PREZADOS SRS. CRISTIAN E IASMIN COM GRANDE SATISFAÇÃO VENHO APRESENTAR-LHES A RESOLUÇÃO DO EXERCÍCIO PROPOSTO E ME COMPROMETO A UTILIZAR ESTRATÉGICAMENTE MEU RACIOCÍNIO E CRESCER DIA A DIA PARA O PROJETO DOS SRS. NA EMPRESA ATECH E ME COLOCO SOBRE O COMANDO DOS SRS. PARA ATENDER ÀS EXIGENCIAS E COM RESPEITO E ÉTICA OFERECER MEU TRABALHO NA ORGANIZAÇÃO.

RESOLUÇÃO DE EXERCÍCIO PROPOSTO SOBRE MATRIZES (MULTIDIMENSIONAL ARRAYS) PELA ATECH – NEGÓCIOS EM TECNOLOGIAS S/A.

BRUNO PRESTES SILVA OLIVEIRA

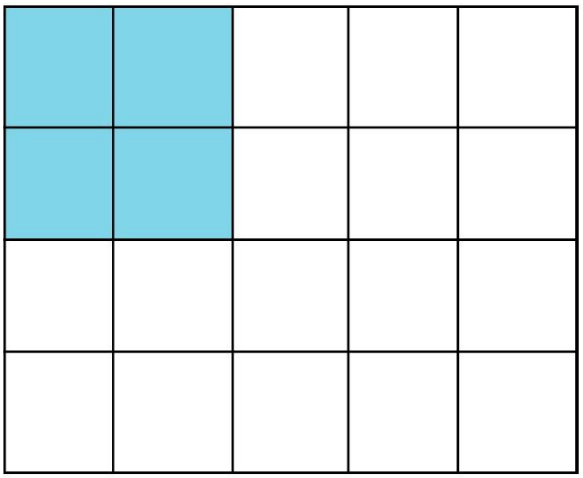
SÃO PAULO – SP

2020

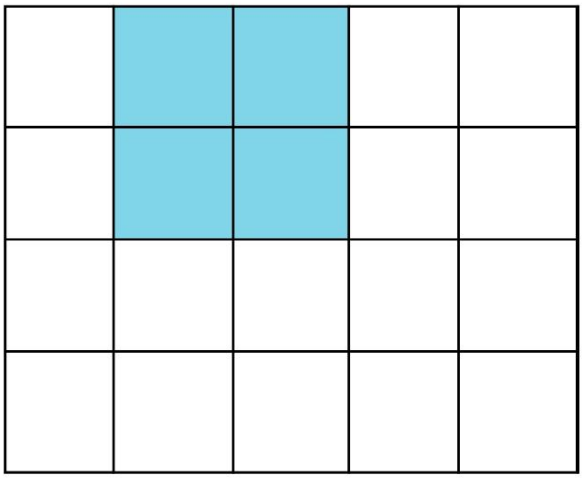
CLASSE RECTANGLEFINDER

O objetivo desta classe é preencher uma ArrayList de retângulos perfeitos(quadrados) sendo estes os 12 possíveis quadrados dentro de uma matriz [4][5], confira abaixo a linha de raciocínio para a visualização de 12 possíveis quadrados dentro de uma matriz[4][5].

