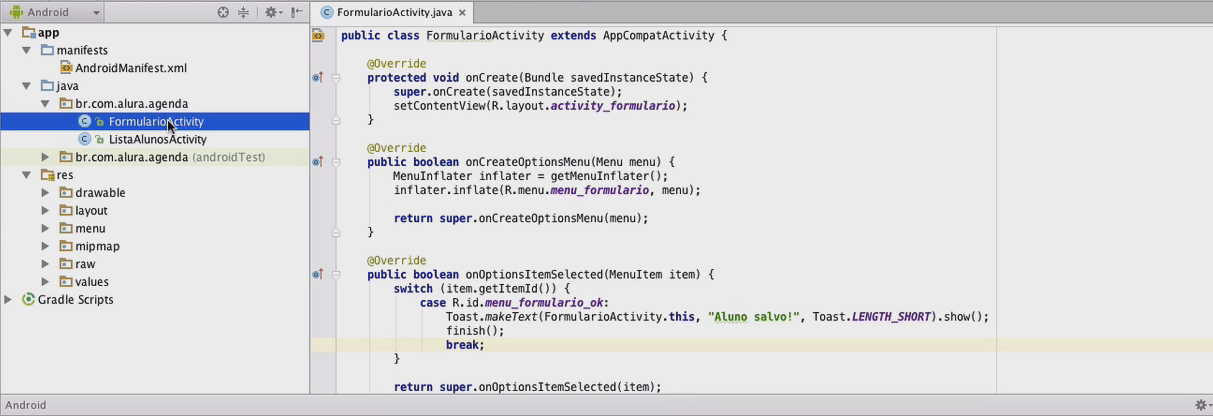
Recuperando dados do aluno

Nas aulas anteriores introduzimos novos botões! Agora, nós desejamos guardar de fato o nome dos alunos que são acrescentados.

Para isso, realizaremos alterações no arquivo, Formulário Activity. Lembrando que nossa tela está da seguinte maneira:



***Lembrando***

O *Android* avisa qual item do menu é clicado através do onOptionsItemSelected. Na *Override* onOptionsItemSelected acrescentamos também um caso, o Toast para que uma mensagem aparecesse quando o botão fosse clicado.

Bom, para salvar novos alunos, o que queremos na verdade, é recuperar os dados novos que inserirmos. Então, vamos inserir algumas coisas na *Override* da onOptionsItemSelected. Temos:

@Override   
 public boolean onOptionsItemSelected(MenuItem item) {   
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH\_SHORT).show();  
  
 EditText campoNome = (EditText) findViewById(R.Id.formulario\_nome);  
 String nome = campoNome.getText().toString();  
  
 finish();  
 break;  
 }  
  
 return super.onOptionsItemSelected(item);   
 }

Damos um enter e uma linha abaixo de 'Toast' iremos acrescentar algumas informações. Como necessitamos armazenar os dados que introduzimos no formulário. A referência vai ser buscada através do findViewById, utilizaremos entre os parênteses a classe R, ".", Id e o nome da tela, que é formulario\_nome que indica quem está sendo procurado. Ao digitar a palavra "formulário" já serão mostradas algumas opções. Ficaremos com, findViewById(R.Id.formulario\_nome).

Como esse componente é do tipo EditText, temos que adicionar EditText e o campoNome, ficaremos com, EditText campoNome = findViewById(R.Id.formulario\_nome).

Vamos em cima do "campoNome" e para importar a classe utilizamos o atalho "Alt+Enter". O *Android* vai pedir para fazermos o *Cast*, então, damos um "Alt+Enter" de novo e pronto.

Ficaremos com EditText campoNome = (EditText) findViewById(R.Id.formulario\_nome). Bom, o que fizemos foi pegar o primeiro campo, do nome, e agora, queremos pegar o valor desse "campoNome", damos mais um enter e para isso acrescentamos campoNome, "." e getText. Adicionaremos também uma String na frente disso, seguida de "nome" e um "=". Teremos, String nome = campoNome.getText.

