AGENDA 6

TÉCNICAS DE PROGRAMAÇÃO NO ANDROID STUDIO

GEEaD - Grupo de Estudos de Educação a Distância

Centro de Educação Tecnológica Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO

CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

PROGRAMAÇÃO MOBILE I

Expediente

Autor:

GUILHERME HENRIQUE GIROLLI

Revisão Técnica:

Eliana Cristina Nogueira Barion

Revisão Gramatical:

Juçara Maria Montenegro Simonsen Santos

Editoração e Diagramação:

Flávio Biazim

São Paulo – SP, 2020

MERGULHANDO NO TEMA...

Utilização de métodos

Em determinados momentos da programação de aplicativos, encontramos a necessidade de repetir inúmeras vezes o mesmo código ou trechos da programação em vários locais diferentes do nosso programa. Para evitar que o programador reescreva várias vezes os mesmos códigos em locais diferentes, utilizamos os métodos.

Um exemplo de método é a função de “trocaJogador”. Essa função é chamada todas as vezes que um jogador escolhe uma posição do tabuleiro, e passa a jogada para o seu adversário.

```
public void trocaJogador()
{
    if(jogador.equals("X"))
    {
        jogador = "O";
    }
    else
    {
        jogador = "X";
    }
}
```

Os métodos possuem uma visibilidade, um tipo de retorno e argumentos. A visibilidade “public” é responsável por deixar o método acessível para ser utilizado em qualquer parte do nosso app. Já a visibilidade “private” deixa o método acessível apenas na classe que ele foi desenvolvido. Existem outras opções de visibilidade provenientes da linguagem Java, em nosso projeto vamos utilizar apenas a visibilidade “public”.

Cada método pode gerar um valor após a sua execução, para isso é necessário informar qual é o tipo de retorno, como por exemplo, um valor inteiro. No caso do nosso método “trocaJogador” não é necessário retornar nada após a sua execução, dessa forma colocamos seu tipo como “void”.

Os argumentos são declarados no meio dos parênteses através de uma lista que obedece a um tipo e um nome. Caso tenha mais de um parâmetro, esses são separados por vírgula. Os argumentos de um método têm a função de transmitir valores para as rotinas que serão executadas internamente no método. Para a função de “trocaJogador” não foi necessário passar nenhum argumento para a sua execução.

Para utilizar um método basta apenas chamar o seu nome, como podemos observar no código que é executado quando o botão “btn11Prog” é pressionado.

```
btn11Prog.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        btn11Prog.setText(jogador);
        //Momento em que o método é chamado
        trocaJogador();
        btn11Prog.setClickable(false);
        txtJogadorProg.setText("Jogador: "+jogador);
    }
});
```

Tipos de dados

Durante a programação de aplicativos nos deparamos com a manipulação de dados na programação. Esses dados obedecem a um tipo e podem ser utilizados para armazenar valores em variáveis e/ou constantes, retornos de métodos, entre outras utilizações.
Encontramos alguns tipos que são utilizados com uma maior frequência no desenvolvimento mobile.

- **Int -** Dados do tipo “int” armazena valores números inteiros, incluindo o zero e números negativos.

- **float -** Armazena dados com valor numérico decimal, ou seja, com virgula.
Dados do tipo “float” não são tão precisos pois armazenam de 6 a 7 casas decimais apenas.

- **double -** Armazena dados com valor numérico decimal. O “double” é utilizado para armazenar valores com uma alta precisão de até 15 casas decimais.

- **String -** Armazena dados com valores compostos de caracteres, como por exemplo, uma frase ou um nome de uma pessoa.

- **boolean –** Armazena dados que correspondem a verdadeiro (true) ou falso (false).
Dados do tipo “boolean” são comumente encontrados em verificações e testes.