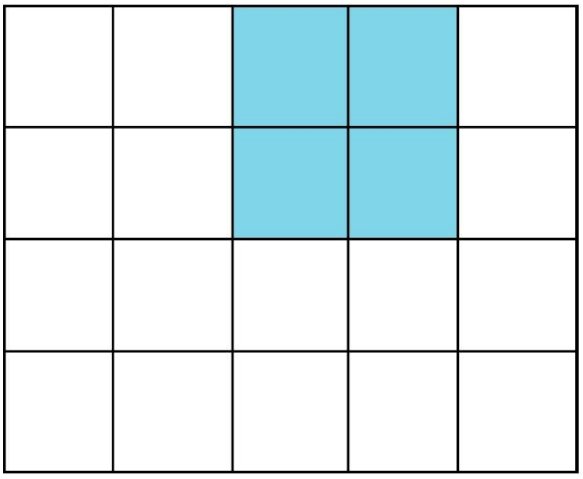
Retângulo 1



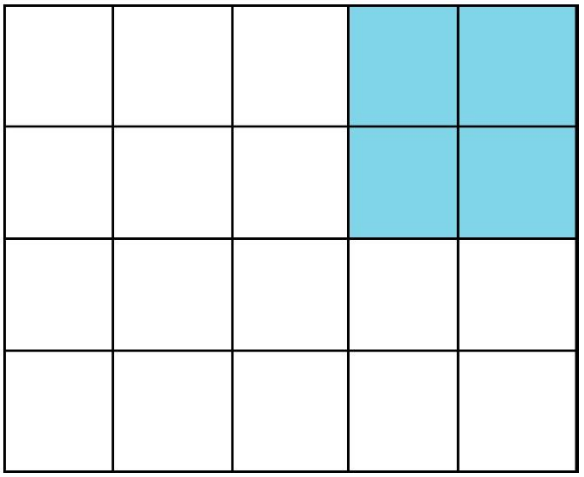
Retângulo 2



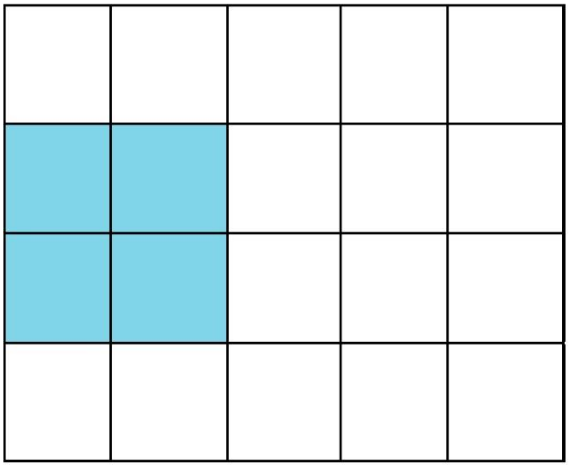
Retângulo 3



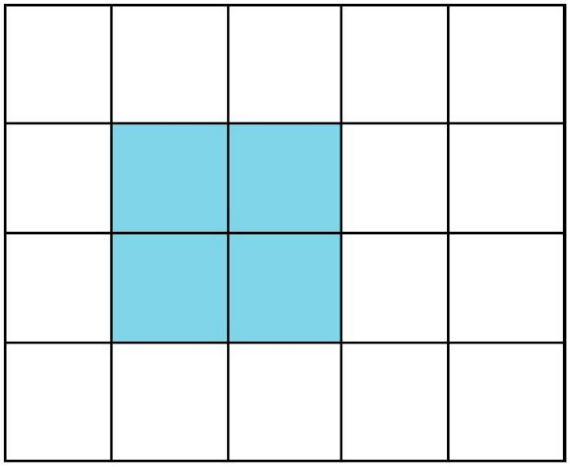
Retângulo 4



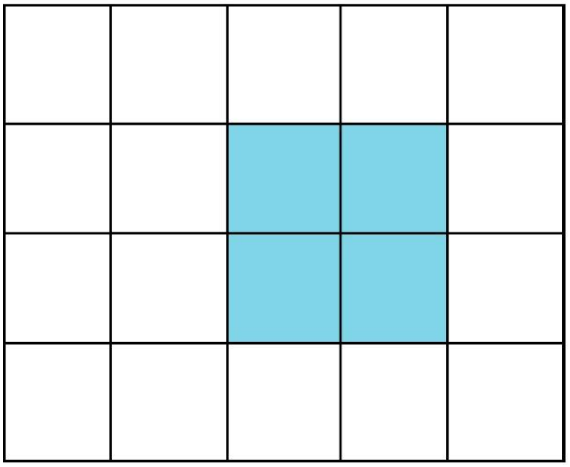
Retângulo 5



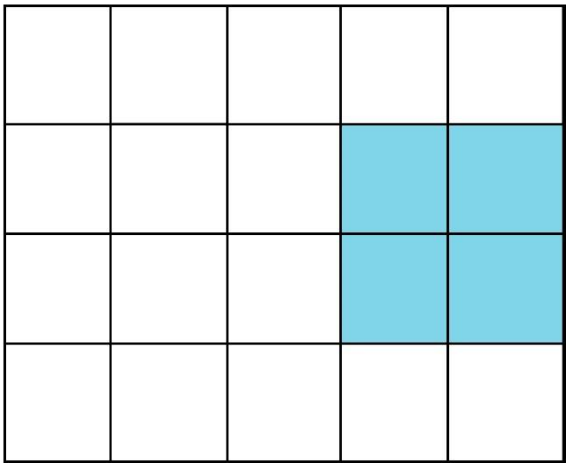
Retângulo 6



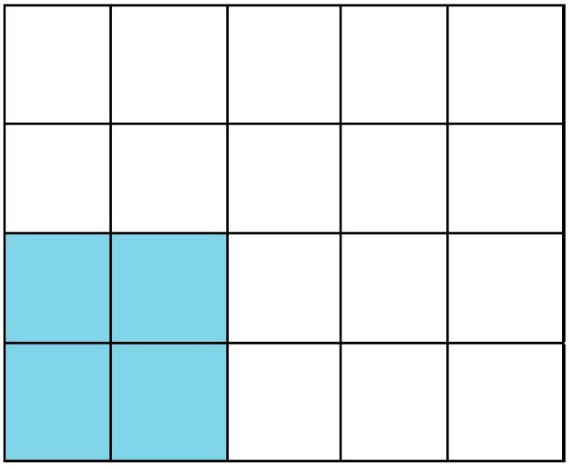
Retângulo 7



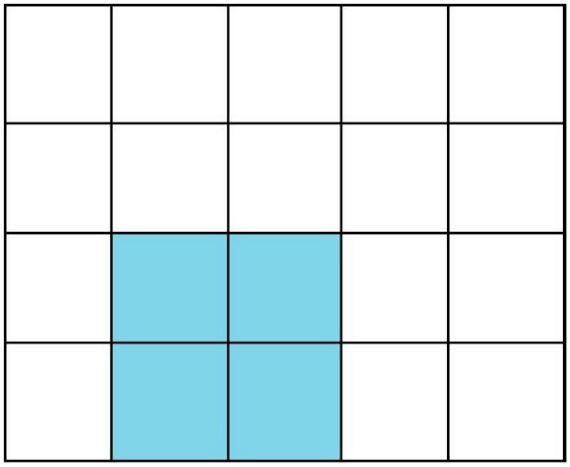
Retângulo 8



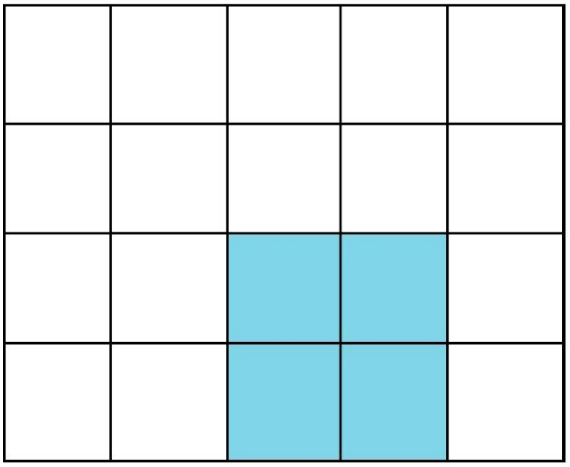
Retângulo 9



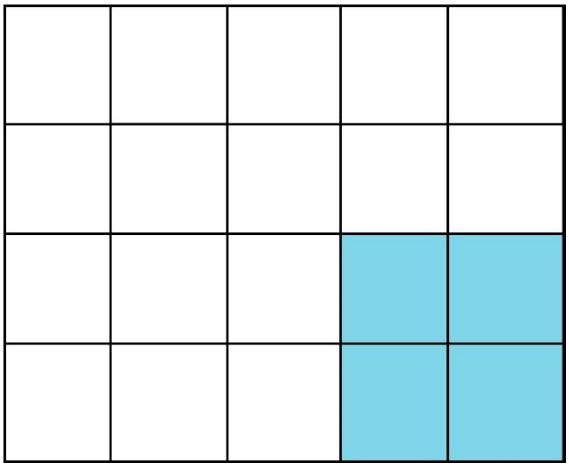
Retângulo 10



Retângulo 11



Retângulo 12