Agora, se fizermos apenas isso ele irá devolver um *editable*, e não queremos isso. Então, temos que adicionar depois do getText mais um "." e toString. Fazemos isso para converter esse campo em algo que possa ser colocado de fato na String. Teremos, String nome = campoNome.getText().toString()

Ficaremos com :

@Override   
 public boolean onOptionsItemSelected(MenuItem item) {   
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH\_SHORT).show();  
  
 EditText campoNome = (EditText) findViewById(R.Id.formulario\_nome);  
 String nome = campoNome.getText().toString();  
  
 finish();  
 break;  
 }  
  
 return super.onOptionsItemSelected(item);   
 }

Fizemos isso com o campo "nome" e agora iremos fazer o mesmo com o campo endereço. Para isso, daremos um enter depois do que acabamos de acrescentar e na próxima linha teremos que pegar novamente a referência do campo, no caso, o Id de endereço. Digitaremos, findViewById(R.id.formulario\_endereço) e como também é um EditText, adicionaremos isso e campo endereço. Na sequência, importaremos e faremos o *Cast*. Por fim, teremos EditText campoEndereço = campoEndereço findViewById(R.id.formulario\_endereço). Na próxima linha adicionaremos oString, endereco',=, 'campoEndereco', ogetText, que serve para pegar o texto e oto String`. Teremos:

EditText campoEndereco = (EditText) findViewById(R.Id.formulario\_Endereco);  
String Endereco = campoEndereco.getText().toString();

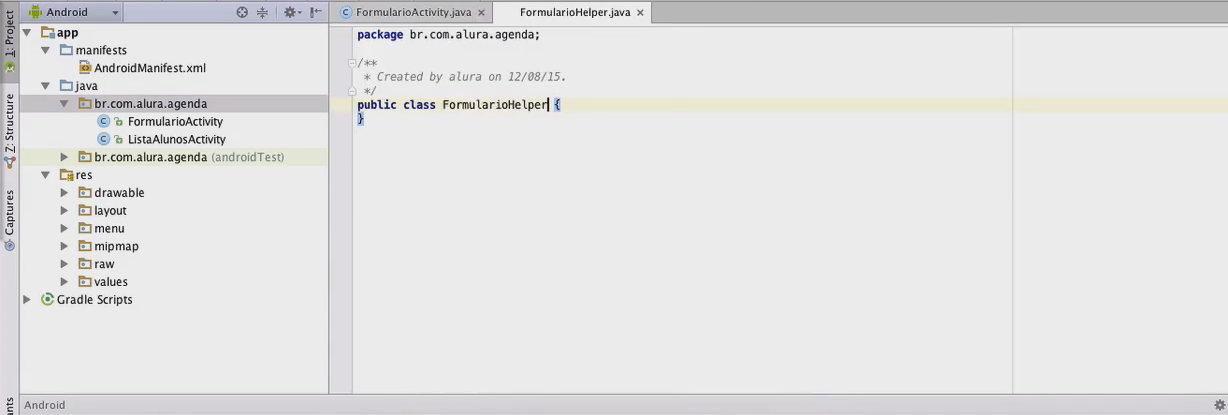
Faremos o mesmo nos campos de telefone e site, basta selecionar o que acabamos de escrever, do EditText do nome até o String do Endereço, dar um "Comand+C" ou "Ctrl+C" e "Comand+V" ou "Ctrl+V". Colaremos isso abaixo do Endereço. Basta alterar os nomes pelos campos que desejamos, no caso "Site" e "Telefone". Teremos:

@Override   
 public boolean onOptionsItemSelected(MenuItem item) {   
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH\_SHORT).show();  
 EditText campoNome = (EditText) findViewById(R.Id.formulario\_nome);  
 String nome = campoNome.getText().toString();  
  
 EditText campoEndereco = (EditText) findViewById(R.Id.formulario\_Endereco);  
 String Endereco = campoEndereco.getText().toString();  
  
