PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Prof. Luiz Carlos Querino Filho
luiz.querino@fatec.sp.gov.br

Fatec Garça – 2019

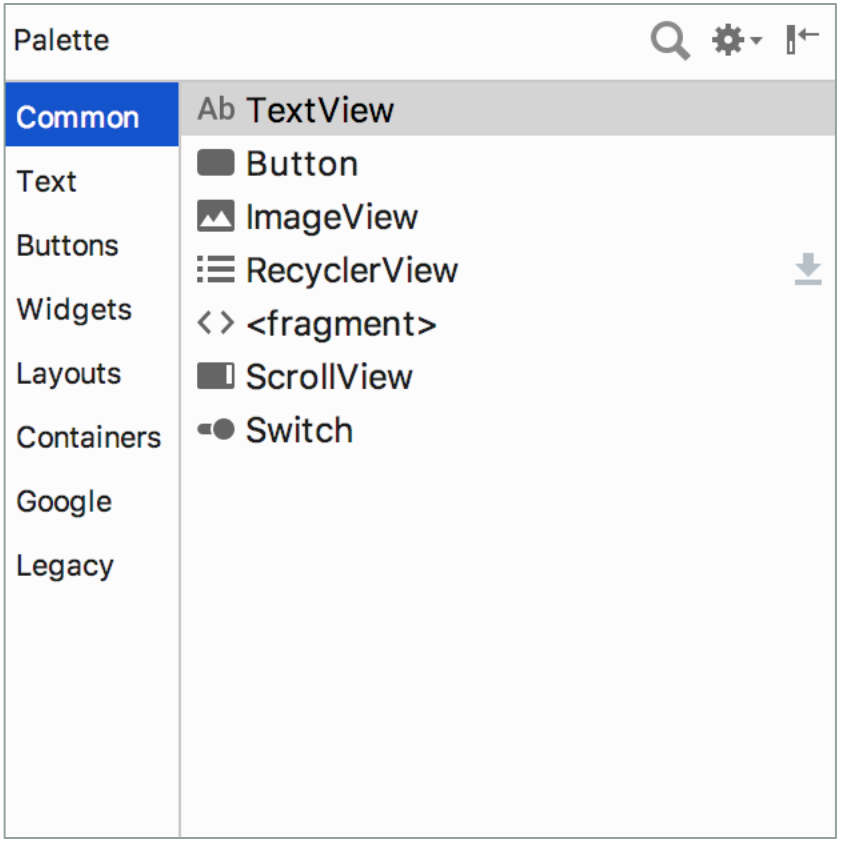
Conteúdo teórico para N1

CONCEITO IMPORTANTE

- Uma **tela** no Android é chamada de **Activity**.
- Para controlar o funcionamento da tela, criamos uma subclasse da **Activity** padrão existente no SDK (ou de uma de suas variações como **ActionBarActivity** e **ListActivity**).
- Os **layouts da tela** (seus componentes e suas configurações) ficam em um **arquivo XML** separado da classe Java, localizado na pasta **res\layout** do projeto.
- O arquivo onde você deve montar a interface é o **activity_main.xml**.
- Nas classes Activity, escrevemos o código Java que vai manipular a tela e seus componentes por meio de **eventos** (como o “toque” – ou clique – em um botão).