Variável e constante

Variável é um local para armazenamento temporário na memória do dispositivo indicado por um nome previamente escolhido no momento da sua declaração. É necessário também identificar um tipo para a variável, como por exemplo, uma variável que armazena caracteres recebe o tipo “String”. A função principal da variável é armazenar valores temporários que serão utilizados nas rotinas do aplicativo, esses valores podem sofrer alterações durante a execução do aplicativo. Veja a seguir o código para gerar uma variável.

```
String jogador = "X";
```

A constante recebe todas as características de uma variável com exceção da característica de alternância do valor do seu conteúdo, ou seja, a constante recebe um valor fixo, que não vai sofrer alteração durante toda a execução do aplicativo. Veja a seguir como é feita a declaração de uma constante na programação.

```
final String jogador = "X";
```

Vetor e Matriz

Uma variável armazena apenas um valor para cada alocação. Para trabalhar com vários valores de um mesmo tipo relacionados há um processamento comum podemos utilizar um vetor. O vetor é indicado por um nome e um índice de acordo com a quantidade de divisões definidas no ato do seu desenvolvimento.
No exemplo a seguir encontramos uma variável para armazenar um nome de um aluno.

```
String aluno = "João da Silva";
```

Na sequência declaramos um vetor para armazenar os nomes de três alunos de um grupo de estudos. É importante ressaltar que em um mesmo local vamos armazenar três nomes, separados internamente por um índice. O tamanho do vetor é definido de acordo com a necessidade da programação.

```
String[] aluno = new String[3];
aluno[0] = "João da Silva";
aluno[1] = "Maria de Souza";
aluno[2] = "Carlos Santos";
```

O índice de um vetor sempre inicia no “0”. Em um vetor de 3 posições vamos encontrar índices 0, 1, e 2.

A matriz é um vetor bidimensional composto de linhas e colunas para armazenar valores de um mesmo tipo. Na sequência declaramos uma matriz com 3 linhas e 2 colunas. Essa matriz vai armazenar em cada linha dados de um aluno específico. Na primeira coluna da matriz vamos encontrar os nomes dos alunos, e na segunda coluna vamos encontrar a nota final desse aluno.

```
String[][] aluno = new String[3][2];
aluno[0][0] = "João da Silva";
aluno[1][0] = "Maria de Souza";
aluno[2][0] = "Carlos Santos";
aluno[0][1] = "R";
aluno[1][1] = "B";
aluno[2][1] = "MB";
```

A figura 4 ilustra como fica os locais alocados pela variável, vetor e matriz na memória do dispositivo mobile.