 EditText campoTelefone = (EditText) findViewById(R.Id.formulario\_Telefone);  
 String Telefone = campoTelefone.getText().toString();  
  
 EditText campoSite = (EditText) findViewById(R.Id.formulario\_Site);  
 String Site = campoSite.getText().toString();  
  
 finish();  
 break;   
 }  
  
 return super.onOptionsItemSelected(item);   
 }

Imagina que a gente tivesse mantido o antigo botão de salvar, aquele que ficava na parte de baixo do formulário. Teríamos que repetir esse código todo que acabamos de escrever se quiséssemos que ele tivesse a mesma função. Quando temos que repetir muitas vezes o mesmo código, podemos criar uma classe, para que ela faça isso por nós. Vamos deslocar esses códigos para uma classe auxiliar do nosso formulário.

Para criar uma nova classe vamos no pacote "br.com.alura", clicamos com o botão direito e "New > Java Class". Vai aparecer uma janela com um campo *Name* para ser preenchido, nomearemos nossa classe de "FormularioHelper".

Nossa tela com a nova classe aparecerá da seguinte maneira,



Vamos fazer todos os findViewById que estão lá na aba FormularioActivity.java na nossa nova classe, justamente, para facilitar. Assim, não precisaremos ficar recriando eles sempre que precisarmos. Vamos introduzir os findViewById no início, no construtor do nosso "FormularioHelper".

Para fazer o construtor basta darmos um enter depois do public class FormularioHelperque já aparece na tela e escrevermos o public FormularioHelper. Agora, dentro, iremos adicionar os findViewById.

Na próxima linha adicionaremos o findViewById e digitaremos o campo que queremos puxar, no caso, o formulario\_nome. Ficaremos com, findViewById(R.Id.formulario\_nome). Repare que o findViewById ficará em vermelho, pois não é possível localizar o método.

O findViewById é um método oriundo da AppCompatActivity, e quando estamos dentro da activity podemos chamar o método normalmente. Mas, estamos no 'FormularioHelper.java'. Para utilizar o findViewById no 'FormularioHelper.java' necessitamos de uma referência da nossa activity. Vamos adicionar essa referência no construtor da nossa classe, adicionaremos entre os parênteses que seguem o public formularioHelper a referência de que estamos trazendo isso da FormularioActivity e nomearemos ela de "activity". Ficaremos com FormularioHelper(FormularioActivity activity).

Como essa referência vem de fora do FormularioHelper.java temos que referenciar também o findViewById. Digitaremos na frente de tudo activity e um ponto, teremos, activity.findViewById(R.Id.formulario\_nome).

Falta acrescentar que é um EditText do campoNome que será = a activity.findViewById(R.Id.formulario\_nome). Teremos, EditText campoNome = activity.findViewById(R.Id.formulario\_nome). Damos um "Alt+Enter" para importar e fazemos também o *Cast*.

Vamos ter o seguinte:

public class FormularioHelper {  
  
 public FormularioHelper(FormularioActivity activity) {  
 EditText campoNome = (EditText) activity.findViewById(R.Id.formulario\_nome);   
 }  
}

Agora, temos o campo do nome pronto!

Basta agora reproduzir o que já escrevemos. Selecionamos com o mouse, 'EditText campoNome = (EditText) activity.findViewById(R.Id.formulario\_nome)' e damos um "Command+C" e "Command+V". Agora, trocaremos apenas os nomes do EditText e alteraremos os Id.

Acrescentaremos um último campo, que é o da nota. Atenção aqui, ele não é um EditView, ele é um RatingBar,então, temos que alterar os tipos das variáveis. Ficaremos com RatingBar campoNota =findViewById(R.Id.formulario\_nota). Não esquecendo de importar a classe com o "Alt+Enter".