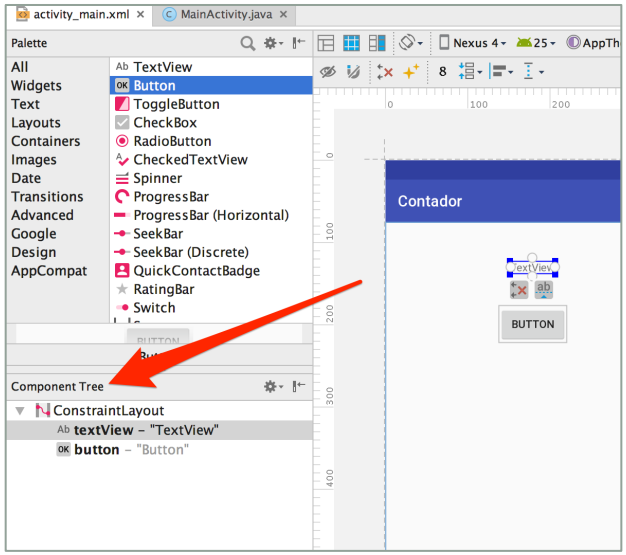
WIDGETS

- **Widgets** são os componentes visuais que usamos para construir as telas de um aplicativo Android.
- Todos eles são definidos por classes dentro do pacote **android.widget**.
- Todos são subclasses de **android.view.View**, que representa um espaço retangular na tela.
- No Editor de telas do Android Studio, os widgets estão disponíveis na Palheta (**Palette**)



MAIS SOBRE WIDGETS

- **Widgets** possuem um nome, o seu **id**. Ele pode ser visto na **janela Component Tree**.
- **Widgets** como bons objetos, tem **propriedades** e métodos.
- Para acessar e alterar as **propriedades** dos widgets no código Java, utilizamos seus **getters** e **setters**.
- Mas enquanto estamos construindo a interface do app, podemos modificar as propriedades do widget clicando sobre ele para selecioná-lo e alterando suas propriedades pela janela Properties do Android Studio.