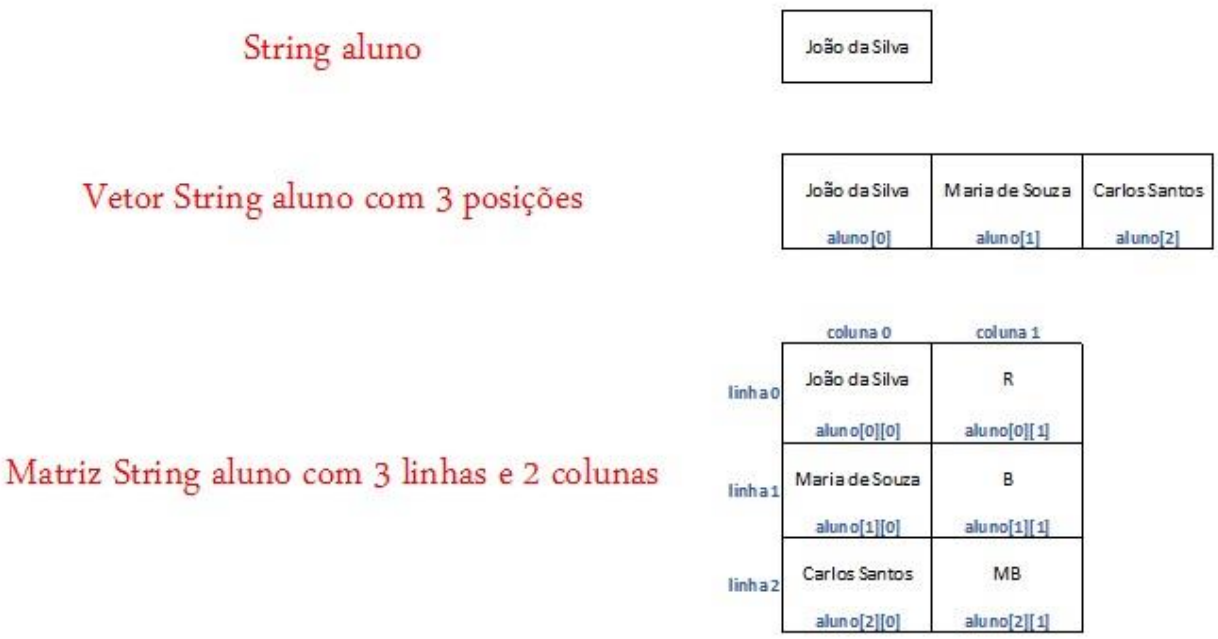


Figura 4 – Variável, vetor e matriz.

Estrutura de decisão

A estrutura de decisão é utilizada para que uma parte do código seja executada apenas se uma determinada condição "booleana" estiver sendo atendida. Caso contrário podemos determinar que o aplicativo execute uma outra parte do código. O código a seguir demonstra a utilização de uma estrutura de decisão "if e else" para alterar o símbolo do jogador. Caso o jogador atual seja o "X", a condição será atendida e alteramos para o "0". Caso o jogador não é o "X", a condição não é atendida e executamos a alteração da variável para "X", uma vez que ela atualmente só pode ser o "0".

```
if(jogador.equals("X"))
{
    jogador = "O";
}
else
{
    jogador = "X";
}
```

Estrutura de repetição

Estrutura de repetição é utilizada para que uma parte do código seja executada e repetida durante uma determinada condição. É uma estrutura muito utilizada na programação e uma das suas principais funções é gerar uma economia de código em certas rotinas.

Na programação Java encontramos alguns tipos de estrutura de repetição:

While – A tradução de “while” é enquanto e essa é a sua principal característica. Essa estrutura de repetição efetua um teste “booleano” antes de efetuar a sua primeira execução dos códigos internos. Como o teste antecede a execução dos códigos, existe a possibilidade da não execução das suas rotinas internas nenhuma vez.

```
int linha = 0;
int coluna = 0;
while(linha < 3)
{
    while(coluna < 3)
    {
        matrizVerificacao[linha][coluna] =
String.valueOf(linha)+String.valueOf(coluna);
        coluna = coluna + 1;
    }
    coluna = 0;
    linha = linha + 1;
}
```

Do-While – Essa estrutura é uma derivação do “While” com um diferencial muito importante, ela executa suas rotinas de códigos internos, e após a execução efetua um teste “booleano” para verificar se continua ou se encerra a sua próxima repetição. Desta forma essa estrutura executa seus códigos no mínimo uma vez.

```
int linha = 0;
int coluna = 0;
do
{
    do
    {
        matrizVerificacao[linha][coluna] =
String.valueOf(linha)+String.valueOf(coluna);
        coluna = coluna + 1;
    }while(coluna < 3);
    coluna = 0;
    linha = linha + 1;
}while(linha < 3);
```

For – É uma estrutura mais compacta que no seu cabeçalho trás além do teste "booleano", uma estrutura de interação e uma estrutura de inicialização. O que não encontramos na estrutura "While". No seu funcionamento é idêntica a estrutura "While".