E ficaremos com:

public class FormularioHelper {  
  
 public FormularioHelper(FormularioActivity activity) {   
 EditText campoNome = (EditText) activity.findViewById(R.id.formulario\_nome);  
 EditText campoEndereco = (EditText) activity.findViewById(R.id.formulario\_endereco);  
 EditText campoTelefone = (EditText) activity.findViewById(R.id.formulario\_telefone);  
 EditText campoSite = (EditText) activity.findViewById(R.id.formulario\_site);  
 RatingBar campoNota = (RatingBar) activity.findViewById(R.id.formulario\_nota);   
 }  
}

Agora, lá na aba FormularioActivity.java podemos apagar os EditText que tínhamos. Poderemos apagar o seguinte:

EditText campoNome = (EditText) findViewById(R.Id.formulario\_nome);  
String nome = campoNome.getText().toString();   
EditText campoEndereco = (EditText) findViewById(R.Id.formulario\_Endereco);  
String endereco = campoEndereco.getText().toString();   
EditText campoTelefone = (EditText) findViewById(R.Id.formulario\_Telefone);  
String telefone = campoTelefone.getText().toString();   
EditText campoSite = (EditText) findViewById(R.Id.formulario\_Site);  
String site = campoSite.getText().toString();

E ficaremos apenas com:

@Override   
 public boolean onOptionsItemSelected(MenuItem item) {   
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Toast.makeText(FormularioActivity.this, "Aluno salvo!" , Toast.LENGTH\_SHORT).show();  
  
  
 finish();  
 break;  
 }  
  
 return super.onOptionsItemSelected(item);   
 }

Apagamos e, agora, iremos substituir. Quando a gente for buscar o item dentro da classe ItemSelected, iremos introduzir a classe do FormularioHelper. Vamos instanciar essa classe em algum lugar, por exemplo, no onCreate. Então, na onCreateadicionamos, helper, = e new FormularioHelper. Ficaremos com helper = new FormularioHelper. Só que ele ainda pede uma referência para a activity, então, digitaremos o this entre os parênteses, como parâmetro. Ficaremos com FormularioHelper helper = new FormularioHelper (this).

Mas, se adicionamos no Toast mais adiante o helper ele não vai encontrar, por isso, apagamos o FormularioHelper que tínhamos escrito e vamos transformar isso em um atributo.

Repare que o helper ficou em vermelho. Então, damos um "Alt+Enter" e escolhemos "Create final helper". O *Android* irá criar, a cima da Override um private FormularioHelper helper.

E no todo, teremos:

private FormularioHelper helper;  
  
 @Override   
 protected void onCreate(Bundle savedInstanceState) {   
 super.onCreate(savedInstanceState);   
 setContentView(R.layout.activity\_formulario);  
  
 helper = new FormularioHelper (this);   
 }

E quando quisermos utilizar o helper, vamos lá embaixo, na classe onOptionsItemSelected e depois do Toast, na próxima linha, acrescentamos helper e damos um .. Se não tivéssemos criado o atributo na onCreate não iríamos conseguir localizar o helper e ele continuaria em vermelho.

O que queremos fazer a partir desse Helper é pegar os dados do aluno. Para isso, poderíamos adicionar uma String. Teríamos String nome = helper.peganome e faríamos isso com todos os outros campos. Mas, não é muito prático.

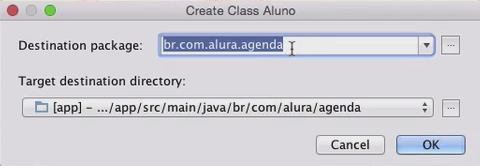
Queremos através do helper, sinalizar todos os campos que aparecem no formulário. Os campos da tela do celular, o nome, o endereço, a nota, o site e o telefone representam um "aluno" no sistema. O ideal é que tivéssemos um único método no helper, que seria String nome = helper.pegaAluno, que já traria todos os dados que queremos. Mas, no lugar de String devemos ter um objeto de tipo Aluno. Ficaremos com Aluno aluno = helper.pegaAluno.