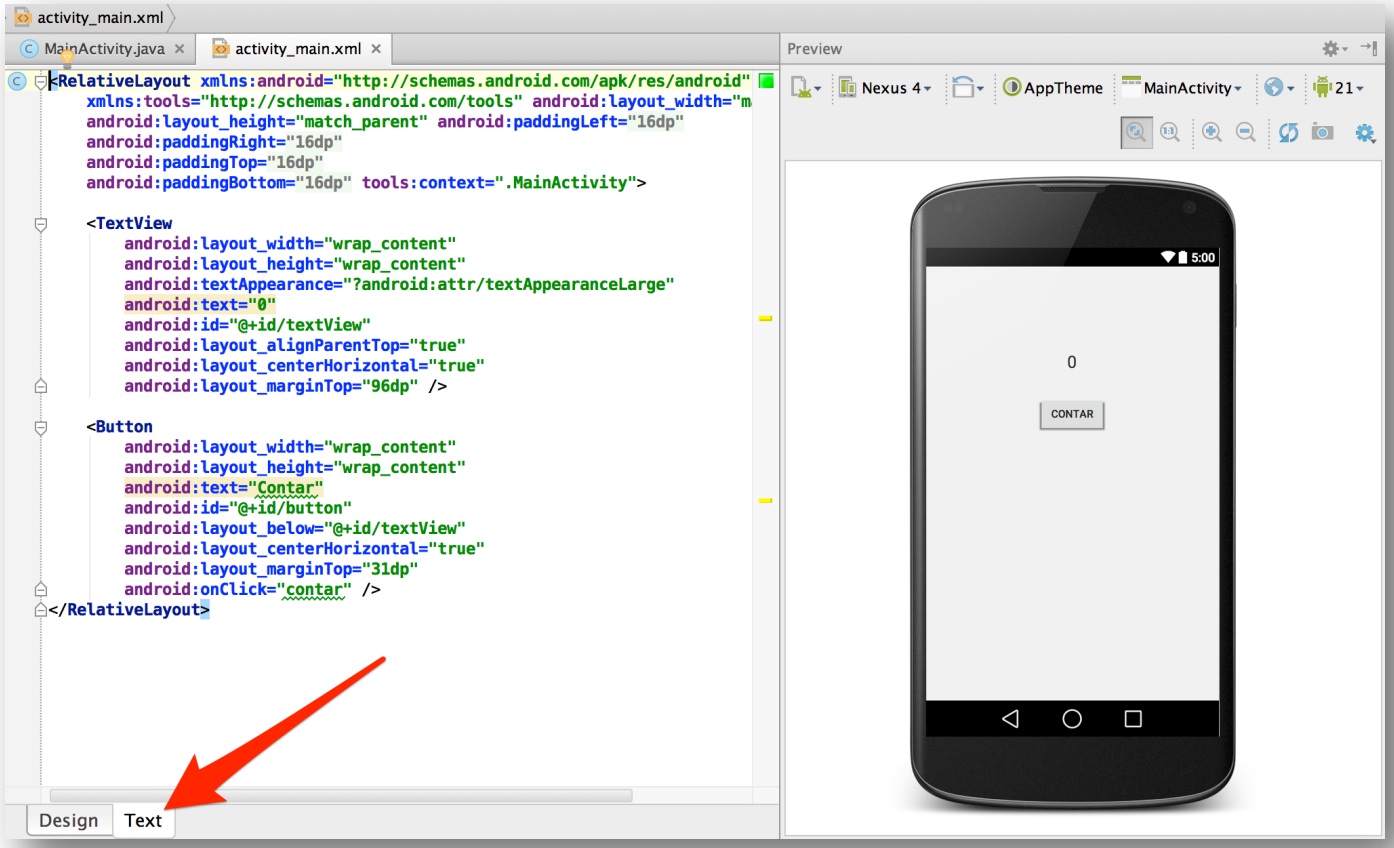


COMPONENTES DE UM PROJETO ANDROID

Arquivos de Layout

- Localizados no seu projeto em **res\layout**
- Arquivos XML que contém a definição dos elementos (**widgets**) existentes na tela e suas propriedades.
- O seu conteúdo pode ser editado diretamente em XML (**com cuidado**) ou graficamente.
- Opcionalmente, a interface também pode ser criada diretamente no código Java (o que **raramente** é recomendado)
- Os elementos ficam agrupados dentro de componentes gerenciadores de layout, como **ConstraintLayout, RelativeLayout** ou **LinearLayout.**

Conteúdo do arquivo `content_main.xml`



- Para alternar entre a edição gráfica e a edição XML, clique na **aba** correspondente (**Design** | **Text**).

/res/values/strings.xml

- Neste arquivo estão as *strings* de texto utilizadas no projeto
- Cada *string* possui um identificador (seu nome) e um valor associado.
- Quando é necessário indicar o texto a ser exibido em um objeto de interface, vincula-se o **identificador** da string ao objeto.
- Dessa forma, o processo de **internacionalização** do aplicativo fica muito mais fácil.
- As *strings* podem ser criadas visualmente ou diretamente pelo arquivo XML.

Activity

- A tela inicial da aplicação é representada por uma classe filha de **`android.app.AppCompatActivity`**
- O projeto deve possuir uma classe filha de **AppCompatActivity**, **Activity** ou **ListActivity** para cada tela.
- O método **`onCreate()`** deve ser implementado dentro da classe filha, sendo invocado pelo Android assim que a tela for criada.
- Para “desenhar” elementos na tela, são usadas classes descendentes de **`android.view.View`**.
- Widgets como botões e caixas de texto são todos “filhos” de **`View`**.

Mais sobre a **Activity**

- **Métodos da Activity:**
- `View findViewById(id)`
  - Esse método é um dos melhores amigos do desenvolvedor Android!
  - Ele "encontra" uma `View` dentro do layout gráfico e retorna uma refeência à ela.
  - Como parâmetro, deve ser passado o identificador **(ID**) da `View`, dentro de **`R.id`**. O identificador (**ID**) pode ser definido no editor de interface, clicando com o botão direito sobre o objeto e selecionando ***"Edit ID..."***
  - Serve para `qualquer tipo descendente de` **View** `(Button, TextView,` EditText, ...). Dessa forma, deve ser feito um "casting" no seu retorno:

```
EditText txtIdade = (EditText)findViewById(R.id.txtIdade);
```

