* **ergunta 1**

0 em 0,6 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Analise as duas lógicas a seguir:  **public void**enigma(View v){      String dado =  ((EditText)findViewById(R.id.txtTel)).getText().toString();        Intent intent = **new**Intent();     intent.setAction(Intent.***ACTION\_CALL***);     intent.setData(Uri.*parse*(**"tel:"**+ dado));     startActivity(intent);}    **public void**incognita(View v){      String dado =  ((EditText)findViewById(R.id.txtEnd)).getText().toString();        Uri uri = Uri.*parse*(**"http://"**+ dado);     Intent intencao = **new**Intent (Intent.***ACTION\_VIEW***, uri);     startActivity( intencao); } |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Incorreta  O método **enigma**abrirá uma tela dentro do próprio aplicativo na qual será efetuada uma ligação telefónica para um número digitado pelo usuário em um campo de texto que no arquivo .xml tem identificador txtTel.  O método **incognita** abrirá uma tela dentro do próprio aplicativo na qual será aberto o navegador em uma URL digitada pelo usuário em um campo de texto que no arquivo .xml tem identificador txtEnd. | | Resposta Correta: | Correta  O método **enigma**atende um evento no qual será efetuada uma ligação telefónica para um número digitado pelo usuário em um campo de texto que no arquivo .xml tem identificador txtTel.  O método **incognita**atende um evento no qual será aberto o navegador em uma URL digitada pelo usuário em um campo de texto que no arquivo .xml tem identificador txtEnd. | |  |  |  |

* **Pergunta 2**

0,6 em 0,6 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Nas atividades desta disciplina foi solicitado reprogramar na linguagem Kotlin aquela calculadora simples, com as quatro operações básicas:  Suponha que os id dos componentes visuais utilizados no arquivo .xml da tela têm os nomes: v1 e v2 (EditText para digitação de dados), tipos (grupo de radiobuttons), resul (TextView para mostrar o resultado) e soma, sub, mult, div (radiobuttons).  Suponha, também, que no .xml o método para atender o evento de click foi especificado com o nome 'operacoes'.  Qual seria uma codificação correta, na linguagem Kotlin, para resolver este problema? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Correta  **fun** operacoes(v: View) {  **val** op = tipos.*checkedRadioButtonId* **val** valor1 = v1.*text*.toString().*toFloat*()  **val** valor2 = v2.*text*.toString().*toFloat*()  **if** (op == R.id.*soma*) {  resul.*text* = **"Soma: "** + (valor1 + valor2)  } **else if** (op == R.id.*sub*) {  resul.*text* = **"Subtração: "** + (valor1 - valor2)  } **else if** (op == R.id.*mult*) {  resul.*text* = **"Produto: "** + (valor1 \* valor2)  } **else if** (op == R.id.*div*) {  resul.*text* = **"Divisão: "** + (valor1 / valor2)  } } | | Resposta Correta: | Correta  **fun** operacoes(v: View) {  **val** op = tipos.*checkedRadioButtonId* **val** valor1 = v1.*text*.toString().*toFloat*()  **val** valor2 = v2.*text*.toString().*toFloat*()  **if** (op == R.id.*soma*) {  resul.*text* = **"Soma: "** + (valor1 + valor2)  } **else if** (op == R.id.*sub*) {  resul.*text* = **"Subtração: "** + (valor1 - valor2)  } **else if** (op == R.id.*mult*) {  resul.*text* = **"Produto: "** + (valor1 \* valor2)  } **else if** (op == R.id.*div*) {  resul.*text* = **"Divisão: "** + (valor1 / valor2)  } } | |  |  |  |