Teremos:

@Override   
 public boolean onOptionsItemSelected(MenuItem item) {   
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH\_SHORT).show();  
 Aluno aluno = helper.pegaAluno();  
  
 finish();  
 break;  
 }  
  
 return super.onOptionsItemSelected(item);   
 }

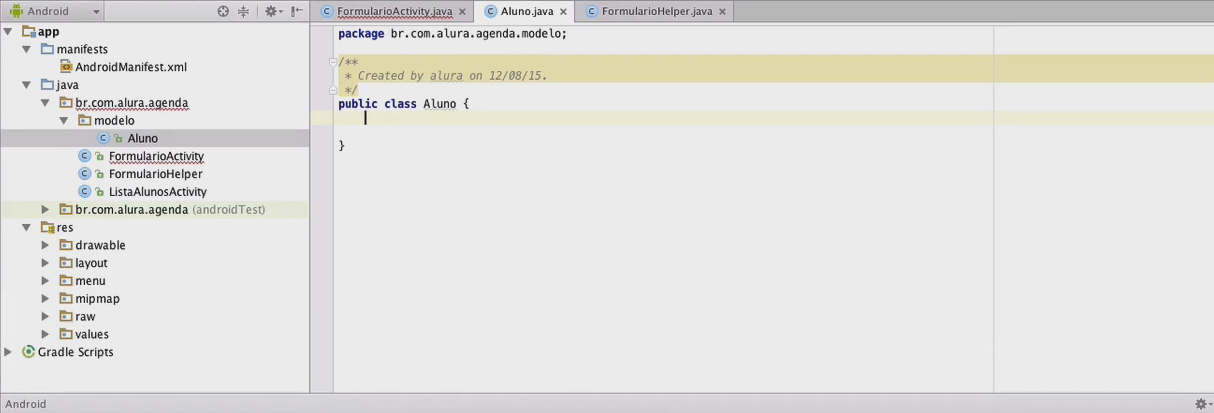
Vamos fazer um "Alt+Enter" no objeto Aluno. O *Android Studio* sugere que nós criemos a classe alunos, "Create class Aluno". Vamos selecionar essa opção e criar a classe.

Irá abrir uma janela, que tem um campo Destination package, vamos criar em um pacote que já conhecemos, vamos chamá-lo de modelo, br.com.alura.agenda.modelo. Que é o modelo da nossa lógica de negócio da aplicação e o aluno passará a representar um dos domínios. Damos um "Ok".



Ele já vai criar uma pasta chamada "modelo", ela estará junto com as outras no lado direito da tela. Nela já teremos a classe "Aluno".

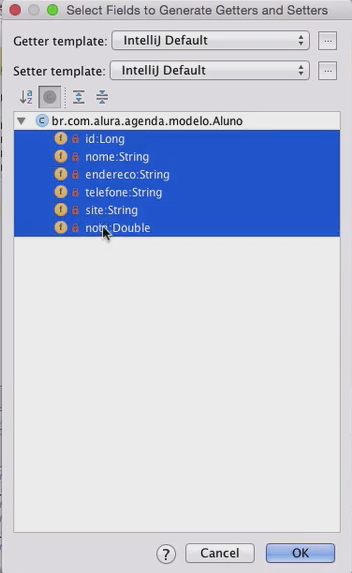
Dentro do 'Aluno.java' colocaremos todos os dados que nosso aluno precisa ter. Antes, vamos dar uma olhada de como é a tela:



Vamos acrescentar abaixo do public class, na próxima linha, private String nomee faremos a mesma coisa com os outros campos que queremos trazer para o Aluno.java. A nota, entretanto, será Double e não String. E, como, logo menos, começaremos a trabalhar com banco de dados, vamos acrescentar também um Id para esse aluno e já deixar isso para o futuro. Logo na linha de baixo de public classdigitamos private Long id. Teremos:

public class Aluno {  
 private Long id;  
 private String nome;  
 private String endereco  
 private String telefone;  
 private String site;  
 private Double nota;  
}

Como colocamos que todos os nossos objetos estão privados, teremos que criar um Getter and Setter. Damos uns dois "Enter" depois da última linha e no *Mac* utilizamos o atalho "Command+M" e no *Windows* "Alt+Insert" e selecionamos o "Getter and Setter". Vai abrir um janela e nela selecionaremos todos os campos que gostaria que ele gerasse um "Getter" e um "Setter" e damos um "Ok".