Gerenciadores de Layout

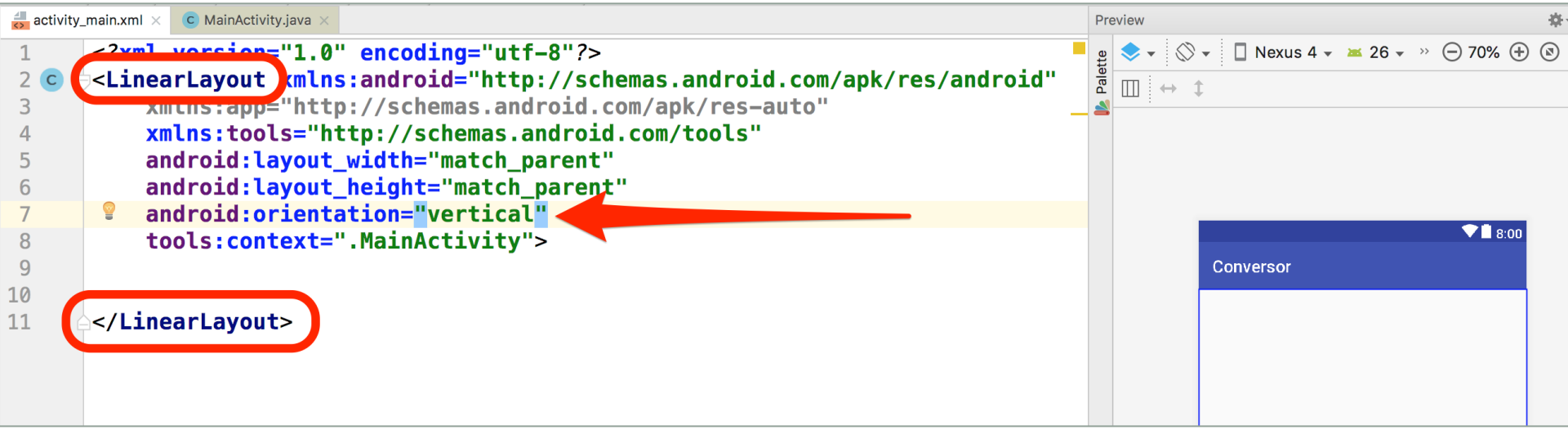
- Dentro de um arquivo de layout, os widgets ficam agrupados em “containers” de widgets.
- Estes containers são os **Gerenciadores de Layout**
- São “espaços retangulares” onde colocamos os **widgets** obedecendo algumas regras de posicionamento.
- O gerenciador de layout padrão do Android é o **ConstraintLayout**.
- Com ele, os widgets são posicionados com restrições de posicionamento (por exemplo: centralizado, com uma margem de x pontos, etc.).
- O uso de um gerenciador como o **ConstraintLayout** possibilita uma melhor adaptação da tela do programa aos diferentes tamanhos de aparelhos.

Outro tipo de Layout: **LinearLayout**

- Layouts são instâncias da classe `android.view.ViewGroups`
- O tipo de Layout mais básico é o **LinearLayout**. Com ele, os elementos são agrupados um após o outro, horizontalmente ou verticalmente.
- A definição da **orientação** (vertical ou horizontal) do **LinearLayout** é feita pela propriedade **orientation**.
  - Orientação vertical: widgets são “empilhados” (um abaixo do outro)
  - Orientação horizontal: widgets são colocados um ao lado do outro. Este é o padrão para o **LinearLayout**.
- Para criar layouts com várias linhas e colunas, mescle **LinearLayouts** com diferentes orientações.

