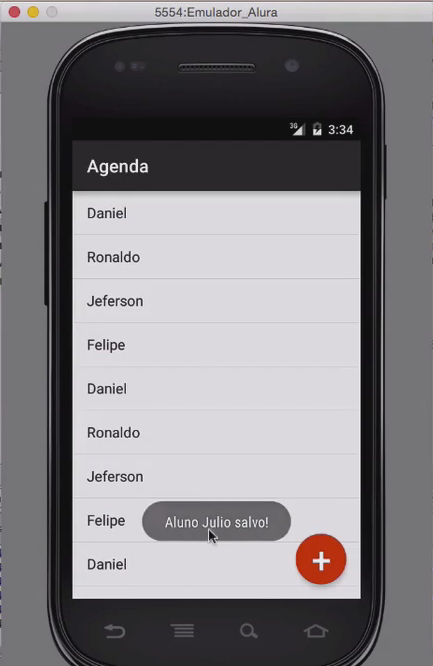
Adicionando alunos a nossa lista

Já conseguimos armazenar informações que são preenchidas no formulário e também já conseguimos jogar isso no objeto "Aluno". Assim, quando acrescentarmos um novo nome de aluno no formulário e damos um "check" o aplicativo vai reconhecer e dar uma resposta. Por exemplo, preenchemos o formulário com o aluno "Paulo", o celular vai nos responder após clicarmos no símbolo de "check", "O aluno Paulo foi salvo!".



Agora, queremos que todos os novos formulários que criarmos apareçam de fato em nossa lista. Isto é, que o nome de "Paulo" esteja junto aos demais!

Agora, vamos começar a abrir esse caminho para um banco de dados, que é o que vai possibilitar que isso seja feito.

Para fazer isso poderíamos modificar nossa *Array* na ListaAlunosActivity.java. Mas se fizermos isso "a mão", sempre que desligarmos o celular qualquer novo aluno que cadastrarmos vai ser perdido.

Se tivéssemos as informações dos alunos em um arquivo ou banco de dados e a *Array* populasse isso dessas fontes, não perderíamos os alunos toda vez que a aplicação reiniciar.

Para não perder mais as informações, vamos começar a colocar persistência no aplicativo.

Podemos inserir uma conexão para se ligar ao banco de dados tanto na ListaAlunosActivity.java quanto no FormulárioActivity.java, mas as conexões nem sempre são códigos simples, a medida que formos acrescentando novas informações ele aumentaria e teríamos que espalhá-lo em diversos locais, o que complicaria um pouco as coisas.

Então, vamos fazer o que aprendemos anteriormente, que é isolar toda a parte da conexão em uma classe específica. Assim, sempre que quisermos manipular algo que diga respeito a um aluno poderemos usar uma classe que nos ajudará a fazer as modificações necessárias. Ela também irá abstrair toda a parte de acesso ao banco de dados, operações, conexão e etc.

O nome do objeto que abstrai tudo isso que acabamos de comentar é "DAO".

Vamos criar uma classe nova no *Android*, para isso, vamos na pasta "br.com.alura", clicamos com o botão direito e "New> Java Class". Vai abrir uma janela pedindo um *Name*, vamos nomear de "AlunoDAO", vamos dar um "Enter" e pronto, criou o "AlunoDAO.java".

Mas, queremos colocar ele em outro pacote, para isso vamos de novo em "br.com.alura" e clicamos com o botão direito, selecionamos "New> Package". Vai abrir um a janela pedindo para que digitemos o *Package name*. O nome do pacote que vamos criar será "dao". Agora ele vai criar ali no canto esquerdo o "dao", dentro de "br.com.alura" . Para trocar o "AlunoDAO" de lugar basta selecionar ele e arrastar para o novo *package*. Vai abrir uma janela nova e é só pedir para refaturar o código selecionando o "Refactor".

A nova classe, "AlunoDAO.java" estará da seguinte maneira:

Dentro da classe que acabamos de criar vamos fazer a parte de conexão de bancos de dados. Imagine que a medida que o tempo for passando nós queiramos introduzir uma nova coluna na nossa lista, com por exemplo, a foto dos alunos, teríamos que fazer diversas verificações para que tudo ocorresse bem! Teríamos que pensar no caso de como o *Android* avisaria as pessoas que já tinham a aplicação baixada para atualizá-la e etc...

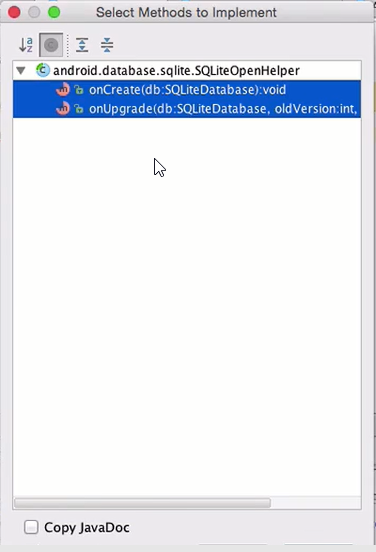
As coisas ficariam complicadas! Assim, vamos facilitar um pouco utilizando uma classe do *Android* que faz tudo isso por nós.

