1. Aula 5

1.1. Estrutura condicional SWITCH()

Outro comando para tomada de decisões é o switch. Seu funcionamento é bem simples, testamos uma expressão e, de acordo com o seu resultado, executamos um determinado bloco de comandos.

O switch vai funcionar como um interruptor, pois, dependendo da entrada que você der a ele, serão acionados somente certo (s) comando (s) dentre os que você disponibilizou.

É como se você criasse um menu, ou cardápio, e com o switch você escolhesse o que vai querer.

Declaração e Sintaxe do comando switch

Em Java, usamos e declaramos o comando switch da seguinte maneira:

```
switch( opção )
{
   case opção1:
        comandos caso a opção 1 tenha sido escolhida
        break;
   case opção2:
        comandos caso a opção 2 tenha sido escolhida
        break;
   case opção3:
        comandos caso a opção 3 tenha sido escolhida
        break;
   default:
        comandos caso nenhuma das opções anteriores tenha sido escolhida
}
```

A variável 'opção' geralmente é um inteiro ou caractere.

Se 'opção' receber 'opção1' como entrada são os códigos contidos na 'case opção1' que serão executados.

Se 'opção' receber 'opção2' como entrada são os códigos contidos na 'case opção2' que serão executados.

Se 'opção' receber 'opção3' como entrada são os códigos contidos na 'case opção3' que serão executados.

Se 'opção' receber qualquer outra coisa que não seja 'opção1', 'opção2' ou 'opção3 são os códigos contidos em 'default' que serão executados. FICOU CONFUSO, É PRECISO REFORMULAR ESTE TRECHO EM VERDE.

Veja um exemplo onde estaremos simulando a escolha do dia da semana.

```
Integer diaDaSemana;
EditText resultado1

resultado1=(EditText)findViewById(R.id.resultado1);
     diaDaSemana=2;
     switch(diaDaSemana){
```

```
        case 1:
            resultado1.setText("Domingo");
            break;
        case 2:
            resultado1.setText("Segunda");
            break;
        case 3:
            resultado1.setText("Terça");
            break;
        case 4:
            resultado1.setText("Quarta");
            break;
        case 5:
            resultado1.setText("Quinta");
            break;
        case 6:
            resultado1.setText("Sexta");
            break;
        case 7:
            resultado1.setText("Sábado");
            break;
        default:
            resultado1.setText("Este não é um dia válido");
            break;
    }
```

1.2. Tratamento de texto

Sobre tratamento de texto, estaremos melhorando o desempenho do código e facilitando a sua manutenção utilizando alguns métodos.

Os métodos disponíveis na classe String podem servir para uma simples junção de texto, comparação de conteúdo, busca de determinados dados, entre muitas outras funcionalidades.

Em Java, String é uma sequência de caracteres utilizada para representação e manipulação de texto. Quando é necessário representar informações textuais em uma aplicação, seja ela para Desktop ou Web, como nome de pessoas, informações sobre endereço ou comentários em uma matéria no jornal, instâncias da classe String serão criadas invariavelmente. Isto é, sempre que precisarmos mostrar alguma informação em um sistema, dificilmente vamos conseguir isso sem o auxílio de Strings.

Conhecendo alguns métodos:

toUpperCase()

O método toUpperCase() retorna uma nova String com o mesmo conteúdo da original, só que com todos os caracteres em letras maiúsculas.

```
String nomeCliente;
EditText resultado1;

resultado1=(EditText)findViewById(R.id.resultado1);
```

```
nomeCliente="joão da silva";
resultado1.setText(nomeCliente.toUpperCase());
```

O resultado deste método vai ser:

JOÃO DA SILVA

toLowerCase()

O método toLowerCase converte toda a String para caixa baixa e o toUpperCase faz o inverso, convertendo toda a String para caixa alta.

```
String nomeCliente;
EditText resultado1;
resultado1=(EditText)findViewById(R.id.resultado1);
nomeCliente="JOÃO DA SILVA";
resultado1.setText(nomeCliente.toLowerCase());
```

O resultado deste método vai ser:

João da silva.

trim()

O método trim remove espaços em branco no inicial e no final da String.

```
String nomeCliente;
EditText resultado1;

resultado1=(EditText)findViewById(R.id.resultado1);

nomeCliente="  JOÃO DA SILVA   ";
resultado1.setText(nomeCliente.trim());
```

O resultado deste método vai ser, os espaços foram removidos conforme o digitado acima:

JOÃO DA SILVA

1.3. Layout

Alguns atributos são essenciais para determinar como exemplo a largura e altura do layout, o alinhamento das informações, como a utilização de cores no fundo da tela, cor de texto, tamanho de letra e o modo de exibição dos dados com relative ou linear, além de definição de margem.