* **Pergunta 3**

0 em 0,6 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Na pasta **raw** de um aplicativo Android está guardado o arquivo **veiculos.json**, com os dados a seguir:  {    "veiculos":    [       {"placa": "HDE-2133", "modelo": "Fiat Palio", "ano": 2001, "preco": 34000.0},       {"placa": "DEF-6566", "modelo": "Ford Ka", "ano": 2011, "preco": 54000.0},       {"placa": "DDD-0987", "modelo": "Audi A1", "ano": 2015, "preco": 90000.0},       {"placa": "FFD-2143", "modelo": "Lancer", "ano": 2016, "preco": 72300.0}    ] }    Sabemos que será necessário um método que retorne uma String a partir dos dados carregados do arquivo JSON anterior. Suponhamos que este método será chamado de **loadJSON**.  Qual seria a programação adequada do método loadJSON e de outro método **mostrar** para apresentar em um TextView com id dados, apenas os veículos do modelo Ford Ka? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Incorreta  public void **loadJSON**() {     String json = null;     try {         InputStream is = this.getResources().openRawResource(R.raw.veiculos);         int size = is.available();         byte[] buffer = new byte[size];         final int res = is.read(buffer);         is.close();         json = new String(buffer, "UTF-8");     } catch (IOException ex) {         ex.printStackTrace();         return null;     }     return json; }    public String **mostrar**() {     TextView resultados = (TextView) findViewById(R.id.dados);     String data = "";     try {         JSONObject objRaiz = new JSONObject(loadJSON());         JSONArray jsonArray = objRaiz.optJSONArray("veiculos");         for(int i=0; i < jsonArray.length(); i++){             JSONObject jsonObject = jsonArray.getJSONObject(i);             String placa = jsonObject.getString("placa");             String modelo = jsonObject.getString("modelo");             int ano = jsonObject.getInt("ano");             double preco = jsonObject.getDouble("preco");             if (modelo.equals("Ford Ka")) data += " \n" + placa + ", " + modelo + ", " + ano + ", " + preco;         }         resultados.setText(data);     } catch (JSONException e) {} } | | Resposta Correta: | Correta  public String **loadJSON**() {     String json = null;     try {         InputStream is = this.getResources().openRawResource(R.raw.veiculos);         int size = is.available();         byte[] buffer = new byte[size];         final int res = is.read(buffer);         is.close();         json = new String(buffer, "UTF-8");     } catch (IOException ex) {         ex.printStackTrace();         return null;     }     return json; }    public void **mostrar**() {     TextView resultados = (TextView) findViewById(R.id.dados);     String data = "";     try {         JSONObject objRaiz = new JSONObject(loadJSON());         JSONArray jsonArray = objRaiz.optJSONArray("veiculos");         for(int i=0; i < jsonArray.length(); i++){             JSONObject jsonObject = jsonArray.getJSONObject(i);             String placa = jsonObject.getString("placa");             String modelo = jsonObject.getString("modelo");             int ano = jsonObject.getInt("ano");             double preco = jsonObject.getDouble("preco");             if (modelo.equals("Ford Ka")) data += " \n" + placa + ", " + modelo + ", " + ano + ", " + preco;         }         resultados.setText(data);     } catch (JSONException e) {} } | |  |  |  |

* **Pergunta 4**

0,2 em 0,2 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Conhecemos que em Android podemos utilizar diferentes tipos de layouts, que definem a estratégia de colocação de componentes em uma tela. Analise as seguintes afirmações quanto ao layout linear ou LinearLayout:  I - A ordem em que os componentes são exibidos na tela é a mesma em que colocamos os componentes, no arquivo XML, dentro de um layout linear.  II - Esse tipo de layout suporta orientação vertical e horizontal.  III - É permitido colocar um ou vários layouts lineares dentro de outro.  IV - Um layout linear específico poderá especificar apenas uma orientação: horizontal ou vertical.    Marque a alternativa que você considera correta sobre o LinearLayout (layout linear) em Android. |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Correta  Todas as afirmações (I, II, III, IV) são corretas. | | Resposta Correta: | Correta  Todas as afirmações (I, II, III, IV) são corretas. | |  |  |  |

* **Pergunta 5**

0,2 em 0,2 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Para receber informações de retorno de uma Activity em Android, ou seja, para abrir uma segunda tela e receber dados da mesma, devemos iniciar essa segunda Activity com qual método? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Correta  startActivityForResult | | Resposta Correta: | Correta  startActivityForResult | |  |  |  |