Nossa aba 'AlunoDAO.java' está como a figura de cima mostra!

O banco de dados que vamos utilizar no *Android* é o SQLite, uma versão simplificada do sql. Então, vamos transformar nossa classe em um SQLiteOpenHelper, para fazer isso digitaremos SQLiteOpenHelper na mesma linha do public class AlunoDAO. Vamos importar o SQLiteOpenHelper usando em cima dela o atalho "Alt+Enter".

Essa classe vai nos ajudar em todas as operações relativas a banco de dados.

Mas, perceba que o que acabamos de digitar estará em vermelho, pois, ainda resta um pequeno erro. O *Android* ainda requer um construtor uma vez que o SQLiteOpenHelperpede, obrigatoriamente, alguns parâmetros que são fornecidos, justamente, pelo construtor. Clicando em cima do SQLiteOpenHelper podemos dar um "Alt+Enter" e pedir para que ele "Implement methods". Na hora que dermos um "Enter" ele vai pedir para que implementemos dois métodos, que são obrigatórios, o onCreatee o onUpgrade. Basta dar um "Ok".

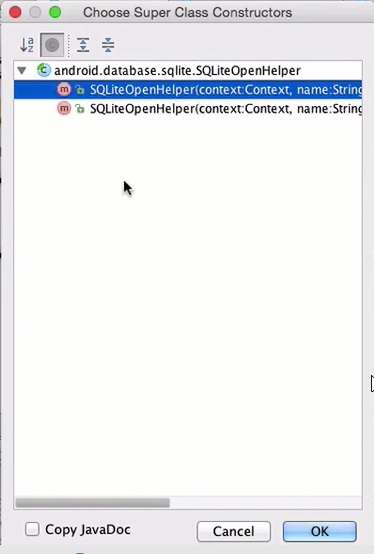


Nossa tela ficará da seguinte maneira:

public class AlunoDAO extends SQLiteOpenHelper{  
 @Override  
 public void onCreate (SQLiteDatabase db) {   
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int, oldVersion, int newVersion) {   
 }  
}

Só que, ao fazer isso, o SQLiteOpenHelper continuará em vermelho, agora damos mais um "Alt+Enter" e pedimos para ele gerar o seguinte construtor: Create constructor matching super. Uma nova janela será aberta e ele vai sugerir dois construtores, nós vamos utilizar apenas o primeiro que possui quatro parâmetros: !(Super Class Constructor)["<https://s3.amazonaws.com/caelum-online-public/Android+III/Android+I+Super+class+Constructor.png>"]

Selecionamos apenas ele e damos "Ok".



O que ele irá gerar é um construtor automático:

Context context, String name, SQLiteDatabase.CursorFactory factory, int version

O trecho do nosso código ficará assim:

public class AlunoDAO extends SQLiteOpenHelper{  
 public AlunoDAO (Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {  
 super(context, name, factory, version);  
 }  
  
 @Override  
 public void onCreate (SQLiteDatabase db) {   
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {   
 }   
}

Veremos o que está sendo pedido no 'super'. Primeiramente , o construtor pediu um contexto, que é a nossa identificação para o *java*. Ele vai pedir em seguida um *name*, mas com isso não precisaremos nos preocupar, temos a opção de usar qualquer nome. Por exemplo, iremos substituir name, a palavra "Agenda". Com isso poderemos tirar o primeiro parâmetro que colocamos no AlunoDAO, apagando o String name.

Ele irá pedir um factory, se quisermos customizar alguns parâmetros, mas como nem sempre precisamos fazer essas modificações, vamos apagar e substituir o factory por um null e apagaremos, por consequência o SQLiteDatabase.CursorFactory factory.

Por fim, ele perguntará qual a versão do nosso banco de dados, falaremos que é a primeira, e por isso escreveremos apenas um 1 e assim, podemos apagar também o int version. No super ficaremos apenas com context, "Agenda", null, 1 entre os parênteses e no public apenas com Context context. Agora temos tudo para que o construtor possa funcionar.

Ficaremos no todo com:

public class AlunoDAO extends SQLiteOpenHelper {  
 public class AlunoDAO (Context context) {  
 super(context, "Agenda", null, 1);   
 }  
  
 @Override  
 public void onCreate (SQLiteDatabase db) {   
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {   
 }   
}

Perceba que temos ainda dois métodos embaixo. O primeiro é o 'onCreate', que é usado assim que o banco de dados é criado.