Portanto considerando que cada espaço preenchido pelo azul na ilustração na realidade será preenchido pelo número 1, dentre cada posição possível das 12 serão criadas as variáveis quadrado1, quadrado2 etc. que serão formadas por uma condição booleana onde se cada elemento das posições estiver preenchido por 1 dada a condição proposta será considerado quadrado e então adicionado à lista quadrados, veja abaixo o código da condição booleana.

```
public boolean isQuad (int x) {

		if(x == 4) {
			return true;
		}else {
			return false;
		}

	}
```

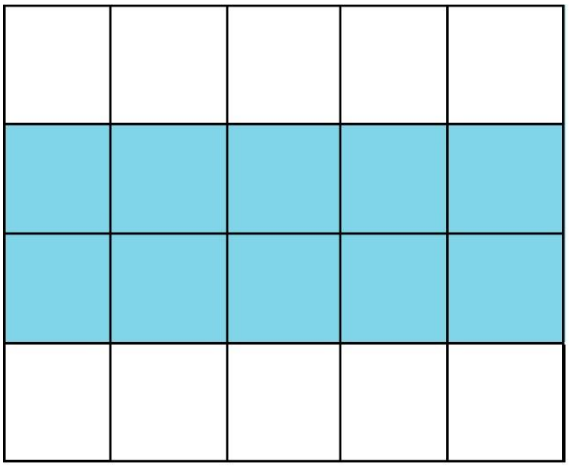