* **Pergunta 6**

0,6 em 0,6 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Foi elaborado um aplicativo Android para calcular o valor final de uma compra, conhecido o preço normal da compra e considerando dois possíveis descontos: desconto de 10% se for pagamento à vista e desconto especial de R$ 5,00 (cinco Reais) se o cliente for aposentado. Veja na figura a seguir.      Para construir esta tela foi elaborado um arquivo .xml como mostrado a seguir (simplificado):       |  |  | | --- | --- | | <**EditText         android:id="@+id/edPreco"         ...         android:ems="10"         android:inputType="numberDecimal"**  />      <**RadioGroup         android:id="@+id/grAvista"         android:layout\_width="220dp"         android:layout\_height="63dp"**  **...**>       <**RadioButton             android:id="@+id/rbSemDesc"             ...             android:layout\_weight="1"             android:checked="true"             android:text="Sem desconto"**/>       <**RadioButton             android:id="@+id/rbDezPorc"             ...             android:text="Desconto de 10%**  **à vista"**/>     </**RadioGroup**> | <**CheckBox         android:id="@+id/chApos"         ...         android:text="desconto especial**  **para aposentados"**/>      <**Button         android:id="@+id/btnCalcular"         ...         android:text="Calcular"         android:onClick="calcularFinal"**/>      <**TextView         android:id="@+id/txtRes"         ...         android:text="Preço final**  **a pagar: "         android:textSize="18sp"**/> |         Em sua opinião, qual seria o método que atenderia a todos os requisitos antes apresentados? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Correta  **public void**calcularFinal(View v) {     **float**preco =            Float.*parseFloat*(((EditText)findViewById(R.id.***edPreco***)).getText().toString());     **int**op = ((RadioGroup)findViewById(R.id.***grAvista***)).getCheckedRadioButtonId();     **boolean**apo = ((CheckBox)findViewById(R.id.***chApos***)).isChecked();     **float**precofinal = preco;     **if**(op == R.id.***rbDezPorc***) precofinal = 0.9f \* precofinal;     **if**(apo) precofinal = precofinal - 5.0f;     ((TextView)findViewById(R.id.***txtRes***)).setText(**"Preço final a pagar: R$ "**  + precofinal); } | | Resposta Correta: | Correta  **public void**calcularFinal(View v) {     **float**preco =            Float.*parseFloat*(((EditText)findViewById(R.id.***edPreco***)).getText().toString());     **int**op = ((RadioGroup)findViewById(R.id.***grAvista***)).getCheckedRadioButtonId();     **boolean**apo = ((CheckBox)findViewById(R.id.***chApos***)).isChecked();     **float**precofinal = preco;     **if**(op == R.id.***rbDezPorc***) precofinal = 0.9f \* precofinal;     **if**(apo) precofinal = precofinal - 5.0f;     ((TextView)findViewById(R.id.***txtRes***)).setText(**"Preço final a pagar: R$ "**  + precofinal); } | |  |  |  |

* **Pergunta 7**

0,6 em 0,6 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Uma empresa elabora um app Android para processar os dados dos veículos que possui.  Suponhamos que temos um banco de dados **bdempresa**, com uma tabela chamada **tabveiculos** que possui os campos: **placa** (chave primária), **codigo**, **marca** e **preco**. O código é um valor inteiro e o preço do veículo é um valor real. A placa e a marca são textos.  Suponha, também, que já existe no aplicativo Android uma classe chamada MeuBancoOpenHelper, que facilita a abertura/criação do banco de dados e suas tabelas.  Em sua opinião, qual seria a lógica correta (simplificada) para eliminar um veículo específico cuja placa é FFW-7734? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Correta  MeuBancoOpenHelper ch = new MeuBancoOpenHelper (getApplicationContext());  SQLiteDatabase bd = ch.getWritableDatabase();  long res = bd.delete("tabveiculos", "placa = 'FFW-7734' " , null);  if (res == 0) {     //não foi possível eliminar o registro  }  else {    //eliminação do registro ok  } | | Resposta Correta: | Correta  MeuBancoOpenHelper ch = new MeuBancoOpenHelper (getApplicationContext());  SQLiteDatabase bd = ch.getWritableDatabase();  long res = bd.delete("tabveiculos", "placa = 'FFW-7734' " , null);  if (res == 0) {     //não foi possível eliminar o registro  }  else {    //eliminação do registro ok  } | |  |  |  |

