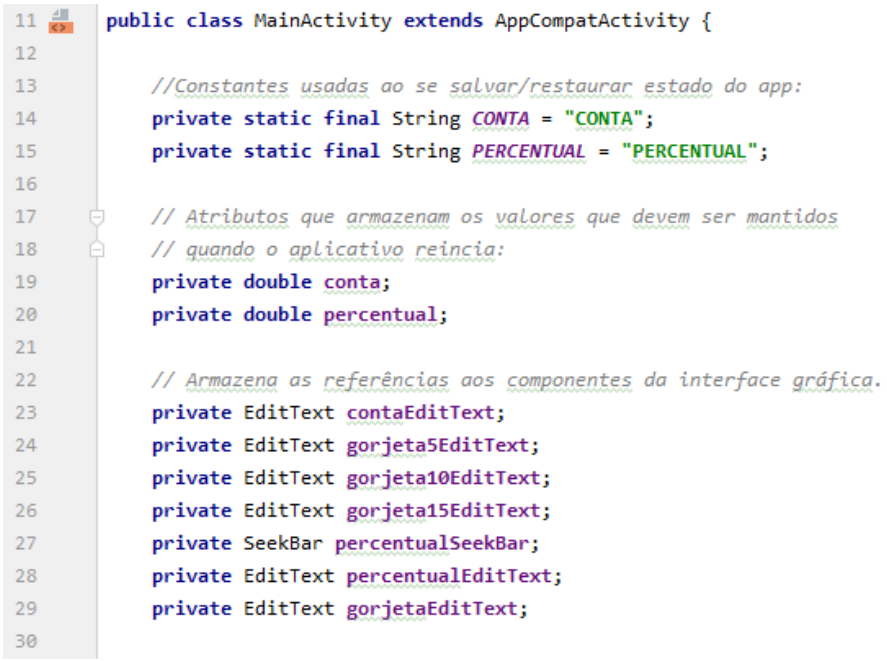
No código apresentado na Figura 1, usamos um formato especial de comentário, usando o símbolo “/**”. Este símbolo, que funciona como o comentário em bloco, além de estabelecer um bloco de comentário, estabelece também um bloco de documentação. Para usar este recurso, primeiro escreva ao menos a assinatura do método. A seguir, na linha acima do método, digite “/**” e tecle [Enter]. A IDE já irá criar um bloco de documentação listando os parâmetros do método e o seu retorno (se houver). Escreva agora uma descrição para o método e para cada um dos parâmetros, assim como para o valor de retorno. Agora, sempre que este método for usado em qualquer parte do aplicativo, esta documentação pode ser apresentada para auxiliar o programador. Por exemplo, na linha 24, com o cursor sobre o método gorjeta(), clique em [CTRL] + [Q]. Veja como o Android Studio apresenta a documentação que você acabou de criar. Isto funciona em qualquer IDE que não seja absurdamente antiga.

Agora vamos editar o arquivo *MainActrivity.java*.

```
1  package br.unip.calculadoraDeGorjeta;
2
3  import androidx.appcompat.app.AppCompatActivity;
4  import android.os.Bundle;
5  import android.os.PersistableBundle;
6  import android.text.Editable;
7  import android.text.TextWatcher;
8  import android.widget.EditText;
9  import android.widget.SeekBar;
```