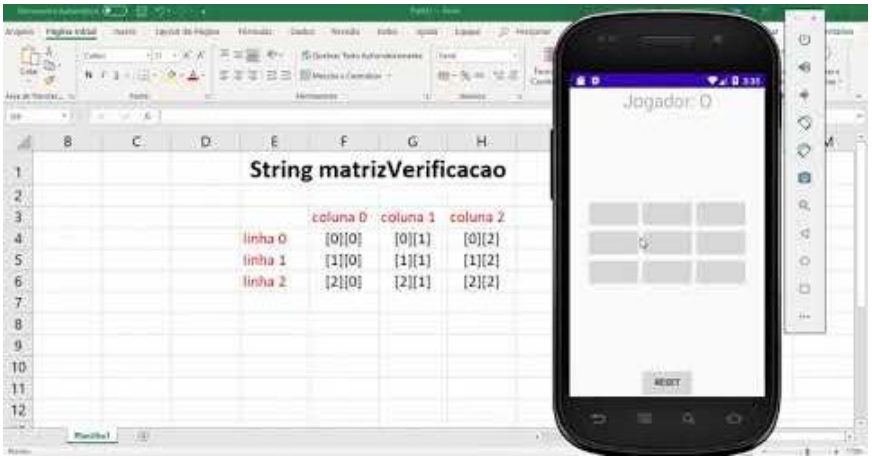
```
for(int linha = 0; linha < 3; linha++)
{
    for(int coluna = 0; coluna < 3; coluna++)
    {
        matrizVerificacao[linha][coluna] =
String.valueOf(linha)+String.valueOf(coluna);
    }
}
```

VOCÊ NO COMANDO

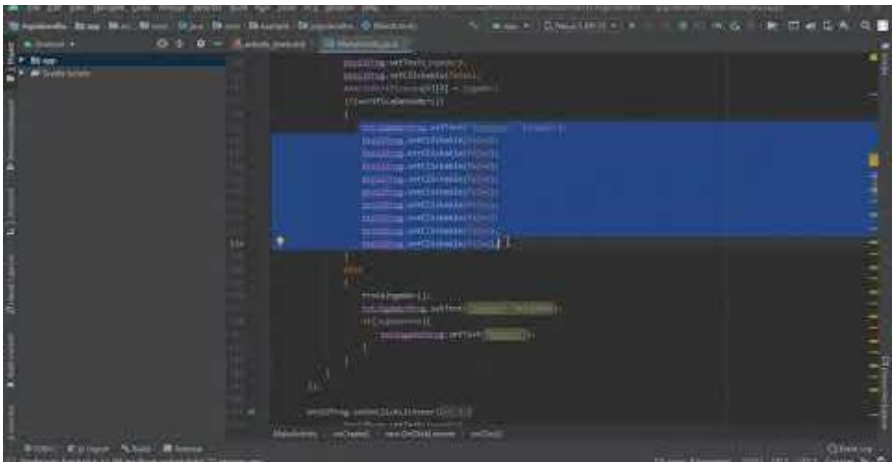
Assista o vídeo a seguir para compreender algumas rotinas necessárias para o bom funcionamento do aplicativo jogo da velha.

Acompanhe o desenvolvimento dos métodos no projeto “Jogodavelha” disponível em: **app > java > com.example.jogodavelha > MainActivity.java.**

Vídeo explicativo do aplicativo jogo da velha. Youtube.com: https://youtu.be/MWTZbcxKqbo