Este método Booleano irá receber um valor x que será praticamente o seguinte o programa compara as quatro variáveis possíveis para se formar o retângulo, caso o retângulo seja formado a variável retângulo1 ou 2 etc. irá receber 4 dado essa condição boolean se o valor x que é o quadrado for igual a 4 logicamente ele será um retângulo e irá ser considerado true e adicionado à lista retângulos.

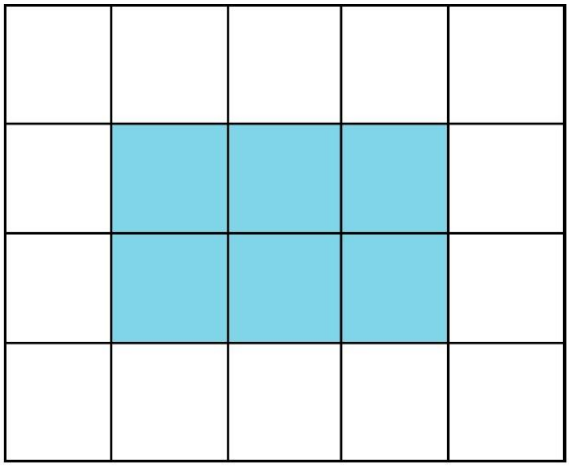
Veja isto abaixo:

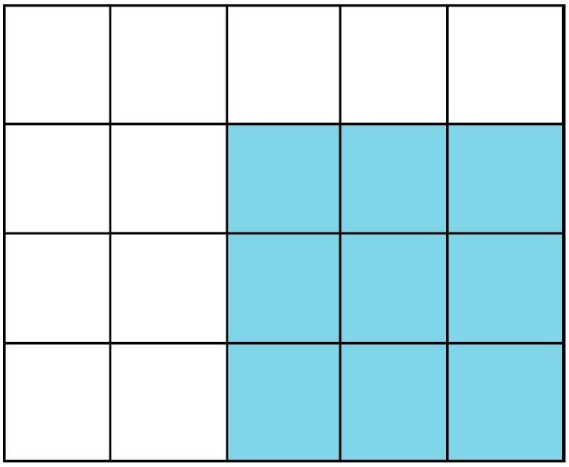
```
if (a == 1
   && b == 1
   && g == 1
                && f == 1) {

   setRetangulo1(4);

   if(isQuad(retangulo1) == true) {

         retangulos.add(retangulo1);
   }
```