Automaticamente nossa classe terá vários "Getter" e "Setter" definidos



Agora que a classe "Aluno" está certa, vamos voltar no FormularioActivity. Veja que no Helper que acrescentamos o pegaAluno, este ainda está em vermelho, falta acrescentar um método para ele. Para isso, damos um "Alt+Enter" em cima de do 'pegaAluno' e selecionamos "Create method 'pegaAluno'". Pronto, agora temos um public Aluno pegaAluno. Ficaremos com:

public class FormularioHelper {  
  
 public FormularioHelper(FormularioActivity activity) {  
 EditText campoNome = (EditText) activity.findViewById(R.Id.formulario\_nome);  
 EditText campoEndereco = (EditText) activity.findViewById(R.Id.formulario\_endereco);  
 EditText campoTelefone = (EditText) activity.findViewById(R.Id.formulario\_telefone);  
 EditText campoSite = (EditText) activity.findViewById(R.Id.formulario\_site);  
 RatingBar campoNota = (RatingBar) activity.findViewById(R.Id.formulario\_nota);  
 }  
  
 public Aluno pegaAluno() {  
 return null;  
 }  
}

Podemos introduzir abaixo da classe pegaAluno, um aluno, introduzindo o seguinte, Aluno aluno = new Aluno. Agora, na linha de baixo digitamos, ainda, aluno.setNome. O nome do aluno vêm do formulário, então acrescentamos um, campoNome, . getText. Ficaremos com aluno.setNome(campoNome.getText).

Mas, o campo nome ficará em vermelho, isso ocorre por que antes declaramos que ele era um EditText, para resolver esse problema basta apagar o EditText de todos os campos da classe FormularioHelper, deixaremos apenas o uso deles:

public class FormularioHelper {  
  
 public class FormularioHelper(FormularioActivity activity) {  
 campoNome = (EditText) activity.findViewById(R.Id.formulario\_nome);  
 campoEndereco = (EditText) activity.findViewById(R.Id.formulario\_endereco);  
 campoTelefone = (EditText) activity.findViewById(R.Id.formulario\_telefone);  
 campoSite = (EditText) activity.findViewById(R.Id.formulario\_site);  
 campoNota = (RatingBar) activity.findViewById(R.Id.formulario\_nota);  
 }  
  
 public Aluno pegaAluno() {  
 Aluno aluno = new Aluno ();  
 aluno.setNome(campoNome.getText());  
  
 return null;  
 }   
}

Agora, nenhum deles está definido, então, em todos os campos faremos "Alt+Enter" e criaremos "Fields". Agora, todos os campos estão como atributos:

public class FormularioHelper {  
  
 private EditText campoNome;  
 private EditText campoEndereco;  
 private EditText campoTelefone;  
 private EditText campoSite;  
 private Ratingbar campoNota;  
  
 public class FormularioHelper(FormularioActivity activity) {  
 campoNome = (EditText) activity.findViewById(R.Id.formulario\_nome);  
 campoEndereco = (EditText) activity.findViewById(R.Id.formulario\_endereco);  
 campoTelefone = (EditText) activity.findViewById(R.Id.formulario\_telefone);  
 campoSite = (EditText) activity.findViewById(R.Id.formulario\_site);  
 campoNota = (RatingBar) activity.findViewById(R.Id.formulario\_nota);   
 }  
  
 public Aluno pegaAluno() {  
 Aluno aluno = new Aluno ();  
 aluno.setNome(campoNome.getText());  
  
 return null;   
 }  
}

Bom, agora, conseguimos acessar a classe 'pegaAluno'!