Figura 3 – Importações de pacotes da *MainActivity*

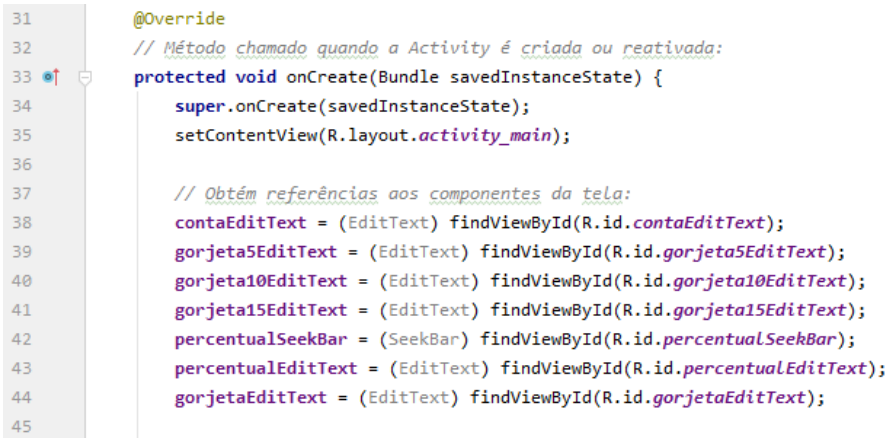
As primeiras linhas do arquivo definem o pacote e as importações. Você pode definir as importações manualmente ou permitir que a correção automática crie estas linhas para você conforme for escrevendo o restante do código.



```
11  public class MainActivity extends AppCompatActivity {
12
13      //Constantes usadas ao se salvar/restaurar estado do app:
14      private static final String CONTA = "CONTA";
15      private static final String PERCENTUAL = "PERCENTUAL";
16
17      // Atributos que armazenam os valores que devem ser mantidos
18      // quando o aplicativo reincia:
19      private double conta;
20      private double percentual;
21
22      // Armazena as referências aos componentes da interface gráfica.
23      private EditText contaEditText;
24      private EditText gorjeta5EditText;
25      private EditText gorjeta10EditText;
26      private EditText gorjeta15EditText;
27      private SeekBar percentualSeekBar;
28      private EditText percentualEditText;
29      private EditText gorjetaEditText;
30
```

Figura 4 – Atributos da classe *MainActivity*

No início da classe são declarados seus atributos. Nem todos os atributos estão declarados aqui. Deixamos dois atributos mais para o final do código, pois eles requerem mais explicações.



```
31      @Override
32      // Método chamado quando a Activity é criada ou reativada:
33      protected void onCreate(Bundle savedInstanceState) {
34          super.onCreate(savedInstanceState);
35          setContentView(R.layout.activity_main);
36
37          // Obtém referências aos componentes da tela:
38          contaEditText = (EditText) findViewById(R.id.contaEditText);
39          gorjeta5EditText = (EditText) findViewById(R.id.gorjeta5EditText);
40          gorjeta10EditText = (EditText) findViewById(R.id.gorjeta10EditText);
41          gorjeta15EditText = (EditText) findViewById(R.id.gorjeta15EditText);
42          percentualSeekBar = (SeekBar) findViewById(R.id.percentualSeekBar);
43          percentualEditText = (EditText) findViewById(R.id.percentualEditText);
44          gorjetaEditText = (EditText) findViewById(R.id.gorjetaEditText);
45
```

Figura 5 – Início da codificação do método onCreate()

Na linha 33 declaramos a sobrescrita do método onCreate(), que é chamado sempre que a *Activity* é carregada ou recarregada. Na linha 34 chamamos o método onCreate() da superclasse. A linha 35 especifica que o arquivo de *layout* desta *Activity* é o activity_main. Este *layout* é obtido da classe R, que é uma classe do aplicativo que é escrita para nós pelo compilador *Gradle*. Essa classe não deve ser alterada. Nas linhas 38 a 44 obtemos referências aos componentes interativos usando o método findViewById(). Só podemos obter referências de componentes que tenham sua propriedade id definida, como fizemos anteriormente.

```
46      // Cria os ouvintes de eventos para as views interativas:
47      contaEditText.addTextChangedListener(ouvinteContaEditText);
48      percentualSeekBar.setOnSeekBarChangeListener(ouvintePercentualSeekBar);
```

Figura 6 – Associação dos ouvintes de eventos

Nestas linhas, definimos quais são os objetos que irão tratar os eventos de mudança de texto do contaEditText e de mudança do progresso do percentualSeekBar. Note que enquanto você digita estas linhas, os atributos ouvinteContaEditText e ouvintePercentualSeekBar ainda não foram declarados e serão destacados em vermelho como um erro. Não se preocupe. Mais abaixo estes atributos serão declarados e este erro desaparecerá.

```
50      // Verifica se o aplicativo acabou de ser iniciado ou se está sendo
51      // restaurado:
52      if (savedInstanceState == null){
53          conta = 0;
54          percentual = 7;
55      } else {
56          // o aplicativo está sendo restaurado da memória, não está sendo
57          // executado a partir do zero. Assim, os valores de conta e
58          // percentual são restaurados.
59          conta = savedInstanceState.getDouble(CONTA);
60          percentual = savedInstanceState.getDouble(PERCENTUAL);
61      }
62
63      // Atualiza os componentes gráficos com os valores atualizados:
64      contaEditText.setText(String.format("%.2f", conta));
65      percentualSeekBar.setProgress((int) percentual);
66  }
```