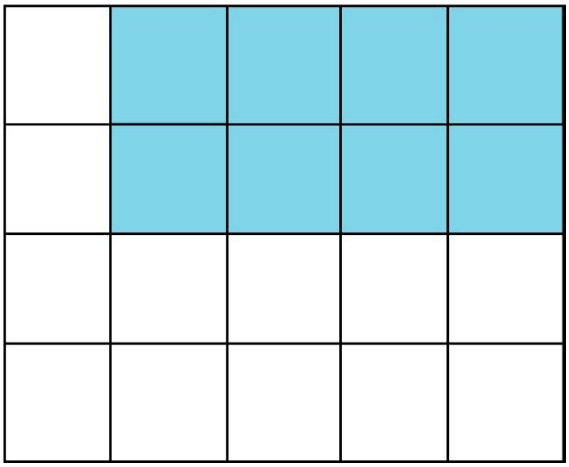
CLASSE RETANGLEVARIANT

Esta classe implementa variantes da classe RectangleFinder, por isso, então ela herda a classe RectangleFinder para formar variações de sua classe mãe, abaixo vou listar algumas possibilidades de variações da RectangleFinder para que consigam entender a minha linha de raciocínio, não irei apresentar todas para não sobrecarregar a documentação, porém algumas para que possam entender.









Como vemos na figura acima as variantes são combinações de outros retângulos da classe RectangleFinder, no desenrolar da aplicação a classe RectangleFinder serviu apenas para testes e para poder usar as mesmas variáveis booleanas para comparar se a posição dada forma retângulos e no dado caso variantes de retângulo, com essas variantes comparamos se a mesma combinação de posições dos retângulos da RectangleFinder mais outra combinação que forma retângulo é igual a 1 ou seja se é preenchido com 1 então se sim a variação recebe um valor específico, exemplo um possível retângulo na matriz formado por 6 posições de vetor recebe valor 6 e é adicionado em uma ArrayList que futuramente será usada para retornar o valor máximo e mínimo dentro da lista usando padrões lamba.

Veja um exemplo no código:

```
if (variante1 == 4 && variante9 == 4 && variante2 == 4 && variante10 == 4 &&
variante3 == 4
                                && variante11 == 4) {
                            variante33 = 16;

                            variantes.add(variante33);

                    }
```

Veja também a função lambda chamada futuramente para retornar o valor máximo dessa lista de variantes, lembrando que cada variante encontrada foi adicionada na lista que futuramente será usada para nos retornar seu valor máximo e mínimo.

```
Integer max = variantes.stream().mapToInt(v -> v).max().orElse(9000);

Integer min = variantes.stream().mapToInt(v -> v).min().orElse(9000);
```

Para entender o porque usei o número 9000 dentro dos parênteses de orElse vou explicar, caso dentro da lista de variantes o programa não encontre valor máximo ou mínimo tanto max e mínimo recebem 9000.

Porém logo após se esta condição de não haver possíveis retângulos visto que a lista variantes será preenchida apenas quando o sistema encontrar retângulo na matriz, então se a variável receber 9000 iremos em seguida para uma condição if else já que na condição 9000 não teremos retângulos teríamos o seguinte.

```
if(max == 9000 && min == 9000) {
                        System.out.println("Não existem retângulos possíveis");
                }
```

Caso contrário, ou seja se houverem retângulos ou um retângulo apenas o programa segue.

```
if (max != min) {
                        System.out.println("Considerando que é possível se formar
retângulos menores dentro de um retângulo maior");
                        System.out.println("O maior retângulo formado tem área: " + max);
                        System.out.println();
                        System.out.println("O menor retângulo formado tem área: " + min);
                }else {
                        System.out.println("O maior retângulo formado tem área: " + max);
                        System.out.println();
                }
```

Dentro de outra condição de validação se o valor máximo for diferente de mínimo, quer dizer que temos um valor máximo e outro mínimo possível então o programa trará um output para os dois, caso contrário apenas para o pequeno quadrado de área igual a 4 na condição else.