Vídeo sobre a codificação de métodos. Youtube.com: https://youtu.be/2FguZiwVtDA



Vídeo sobre a codificação de métodos. Youtube.com: https://youtu.be/5qN-JCjPWW8

Verifique os códigos do projeto "Jogodavelha" disponível em: **app > java > com.example.jogodavelha > MainActivity.java.**

```
package com.example.jogodavelha;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    String jogador = "X";
    //matriz para verificação
    String[][] matrizVerificacao = new String[3][3];
    //Empate
    int jogadas = 0;

    public void trocaJogador()
    {
        if(jogador.equals("X"))
        {
            jogador = "O";
        }
        else
        {
            jogador = "X";
        }
        jogadas = jogadas + 1;
    }

    private void inicializaMatrizVerificacao()
    {
        for(int linha = 0; linha < 3; linha++)
        {
            for(int coluna = 0; coluna < 3; coluna++)
            {
                matrizVerificacao[linha][coluna] =
String.valueOf(linha)+String.valueOf(coluna);
            }
        }
    }

    private boolean verificaGanhador()
    {
        boolean ganhador = false;

        //horizontal
        if(matrizVerificacao[0][0].equals(matrizVerificacao[0][1]) &&
                matrizVerificacao[0][0].equals(matrizVerificacao[0][2]))
        {
            ganhador = true;
        }
```

```
        if(matrizVerificacao[1][0].equals(matrizVerificacao[1][1]) &&
                matrizVerificacao[1][0].equals(matrizVerificacao[1][2]))
        {
            ganhador = true;
        }

        if(matrizVerificacao[2][0].equals(matrizVerificacao[2][1]) &&
                matrizVerificacao[2][0].equals(matrizVerificacao[2][2]))
        {
            ganhador = true;
        }

        //vertical
        if(matrizVerificacao[0][0].equals(matrizVerificacao[1][0]) &&
                matrizVerificacao[0][0].equals(matrizVerificacao[2][0]))
        {
            ganhador = true;
        }

        if(matrizVerificacao[0][1].equals(matrizVerificacao[1][1]) &&
                matrizVerificacao[0][1].equals(matrizVerificacao[2][1]))
        {
            ganhador = true;
        }

        if(matrizVerificacao[0][2].equals(matrizVerificacao[1][2]) &&
                matrizVerificacao[0][2].equals(matrizVerificacao[2][2]))
        {
            ganhador = true;
        }

        //diagonal
        if(matrizVerificacao[0][0].equals(matrizVerificacao[1][1]) &&
                matrizVerificacao[0][0].equals(matrizVerificacao[2][2]))
        {
            ganhador = true;
        }

        if(matrizVerificacao[0][2].equals(matrizVerificacao[1][1]) &&
                matrizVerificacao[0][2].equals(matrizVerificacao[2][0]))
        {
            ganhador = true;
        }
        return ganhador;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();

        final Button btn11Prog = (Button) findViewById(R.id.btn11);
        final Button btn12Prog = (Button) findViewById(R.id.btn12);
        final Button btn13Prog = (Button) findViewById(R.id.btn13);
        final Button btn21Prog = (Button) findViewById(R.id.btn21);
        final Button btn22Prog = (Button) findViewById(R.id.btn22);
```

```
        final Button btn23Prog = (Button) findViewById(R.id.btn23);
        final Button btn31Prog = (Button) findViewById(R.id.btn31);
        final Button btn32Prog = (Button) findViewById(R.id.btn32);
        final Button btn33Prog = (Button) findViewById(R.id.btn33);
        final TextView txtJogadorProg = (TextView) findViewById(R.id.txtJogador);
        final Button btnResetProg = (Button) findViewById(R.id.btnReset);

        inicializaMatrizVerificacao();

        btn11Prog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btn11Prog.setText(jogador);
                btn11Prog.setClickable(false);
                matrizVerificacao[0][0] = jogador;
                if(verificaGanhador())
                {
                    txtJogadorProg.setText("Ganhador: "+jogador);
                    btn11Prog.setClickable(false);
                    btn12Prog.setClickable(false);
                    btn13Prog.setClickable(false);
                    btn21Prog.setClickable(false);
                    btn22Prog.setClickable(false);
                    btn23Prog.setClickable(false);
                    btn31Prog.setClickable(false);
                    btn32Prog.setClickable(false);
                    btn33Prog.setClickable(false);
                }
                else
                {
                    trocaJogador();
                    txtJogadorProg.setText("Jogador: "+jogador);
                    if(jogadas==9){
                        txtJogadorProg.setText("Empate!");
                    }
                }
            }
        });

        btn12Prog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btn12Prog.setText(jogador);
                btn12Prog.setClickable(false);
                matrizVerificacao[0][1] = jogador;
                if(verificaGanhador())
                {
                    txtJogadorProg.setText("Ganhador: "+jogador);
                    btn11Prog.setClickable(false);
                    btn12Prog.setClickable(false);
                    btn13Prog.setClickable(false);
                    btn21Prog.setClickable(false);
                    btn22Prog.setClickable(false);
                    btn23Prog.setClickable(false);
                    btn31Prog.setClickable(false);
                    btn32Prog.setClickable(false);
                    btn33Prog.setClickable(false);
                }
```

```
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }

        }
    });

    btn13Prog.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn13Prog.setText(jogador);
            btn13Prog.setClickable(false);
            matrizVerificacao[0][2] = jogador;
            if(verificaGanhador())
            {
                txtJogadorProg.setText("Ganhador: "+jogador);
                btn11Prog.setClickable(false);
                btn12Prog.setClickable(false);
                btn13Prog.setClickable(false);
                btn21Prog.setClickable(false);
                btn22Prog.setClickable(false);
                btn23Prog.setClickable(false);
                btn31Prog.setClickable(false);