Já digitamos aluno.setNome(campoNome.getText) e como ele nós devolve um *editable*, temos que converter isso, usaremos, para tanto um toString. Não esqueça de fechar com um ";". Vamos repetir o aluno.setNome(campoNome.setText().toString()) por quatro vezes para inserirmos também os outros campos. Alteramos apenas os nomes do campo, inserindo o que ainda falta: endereço, telefone, site e nota.

A nota é um pouco diferente, para descrevê-la temos que acrescentar um getProgressque nos devolverá um Double e nele escreveremosvalueOf. Para a nota teremos o seguinte, aluno.setNota(Double.valueOf(campoNota.getProgress())).

Por fim, vamos devolver o aluno que acabamos de construir com return aluno.

Termos:

public Aluno pegaAluno() {  
 Aluno aluno = new Aluno ();  
 aluno.setNome(campoNome.getText().toString());  
 aluno.setEndereco(campoEndereco.getText().toString());  
 aluno.setTelefone(campoTelefone.getText().toString());  
 aluno.setSite(campoSite.getText().toString());  
 aluno.setNota(Double.valueOf(campoNota.getProgress()));  
 return aluno;   
 }

Vamos voltar ao FormularioActivity.java e vamos lembrar onde estávamos:

@Override   
 public boolean onOptionsItemSelected(MenuItem item) {   
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH\_SHORT).show();  
 Aluno aluno = helper.pegaAluno();  
  
 finish();  
 break;  
 }  
  
 return super.onOptionsItemSelected(item);   
 }

Apenas para confirmar se deu certo o que fizemos vamos deslocar o Toast.makeText(FormularioActivity.this, "Aluno salvo!" para a linha de baixo do Aluno aluno = helper.pegaAluno, através do "Command+V" e Command+C".

No Toast concatearemos, depois de "Aluno", o nome do aluno. Digitaremos, um "+" e "aluno.getNome() + salvo!". Ficaremos com

Toast.makeText(FormularioActivity.this. "Aluno" + aluno.getNome() + "salvo!", Toast.LENGTH\_SHORT).show();

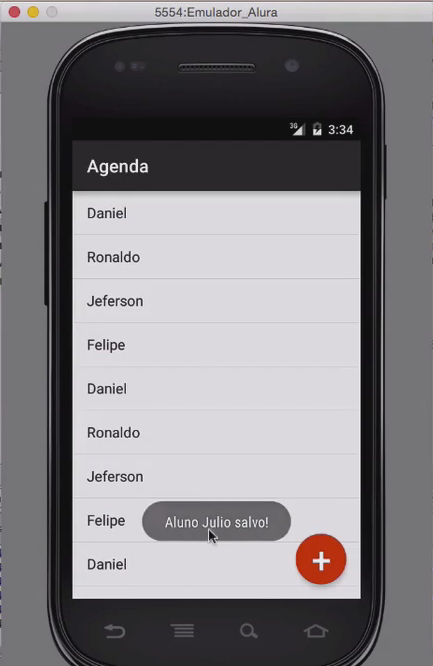
E no todo teremos:

@Override   
 public boolean onOptionsItemSelected(MenuItem item) {   
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
  
 Aluno aluno = helper.pegaAluno();  
 Toast.makeText(FormularioActivity.this, "Aluno" + aluno.getNome() + "salvo!", Toast.LENGTH\_SHORT).show();  
  
 finish();  
 break;   
 }  
  
 return super.onOptionsItemSelected(item);   
 }

Vamos rodar o emulador para ver como ficou! No campo formulário preencheremos com um novo aluno, o Júlio.



Agora, quando clicarmos no "Salvar", deve aparecer o "Aluno Júlio salvo".



Agora conseguimos preparar o caminho para salvar o nome do aluno no banco de dados!