Figura 7 – Recuperação do estado do aplicativo

Se o aplicativo está sendo reiniciado, por exemplo, se a orientação do dispositivo mudou entre retrato e paisagem, é necessário recarregar os valores de conta e percentual. A linha 52 testa se o aplicativo não está sendo reiniciado. Se é a primeira vez que é executado, as linhas 53 e 54 definem valores padrão

para conta e percentual. Caso contrário, as linhas 59 e 60 recuperam os valores de conta e percentual que foram gravados pelo aplicativo anteriormente. O método que grava estes valores será visto mais adiante.

> Note que na linha 66 fechamos a chave do método *onCreate()*. A partir de agora, você vai criar novos métodos na classe. Esse é um ponto em que muita gente erra.

```
68      // Atualiza o valor das gorjetas padrão:
69      private void atualizaGorjetas(){
70          double [] gorjetas = Calculadora.gorjeta(conta);
71          gorjeta5EditText.setText(String.format("%.1f", gorjetas[0]));
72          gorjeta10EditText.setText(String.format("%.1f", gorjetas[1]));
73          gorjeta15EditText.setText(String.format("%.1f", gorjetas[2]));
74      }
75
76      // Atualiza o valor da gorjeta personalizada:
77      private void atualizaGorjetaPersonalizada(){
78          gorjetaEditText.setText(String.format("%.1f", Calculadora.gorjeta(conta,percentual)));
79      }
```

Figura 8 – Métodos que atualizam os valores calculados da *Activity*

Esses métodos são responsáveis por calcular os valores da gorjeta padrão (método atualizaGorjetas()) quanto para o valor da gorjeta personalizada (atualizaGorjetaPersonalizada()).

```
81      //Define o objeto ouvinte de mudança de texto do contaEditText:
82      private TextWatcher ouvinteContaEditText = new TextWatcher() {
83          @Override
84          public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
85              // Este método deve ser sobrescrito, mas não é usado neste aplicativo.
86          }
87
88          @Override
89          public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
90              try {
91                  conta = Double.parseDouble(contaEditText.getText().toString());
92              } catch (NumberFormatException e) {
93                  conta = 0;
94              }
95              atualizaGorjetas();
96              atualizaGorjetaPersonalizada();
97          }
98
99          @Override
100         public void afterTextChanged(Editable editable) {
101             // Este método deve ser sobrescrito, mas não é usado neste aplicativo.
102         }
103     };
```

Figura 9 – Evento de ação associado à mudança de texto de *contaEditText*

Essa é a declaração do atributo ouvinte *contaEditText* que foi atribuído anteriormente, na linha 47, como ouvinte de mudança de texto do *contaEditText*. Esse atributo recebe uma instância de uma classe anônima, ou seja, uma instância de uma classe que não recebeu um nome. Essa classe implementa a interface *TextWatcher* que define os métodos que estão sobrescritos nas linhas 84, 89 e 100. Todos eles devem ser sobrescritos, mas o único que é utilizado pelo nosso aplicativo é o método *onTextChanged()*. Ele obtém o valor da conta digitada pelo usuário, converte para *double*, armazena do atributo conta e chama os métodos que atualizam o cálculo das gorjetas.

Ao se digitar a linha “private TextWatcher ouvinte = new ”, após a palavra reservada “new”, dê um espaço e use a tecla “mágica” [Ctrl] + [Espaço]. O mecanismo de auto completar irá escrever a assinatura de todos os métodos que devem ser sobrescritos.