* **Pergunta 8**

0 em 0,6 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Uma empresa elabora um app Android para processar os dados dos veículos que possui.  Suponhamos que temos um banco de dados **bdempresa**, com uma tabela chamada **tabveiculos** que possui os campos a seguir, nesta ordem: **placa** (chave primária), **codigo**, **marca** e **preco**. O código é um valor inteiro e o preço do veículo é um valor real. A placa e a marca são textos.  Suponha, também, que já existe no aplicativo Android uma classe chamada MeuBancoOpenHelper, que facilita a abertura/criação do banco de dados e suas tabelas.  Em sua opinião, qual seria a lógica correta simplificada para listar em um TextView (cujo id no arquivo .xml é txtlista) os veículos que tenham um preço inferior a R$40.000,00? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Incorreta  String str="";  MeuBancoOpenHelper ch = new MeuBancoOpenHelper (getApplicationContext());  SQLiteDatabase bd = ch.getReadableDatabase();  Cursor cursor = bd.query("tbveiculos", null, null, null, null, null, null); while (cursor.moveToNext()) {         String a = cursor.getString(0);         String b = cursor.getString(1);         String c = cursor.getString(2);         String d = cursor.getString(3);         if(Float.parseFloat(d) < 40000.0f) {            str += a + ", "  + b + ", "  + c + ", "  + d + "\n\n";          } } ((TextView)findViewById(R.id.lista)).setText(str); | | Resposta Correta: | Correta  String str="";  MeuBancoOpenHelper ch = new MeuBancoOpenHelper (getApplicationContext());  SQLiteDatabase bd = ch.getReadableDatabase();  Cursor cursor = bd.query("tabveiculos", null, null, null, null, null, null); while (cursor.moveToNext()) {         String a = cursor.getString(0);         String b = cursor.getString(1);         String c = cursor.getString(2);         String d = cursor.getString(3);         if(Float.parseFloat(d) < 40000.0f) {            str += a + ", "  + b + ", "  + c + ", "  + d + "\n\n";          } } ((TextView)findViewById(R.id.txtlista)).setText(str); | |  |  |  |

* **Pergunta 9**

0,5 em 0,5 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Um projeto Android deverá ter uma única tela para cadastrar, eliminar, consultar, mostrar os contatos cadastrados pelo aplicativo e mostrar apenas as mulheres.        Os contatos serão objetos da classe **Contato** e serão guardados em um ArrayList:    **public class MainActivity** **extends** AppCompatActivity {      **private** ArrayList **contatos** = **new** ArrayList();    A classe **Contato** terá apenas os quatro atributos mostrados a seguir e contará com os tradicionais métodos get e set, um método construtor com parâmetros e o método toString.    **public class Contato** {      **private** String **nome**, **celular**, **residencial**;  **private** char **sexo**;    Para permitir a digitação dos dados de um contato (nome, telefone celular, telefone residencial e sexo), no arquivo .xml que define o visual da tela foram especificadas as tags a seguir:    <**TextView      android:id="@+id/textn"      android:layout\_width="150dp"      android:layout\_height="wrap\_content"      android:text="Nome:"**/>  <**EditText      android:id="@+id/edNome"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"**/>    <**TextView      android:id="@+id/text1"      android:layout\_width="150dp"      android:layout\_height="wrap\_content"      android:text="Fone (cel):"** />  <**EditText      android:id="@+id/edCelular"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:inputType="phone"** />    <**TextView      android:id="@+id/text2"      android:layout\_width="150dp"      android:layout\_height="wrap\_content"      android:text="Fone (res):"** />  <**EditText      android:id="@+id/edResid"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:inputType="number|phone"** />    <**TextView      android:id="@+id/text3"      android:layout\_width="150dp"      android:layout\_height="wrap\_content"      android:text="Sexo:"** />  <**EditText      android:id="@+id/edSexo"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"** />    Os cinco botões mostrados no visual da tela principal foram definidos no arquivo .xml com as tags a seguir:    <**Button      android:id="@+id/btn1"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="cadastrar"      android:text="Cadastrar"** />  <**Button      android:id="@+id/btn2"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="eliminar"      android:text="Eliminar"** />  <**Button      android:id="@+id/btn3"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="consultar"      android:text="Consultar"** />  <**Button      android:id="@+id/btn4"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="mostrar"      android:text="Mostrar"** />  <**Button      android:id="@+id/btn5"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="mulheres"      android:text="Mulheres"** />    Para mostrar os dados dos contatos cadastrados (e outros resultados) foi definido um rótulo de texto no final da tela, no arquivo .xml:    <**TextView      android:id="@+id/txtRes"      android:layout\_width="match\_parent"      android:layout\_height="191dp"** />  Qual é a lógica correta do botão Mulheres, que permita mostrar no objeto TextView, que se encontra no final da tela, todas as mulheres cadastradas? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Correta  **public void** mulheres(View v) {  String str = **"\nMulheres cadastradas \n\n"**;  **for** (**int** i=0; i < **contatos**.size(); i++){  Contato cont = (Contato)**contatos**.get(i);  **if**(cont.getSexo() == **'F'**) {  str += cont.toString() + **"\n"**;  }  }  ((TextView)findViewById(R.id.***txtRes***)).setText(str); } | | Resposta Correta: | Correta  **public void** mulheres(View v) {  String str = **"\nMulheres cadastradas \n\n"**;  **for** (**int** i=0; i < **contatos**.size(); i++){  Contato cont = (Contato)**contatos**.get(i);  **if**(cont.getSexo() == **'F'**) {  str += cont.toString() + **"\n"**;  }  }  ((TextView)findViewById(R.id.***txtRes***)).setText(str); } | |  |  |  |