LinearLayout como layout raiz

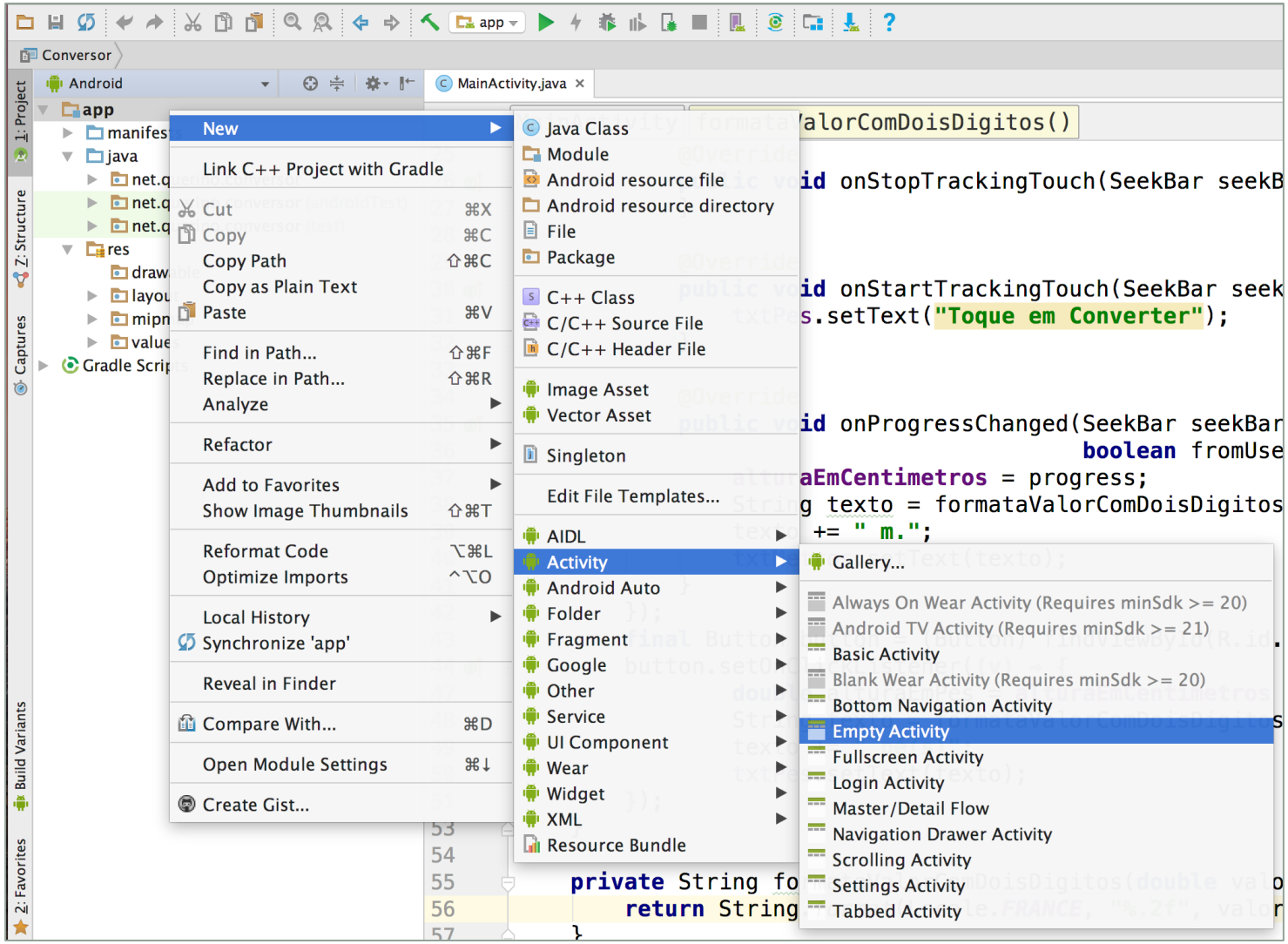
- Para mudar o **layout raiz** de uma tela para **LinearLayout**, você precisará editar diretamente o arquivo em XML.
- Como o padrão do **LinearLayout** é a orientação horizontal, é preciso também incluir a configuração específica para vertical.