Para introduzir algo dentro do banco de dados, a primeira coisa a fazer é colocar uma tabela, para isso, vamos manipular o banco de dados para criar uma tabela. Para criar uma tabela em sql vamos introduzir uma instrução, uma String sql, um comando para criar a tabela, o CREATE TABLE, e um nome para ela, Alunos, e descreveremos entre os parênteses os nomes das colunas da tabela.

Colocaremos na primeira coluna um número de identificação,id INTEGER PRIMARY KEY e em seguida, colocaremos todos os campos do formulário, o nome, o endereço, o telefone, o site e a nota. O dado de nome será acompanhado de TEXT pois ele é um texto e NOT NULL, pois não pode ser um campo sem nada. Aos demais campos acrescentaremos que são apenas do tipo TEXT e a nota é do tipo REAL. Vamos fechar o parênteses e colocar um ; para dizer que nossa instrução acabou e mais um ; devido a linha java. Ficaremos com:

public class AlunoDAO extends SQLiteOpenHelper {  
 public class AlunoDAO (Context context) {  
 super(context, "Agenda", null, 1);   
 }  
  
 @Override  
 public void onCreate (SQLiteDatabase db) {  
 String sql = "CREATE TABLE Alunos (id INTEGER PRIMARY KEY, nome TEXT NOT NULL, endereço TEXT, telefone TEXT, site TEXT, nota REAL);";   
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {   
 }  
}

Depois da linha da String, damos um "Enter" e na próxima linha, para poder executar essas instruções, vamos acrescentar o seguinte db.execSQL(sql);, usando a variável db que o *Android* passa como parâmetro. Vamos ver como fica nosso método:

@Override  
 public void onCreate (SQLiteDatabase db) {  
 String sql = "CREATE TABLE Alunos (id INTEGER PRIMARY KEY, nome TEXT NOT NULL, endereço TEXT, telefone TEXT, site TEXT, nota REAL);";  
 db.execSQL(sql);  
 }

Vamos agora para o próximo método, o onUpgrade. Lembre-se que podemos ter duas situações, ou um novo banco precisa ser criado ou ele precisa ser atualizado. Ele vai saber disso a partir do parâmetro que informamos no construtor, a versão. Toda vez que mudarmos algo no nosso banco de dados vamos ter que alterar a versão, por 2, 3, 4...

Uma vez que o banco de dados é alterado e modificamos o número da versão é o onUpgrade que avisa que ela foi modificada e precisa ser atualizada. Assim, podemos adicionar algo que faça passar da versão 1 para a versão 2 e que avise o que precisa ser adicionado, por exemplo, um novo caminho para introduzir fotos dos alunos. Como não temos uma atualização do banco de dados não precisaríamos introduzir nada mas, vamos introduzir um caminho para facilitar isso no futuro.

Vamos adicionar uma instrução para jogar fora a tabela, digitaremos o comando DROP TABLE IF EXISTS e o nome da tabela, no caso, "Alunos". Teremos, String sql = DROP TABLE IF EXISTS Alunos. Vamos pedir também para que ela seja executada, através de db.execSQL(sql).

Por que estamos fazendo isso?

Imagine que tivéssemos escrito o método onCreate errado, ao em vez de "nome", tivéssemos digitado "nom" e tivéssemos subido a aplicação. Essa seria nossa versão 1 e ao perceber o erro, voltaríamos para alterar isso. Quando arrumarmos o *Android* não vai chamar nem o onCreate, nem o onUpgradepois a aplicação não mudou sua versão. Como erros como esse são comuns é bom deixar a instrução DROP TABLE IF EXISTS, pois assim a tabela velha será substituída pela arrumada.

Também vamos pedir, na linha de baixo de db.execSQL(sql) para ele chamar o onCreate de novo, utilizando o onCreate(db). Sempre que fizermos alguma modificação, vamos alterar a versão (no caso o número 1 por uma versão 2 ou 3 e por aí em diante) e a tabela antiga vai ser "jogada fora" e substituída por uma versão mais atualizada.

Ficaremos com o seguinte método:

@Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 String sql = "DROP TABLE IF EXISTS Alunos";  
 db.execSQL(sql);  
 onCreate(db);   
 }

Por enquanto é isso que faremos! Estamos apenas preparando caminho para que o nosso banco de dados comece a salvar os alunos. Na tela final ficaremos com:

public class AlunoDAO extends SQLiteOpenHelper {  
 public class AlunoDAO (Context context) {  
 super(context, "Agenda", null, 1);   
 }  
  
 @Override  
 public void onCreate (SQLiteDatabase db) {  
 String sql = "CREATE TABLE Alunos (id INTEGER PRIMARY KEY, nome TEXT NOT NULL, endereço TEXT, telefone TEXT, site TEXT, nota REAL);";  
 db.execSQL(sql);   
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 String sql = "DROP TABLE IF EXISTS Alunos";  
 db.execSQL(sql);  
 onCreate(db);   
 }  
}