                btn32Prog.setClickable(false);
                btn33Prog.setClickable(false);
            }
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }
        }
    });

    btn21Prog.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn21Prog.setText(jogador);
            btn21Prog.setClickable(false);
            matrizVerificacao[1][0] = jogador;
            if(verificaGanhador())
            {
                txtJogadorProg.setText("Ganhador: "+jogador);
                btn11Prog.setClickable(false);
                btn12Prog.setClickable(false);
                btn13Prog.setClickable(false);
                btn21Prog.setClickable(false);
                btn22Prog.setClickable(false);
                btn23Prog.setClickable(false);
                btn31Prog.setClickable(false);
```

```
                btn32Prog.setClickable(false);
                btn33Prog.setClickable(false);
            }
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }
        }
    });

    btn22Prog.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn22Prog.setText(jogador);
            btn22Prog.setClickable(false);
            matrizVerificacao[1][1] = jogador;
            if(verificaGanhador())
            {
                txtJogadorProg.setText("Ganhador: "+jogador);
                btn11Prog.setClickable(false);
                btn12Prog.setClickable(false);
                btn13Prog.setClickable(false);
                btn21Prog.setClickable(false);
                btn22Prog.setClickable(false);
                btn23Prog.setClickable(false);
                btn31Prog.setClickable(false);
                btn32Prog.setClickable(false);
                btn33Prog.setClickable(false);
            }
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }
        }
    });

    btn23Prog.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn23Prog.setText(jogador);
            btn23Prog.setClickable(false);
            matrizVerificacao[1][2] = jogador;
            if(verificaGanhador())
            {
                txtJogadorProg.setText("Ganhador: "+jogador);
                btn11Prog.setClickable(false);
                btn12Prog.setClickable(false);
                btn13Prog.setClickable(false);
                btn21Prog.setClickable(false);
                btn22Prog.setClickable(false);
```

```
                btn23Prog.setClickable(false);
                btn31Prog.setClickable(false);
                btn32Prog.setClickable(false);
                btn33Prog.setClickable(false);
            }
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }
        }
    });

    btn31Prog.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn31Prog.setText(jogador);
            btn31Prog.setClickable(false);
            matrizVerificacao[2][0] = jogador;
            if(verificaGanhador())
            {
                txtJogadorProg.setText("Ganhador: "+jogador);
                btn11Prog.setClickable(false);
                btn12Prog.setClickable(false);
                btn13Prog.setClickable(false);
                btn21Prog.setClickable(false);
                btn22Prog.setClickable(false);
                btn23Prog.setClickable(false);
                btn31Prog.setClickable(false);
                btn32Prog.setClickable(false);
                btn33Prog.setClickable(false);
            }
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }
        }
    });

    btn32Prog.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn32Prog.setText(jogador);
            btn32Prog.setClickable(false);
            matrizVerificacao[2][1] = jogador;
            if(verificaGanhador())
            {
                txtJogadorProg.setText("Ganhador: "+jogador);
                btn11Prog.setClickable(false);
                btn12Prog.setClickable(false);
                btn13Prog.setClickable(false);
```

```
                btn21Prog.setClickable(false);
                btn22Prog.setClickable(false);
                btn23Prog.setClickable(false);
                btn31Prog.setClickable(false);
                btn32Prog.setClickable(false);
                btn33Prog.setClickable(false);
            }
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }
        }
    });

    btn33Prog.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn33Prog.setText(jogador);
            btn33Prog.setClickable(false);
            matrizVerificacao[2][2] = jogador;
            if(verificaGanhador())
            {
                txtJogadorProg.setText("Ganhador: "+jogador);
                btn11Prog.setClickable(false);
                btn12Prog.setClickable(false);
                btn13Prog.setClickable(false);
                btn21Prog.setClickable(false);
                btn22Prog.setClickable(false);
                btn23Prog.setClickable(false);
                btn31Prog.setClickable(false);
                btn32Prog.setClickable(false);
                btn33Prog.setClickable(false);
            }
            else
            {
                trocaJogador();
                txtJogadorProg.setText("Jogador: "+jogador);
                if(jogadas==9){
                    txtJogadorProg.setText("Empate!");
                }
            }
        }
    });

    btnResetProg.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            btn11Prog.setClickable(true);
            btn11Prog.setText("");
            btn12Prog.setClickable(true);
            btn12Prog.setText("");
            btn13Prog.setClickable(true);
            btn13Prog.setText("");
            btn21Prog.setClickable(true);
```

```
                btn21Prog.setText("");
                btn22Prog.setClickable(true);
                btn22Prog.setText("");
                btn23Prog.setClickable(true);
                btn23Prog.setText("");
                btn31Prog.setClickable(true);
                btn31Prog.setText("");
                btn32Prog.setClickable(true);
                btn32Prog.setText("");
                btn33Prog.setClickable(true);
                btn33Prog.setText("");
                inicializaMatrizVerificacao();
                jogadas = 0;
                txtJogadorProg.setText("Jogador: "+jogador);
            }
        });
    }
}
```