Uma nova tela no seu aplicativo

- Para adicionar uma nova tela usando o assistente, clique com o botão direito do mouse sobre a estrutura do seu projeto (icone app na aba à esquerda do Android Studio) e selecione **New > Activity > Empty Activity**.
- Veja no próximo slide como chegar até este menu.

Uma nova tela no seu aplicativo



Abrindo uma nova tela

- Na **Activity** da tela principal, implemente agora um método de clique em um botão semelhante a este:

```
public void abrirPergunta(View view) {
    Intent intent = new Intent(this, PerguntaActivity.class);
    startActivity(intent);
}
```

- Primeiro, declaramos e instanciamos um **objeto** da classe **Intent**. Ao seu **construtor** passamos uma referência ao contexto atual (a **Activity** em que o código está sendo escrito – `this`) e o nome da classe da tela que será aberta (`PerguntaActivity.class`).
- Em seguida, para abrir efetivamente a nova tela, chamamos o método **startActivity**, passando o objeto `intent` a ele.

android.widget.EditText

- Este *widget* é usada para entrada de texto.
- No construtor de interface, ele está localizado dentro da categoria ***Text***.
- O Android traz dentro desta categoria variações do **EditText** apropriadas para entrada de diversos tipos de informação, como:
  - Texto Simples
  - Senhas
  - Datas
  - Números Inteiros com ou sem sinal
  - Números decimais
- IMPORTANTE: cabe ao usuário validar os dados!

android.widget.EditText

- Métodos importantes:

`Editable getText();`

- Retorna o texto existente no **EditText** como um tipo `android.text.Editable`. **Editable** é um tipo específico do Android – é basicamente uma **String** que pode ser modificada. Para um **Editable** se tornar uma **String**, use o método **`toString()`**:

```
String minhaString = editText1.getText().toString();
```

`void setText(CharSequence text)`

- Define o texto dentro do **EditText** como o **CharSequence** passado como parâmetro. **CharSequence** é uma *interface* que a classe **String** implementa. Então, uma **String** é um **CharSequence**. Dessa forma, você pode passar uma **String** diretamente como parâmetro:

```
String soma = String.valueOf(numero);
editText.setText(soma);
```

android.widget.Toast

- A classe **Toast** é usada para exibir uma mensagem rápida para o usuário.
- A forma mais básica de se usar uma **Toast** é criar uma mensagem com o método estático **`makeText()`**, e exibi-la com **`show()`**.
- **Toast.makeText(Context contexto, CharSequence texto, int duracao)**
  - **`contexto`**: geralmente, a `Activity` atual (`this`)
  - **`texto`**: a `String` a ser exibida
  - **`duracao`**: uma constante determinando o tempo que a mensagem ficará na tela. Pode ser:
    - **`Toast.LENGTH_SHORT`** para exibição rápida
    - **`Toast.LENGTH_LONG`** para maior duração
- Exemplo de uso:

```
Toast.makeText(this, "Sumindo rapidinho...",
                    Toast.LENGTH_SHORT).show();
```

O QUE MAIS VOCÊ DEVE SABER

- Java básico (variáveis, tipos primitivos, Strings, conbersão de valores, **if**, **for**...)
- Como deve ser programado em método de clique de botão
- O funcionamento e propósito do comando **findViewById**, ou seja, como acessar elementos da interface
- Depois de acessar elementos da interface (*widgets*) como fazer para ler e/ou mudar suas propriedades (**TextView** e **EditText**).
- Qual o procedimento de criação e abertura de novas telas
- A diferença em usar **LinearLayout** e **ConstraintLayout**
- Como podemos internacionalizar um app Android

BIBLIOGRAFIA

- QUERINO FILHO, L. C. Desenvolvendo seu Primeiro Aplicativo Android. Novatec Editora. 2013
- DEITEL, H. et al. Android for Programmers: An App-Driven Approach. Pearson Education. 2012.