O layout define a estrutura visual para uma interface do usuário, como a IU de uma atividade ou de um widget de aplicativo. É possível declarar um layout de dois modos:

Declarar elementos da IU em XML. O Android fornece um vocabulário XML direto que corresponde às classes e subclasses de View, como as de widgets e layouts.

Instanciar elementos do layout em tempo de execução. O aplicativo pode criar objetos View e ViewGroup (e processar suas propriedades) programaticamente.

A estrutura do Android dá a flexibilidade de usar um desses métodos ou ambos para declarar e gerenciar a IU do aplicativo. Por exemplo, você pode declarar os layouts padrão do aplicativo em XML, incluindo os elementos da tela que aparecerão neles e em suas propriedades. Em seguida, você poderia

adicionar código ao aplicativo que modificaria o estado dos objetos da tela, inclusive os declarados em XML, em tempo de execução.

Todos os grupos de exibições contêm largura e altura (layout_width e layout_height), e cada exibição é obrigatória para defini-las.

Width = largura

Height = altura

DP:

A unidade de medida que estamos utilizando é a "DP" Density-independent Pixel. Essa unidade é relativa à resolução da tela. Por exemplo, se a resolução da tela é de 160 dpi, significa que um dp representa 1 pixel em um total de 160.

USO: aconselho: ao invés de usar o px, sempre use o dp.

SP:

DEFINIÇÃO: (Scale-independent Pixels) Idem ao dp, mas também considera o tamanho da fonte que o usuário está utilizando. É recomendado que use essa unidade quando especificar o tamanho de uma fonte para que esta seja automaticamente ajustada conforme as preferências da tela do usuário.

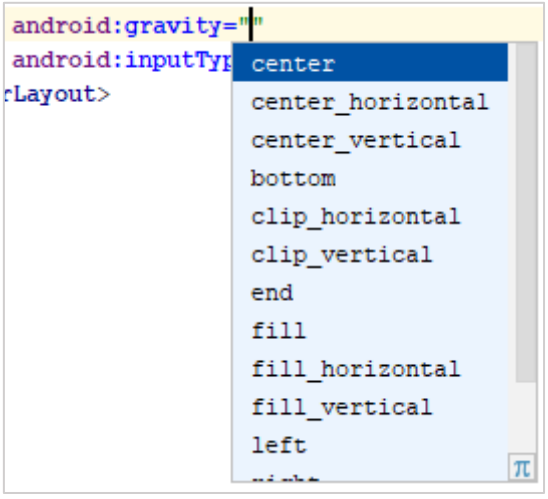
USO: Sempre utilize sp para fontes!

Gravidade Android

android: gravidade é um atributo que define a gravidade do conteúdo da exibição usada.

O android: gravidade especifica como um objeto deve posicionar seu conteúdo nos eixos X e Y.

Os valores possíveis do android: gravidade (superior, inferior, esquerda, direita, centro, center_vertical, center_horizontal, etc).

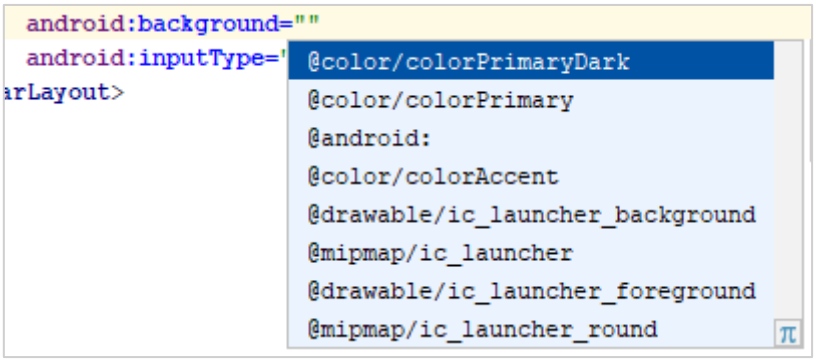


Para alinhar componentes na tela horizontalmente e centralizando, utilizamos o atributo abaixo.

O android:layout_centerHorizontal alinha na horizontal os componentes, apenas passando o valor "true" ele já ativa a propriedade.

Para definir cor de fundo no componente, utilizamos o atributo abaixo.

android:background=”#FF00FF” ou padrões do sistema.



Para definir cor no texto, utilizamos o atributo abaixo.

android:textColor=”#FFFFFF”

Para definir o tamanho do texto, utilizamos o atributo abaixo:

android:textSize=”20sp”;