Repare que no trecho da figura anterior estamos criando uma instância anônima em uma atribuição, a atribuição iniciada na linha 82. Por isso, tal atribuição precisa ser encerrada com o ponto-e-vírgula na linha 103. Esse é outro ponto de erros comum.

```
105     // Define o objeto ouvinte de mudança no percentualSeekBar
106     private SeekBar.OnSeekBarChangeListener ouvintePercentualSeekBar =
107             new SeekBar.OnSeekBarChangeListener() {
108         @Override
109         public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
110             percentual = (double) percentualSeekBar.getProgress();
111             percentualEditText.setText(String.format("%.1f", percentual));
112             atualizaGorjetaPersonalizada();
113         }
114
115         @Override
116         public void onStartTrackingTouch(SeekBar seekBar) {
117             // Este método deve ser sobrescrito, mas não é usado neste aplicativo.
118         }
119
120         @Override
121         public void onStopTrackingTouch(SeekBar seekBar) {
122             // Este método deve ser sobrescrito, mas não é usado neste aplicativo.
123         }
124     };
```

Figura 10 – Evento de ação associado ao movimento do *percentualSeekBar*

Esta é a declaração do atributo ouvinte *PercentualSeekBar*, que foi utilizado na linha 48 quando foi atribuído como ouvinte de mudança de progresso do percentualSeekBar. Sua lógica de criação é análoga ao do atributo ouvinte anterior.

```
126     // Este método é chamado quando o aplicativo é interrompido. Nele criamos nossa lógica
127     // para armazenar as informações que devem ser recuperadas quando o aplicativo é reiniciado.
128
129     @Override
130     public void onSaveInstanceState(Bundle outState, PersistableBundle outPersistentState){
131         super.onSaveInstanceState(outState, outPersistentState);
132         outState.putDouble(CONTA, conta);
133         outState.putDouble(PERCENTUAL, percentual);
134     }
135 }
```

Figura 11 – Método que grava o estado atual do aplicativo

Por fim, o método *onSaveInstanceState()* é chamado pelo Android quando ele está a ponto de suspender o aplicativo. É neste método que gravamos as informações que devem ser recuperadas quando o aplicativo é recarregado. Note que estas informações só são mantidas enquanto o aplicativo está suspenso. Se ele for totalmente descarregado, estas informações serão perdidas.

Execute seu aplicativo em uma máquina virtual Android ou em um dispositivo real. Note que o aplicativo funciona bem em diversas resoluções de tela assim como em diversas orientações do dispositivo.

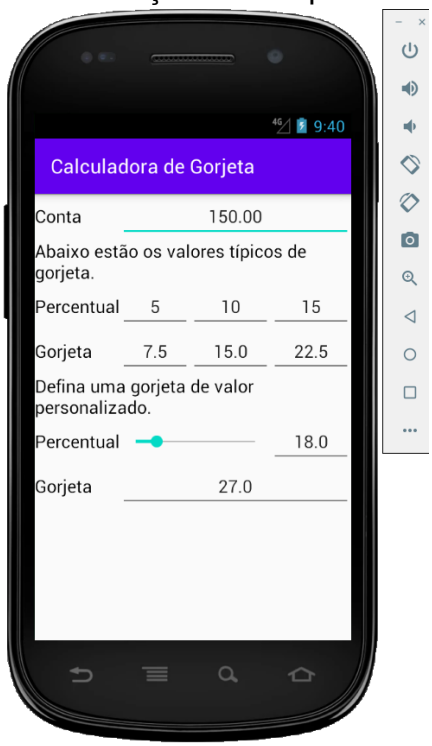


Figura 12 – Aplicativo sendo executado em um dispositivo virtualizado na orientação vertical

A Figura 13 apresenta o mesmo aplicativo sendo executado no mesmo dispositivo virtual, mas desta vez simulando a orientação horizontal. Note que a interface gráfica se ajustou corretamente à nova largura da tela. Como a altura

da tela tornou-se insuficiente para apresentar toda a interface gráfica, o aplicativo apresenta uma barra de rolagem á direita da tela que fica visível quando a interface gráfica é deslocada com o dedo, o ponteiro ou a roda do mouse.

Figura 13 – Aplicativo sendo executado em um dispositivo virtualizado na orientação horizontal

Você pode baixar o projeto para seu computador, assim como instalar o aplicativo diretamente em seu dispositivo Android se quiser vê-lo funcionando. Mas faça isso apenas se não tiver conseguido desenvolver o projeto por conta própria e se o fórum não estiver mais disponível para você.

https://github.com/Cassiano-Gunji/Calculadora-de-Gorjeta/releases/tag/1.0