Vale ressaltar que foi avaliado todas as possibilidades de retângulos dentro da matriz[4][5] gerando mais de 50 variações de retângulos.

CLASSE PROGRAM(MAIN)

Pensando em organização e hierarquia trabalhei para que a classe main pudesse chamar apenas um método estático que chamaria então todas as outras classes do programa, como a classe main é formada apenas pela chamada de um método chamado PrintValues.Imprimir(); abordarei dele em seguida.

CLASSE PRINTVALUES

Antes de iniciarmos de fato falando da classe PrintValues, quero falar o quanto é importante termos demasiada empatia pelo nosso usuário, pois a aplicação é feita para ele, quanto menos empatia tivermos pelo usuário, menos interessado por nossa aplicação ele será, então ele irá procurar algo que lhe traga uma experiência mais interessante afinal ele é livre para escolher o que quiser, então o foco com o cliente é a alma de todo negócio e o princípio que cada programador deve ter.

Inicialmente a aplicação pedia para o usuário digitar um número dada a matriz e a matriz recebia os números retornava a matriz nas classes de serviço sendo estas a RectangleFinder e RectangleVariants, que o final foi apenas a RectangleVariants, porém ao pedir estes números o usuário não podia ter uma visualização do que está acontecendo, ficando a cargo de sua imaginação que na prática ele ficaria muito confuso.

Pensando neste problema desenvolvi para que a cada vez que o usuário digitasse um número a ser guardado no objeto sc de scanner logo abaixo seria mostrado a matriz preenchida.

Então desenvolvi em vários System.out.println um desenho da matriz fazendo várias estruturas condicionais para dizer que o que será mostrado na tela dada a posição da matriz.

Veja logo abaixo:

```
if (i == 0 && j == 1) {

                                        System.out.print("Coluna");
                                        System.out.println("  0 1 2 3 4");
                                        System.out.println("Linha 0|" + m[0][0] + "|"
+ m[0][1] + "|_|_|_|");
                                        System.out.println("Linha 1|_|_|_|_|_|");
                                        System.out.println("Linha 2|_|_|_|_|_|");
                                        System.out.println("Linha 3|_|_|_|_|_|");

                                    }

                                    if (i == 0 && j == 2) {

                                        System.out.print("Coluna");
                                        System.out.println("  0 1 2 3 4");
                                        System.out.println("Linha 0|" + m[0][0] + "|"
+ m[0][1] + "|" + m[0][2] + "|_|_|");
                                        System.out.println("Linha 1|_|_|_|_|_|");
                                        System.out.println("Linha 2|_|_|_|_|_|");
                                        System.out.println("Linha 3|_|_|_|_|_|");

                                    }
```

Dentro dessa classe fiz várias validações dentre elas realizar uma condição para exibir uma matriz sem variáveis caso o programa esteja iniciando, para isso fiz a variável de teste a int aux receber 999 e o programa testa que caso a variável aux receba 999 a matriz exibida seria a sem valores e como com o desenrolar do programa a variável aux receberia outros valores então o programa não entraria naquela estrutura condicional para exibir a matriz sem valores de novo.