* **Pergunta 10**

0,5 em 0,5 pontos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
|  | Um projeto Android deverá ter uma única tela para cadastrar, eliminar, consultar e mostrar os contatos cadastrados pelo aplicativo.        Os contatos serão objetos da classe **Contato** e serão guardados em um ArrayList:    **public class MainActivity** **extends** AppCompatActivity {       **private** ArrayList **contatos** = **new** ArrayList();    A classe **Contato** terá apenas os três atributos mostrados a seguir e contará com os tradicionais métodos get e set, um método construtor com parâmetros e o método toString.    **public class Contato** {      **private** String **nome**, **celular**, **residencial**;    Para permitir a digitação dos dados de um contato (nome, telefone celular e telefone residencial), no arquivo .xml que define o visual da tela foram especificadas as tags a seguir:    <**TextView      android:id="@+id/textn"      android:layout\_width="150dp"      android:layout\_height="wrap\_content"      android:text="Nome:"**/>  <**EditText      android:id="@+id/edNome"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"**/>    <**TextView      android:id="@+id/text1"      android:layout\_width="150dp"      android:layout\_height="wrap\_content"      android:text="Fone (cel):"** />  <**EditText      android:id="@+id/edCelular"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:inputType="phone"** />    <**TextView      android:id="@+id/text2"      android:layout\_width="150dp"      android:layout\_height="wrap\_content"      android:text="Fone (res):"** />   <**EditText      android:id="@+id/edResid"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:inputType="number|phone"** />    Os quarto botões mostrados no visual da tela principal foram definidos no arquivo .xml com as tags a seguir:    <**Button      android:id="@+id/btn1"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="cadastrar"      android:text="Cadastrar"** />  <**Button      android:id="@+id/btn2"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="eliminar"      android:text="Eliminar"** />  <**Button      android:id="@+id/btn3"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="consultar"      android:text="Consultar"** />  <**Button      android:id="@+id/btn4"      android:layout\_width="wrap\_content"      android:layout\_height="wrap\_content"      android:onClick="mostrar"      android:text="Mostrar"** />    Para mostrar os dados de todos os contatos cadastrados foi definido um campo de texto no final da tela, no arquivo .xml:      <**TextView      android:id="@+id/txtRes"      android:layout\_width="match\_parent"      android:layout\_height="191dp"** />      Considerando todos os elementos anteriores, qual é, em sua opinião, a lógica correta para **procurar** um contato cujo nome foi digitado pelo usuário, apresentar seus restantes dados nos campos de texto ou apresentar uma mensagem caso esse nome não for encontrado? |  |  |  |
| |  |  | | --- | --- | | Resposta Selecionada: | Correta  **public void** consultar(View v) {  TextView tc = (TextView)findViewById(R.id.***edCelular***);  TextView tr = (TextView)findViewById(R.id.***edResid***);  String nom = ((EditText)findViewById(R.id.***edNome***)).getText().toString();  **for** (**int** i=0; i < **contatos**.size(); i++){  **if**(((Contato)**contatos**.get(i)).getNome().equalsIgnoreCase(nom)) {  tc.setText(((Contato)**contatos**.get(i)).getCelular());  tr.setText(((Contato)**contatos**.get(i)).getResidencial());  **return**;  }  }  Toast.*makeText*(**this**,**"\n Esse contato não existe!\n"**,Toast.***LENGTH\_LONG***).show(); } | | Resposta Correta: | Correta  **public void** consultar(View v) {  TextView tc = (TextView)findViewById(R.id.***edCelular***);  TextView tr = (TextView)findViewById(R.id.***edResid***);  String nom = ((EditText)findViewById(R.id.***edNome***)).getText().toString();  **for** (**int** i=0; i < **contatos**.size(); i++){  **if**(((Contato)**contatos**.get(i)).getNome().equalsIgnoreCase(nom)) {  tc.setText(((Contato)**contatos**.get(i)).getCelular());  tr.setText(((Contato)**contatos**.get(i)).getResidencial());  **return**;  }  }  Toast.*makeText*(**this**,**"\n Esse contato não existe!\n"**,Toast.***LENGTH\_LONG***).show(); } | |  |  |  |