ATIVIDADE ONLINE

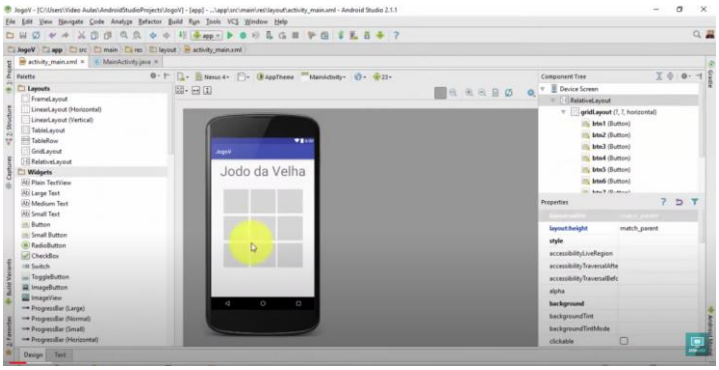
Após codificar todo o projeto, execute alterações em seu design, como alteração de cores, inserção de tela de Splash entre outros recursos de sua escolha. Na sequência capture imagens (fotos) do programa executando e envie para seu professor, envie também o código XML e o código Java utilizado para desenvolvimento do projeto “Jogodavelha”.

AMPLIANDO HORIZONTES

Para auxiliar no processo de aprendizagem dos temas discutidos nesta aula, seguem abaixo algumas dicas de filme e livro que se relacionam com o conteúdo estudado. Estas dicas são muito importantes para você!

Vídeo:

Canal Extraclasse - Criar Jogo da Velha no Android Studio. Vídeo do Youtube.com: https://www.youtube.com/watch?v=_y5cyOsAFA4



Livros:

QUEIROS, R. Android: Desenvolvimento de Aplicações com Android Studio. FCA, 2016.