Veja logo abaixo:

```
if (aux == 999) {
                                    System.out.println("Entre com um valor entre 0 e 1
para preencher a matriz de linha " + (i)
                                                + " e coluna " + (j));
                                    System.out.println("       _C0_C1_C2_C3_C4_");
                                    System.out.println("Linha 0|__|__|__|__|__|");
                                    System.out.println("Linha 1|__|__|__|__|__|");
                                    System.out.println("Linha 2|__|__|__|__|__|");
                                    System.out.println("Linha 3|__|__|__|__|__|");
```

```
}
```

Para entender a função da variável aux vamos abordar por etapas:

1º Variável recebe o valor digitado pelo usuário desfazendo a condição do 999 e criando outras possibilidades a seguir.

2º Logo após é testado se o valor digitado é diferente de 0 e 1 que é o que esperamos para uma matriz binária caso isto aconteça, enquanto o valor digitado for esse dentro desse loop é feito uma condicional para informar que o usuário digite o valor correto e guardar dentro do condicional novamente a variável aux, visto que dentro da estrutura condicional as variáveis não são universais, então se o valor digitado for finalmente 0 ou 1 a matriz[i][j] finalmente recebera o que está dentro da auxiliar.

Veja logo abaixo:

```
	System.out.println(
					"Entre com um valor entre 0 e 1 para
preencher a matriz de linha " + (i) + " e coluna " + (j));


				aux = sc.nextInt();
				if (aux != 0 && aux != 1) {

					while (aux != 0 && aux != 1) {
						if (aux != 0 && aux != 1) {
							System.out.println("digite correto");

							System.out.println("Entre com um valor
entre 0 e 1 para preencher a matriz de linha " + (i)
									+ " e coluna " + (j));

							aux = sc.nextInt();

						}
						if (aux == 0) {
							m[i][j] = (int) aux;
						} else if (aux == 1) {
							m[i][j] = (int) aux;
						}
					}
```

Caso o usuário de fato digitou um valor binário sem erros então a matriz[i][j] recebe a valor da variável auxiliar e para cada condição do vetor ele receberá o design de System.out.println adequado.

Veja abaixo alguns exemplos:

```
else {
                                m[i][j] = (int) aux;
                                if (i == 0 && j == 0) {

                                      ;
                                      System.out.print("Coluna");
                                      System.out.println("  0 1 2 3 4");
                                      System.out.println("Linha 0|" + m[0][0] +
"|_|_|_|_|");
                                      System.out.println("Linha 1|_|_|_|_|_|");
                                      System.out.println("Linha 2|_|_|_|_|_|");
                                      System.out.println("Linha 3|_|_|_|_|_|");

                                }
                                if (i == 0 && j == 1) {

                                      System.out.print("Coluna");
                                      System.out.println("  0 1 2 3 4");
                                      System.out.println("Linha 0|" + m[0][0] + "|"
+ m[0][1] + "|_|_|_|");
                                      System.out.println("Linha 1|_|_|_|_|_|");
                                      System.out.println("Linha 2|_|_|_|_|_|");
                                      System.out.println("Linha 3|_|_|_|_|_|");

                                }

                                if (i == 0 && j == 2) {

                                      System.out.print("Coluna");
                                      System.out.println("  0 1 2 3 4");
                                      System.out.println("Linha 0|" + m[0][0] + "|"
+ m[0][1] + "|" + m[0][2] + "|_|_|");
                                      System.out.println("Linha 1|_|_|_|_|_|");
                                      System.out.println("Linha 2|_|_|_|_|_|");
                                      System.out.println("Linha 3|_|_|_|_|_|");

                                }
```

Após guardar as variáveis digitadas o programa irá instanciar um objeto da classe RectangleVariants para alocar o objeto dentro do seu método RectangleVariantFinder e por fim termos o resultado que esperamos.

Veja logo abaixo:

```
RectangleVariants rv = new RectangleVariants();
            rv.RectangleVariantFinder(m);
            sc.close();
```

CONCLUSÃO

Com este programa Podemos identificar a utilização de métodos estáticos, orientação a objetos, herança, encapsulamento, lógica de programação avançada, arrays, listas, expressões lambda, tudo de acordo com nossa demanda, o desafio foi muito interessante e minha proposta foi abordar um pouco de cada ferramenta que uso como programador, além de me preocupar com a melhor experiência para o usuário.