1. Aula 4

1.1. Operadores matemáticos

Os operadores aritméticos são operadores binários, ou seja, funcionam com dois operandos. Por exemplo, a expressão “a + 1” contém o operador binário “+” (mais) e os dois operandos “a” e “1”.

Operação	Operador	Expressão algébrica	Expressão Java
Adição	+	a + 1	a +1
Subtração	-	b -2	b -2
Multiplicação	*	cm	c * m
Divisão	/	d / e	d / e
Resto	%	f mod g	f % g

Observação importante: a divisão de inteiros produz um quociente do tipo inteiro. Quando possuímos o número 1 maior que o número 2, por exemplo, a expressão 9 / 6 no resultado é interpretado como 1, e a expressão 23 / 8 é avaliada como 2, ou seja, a parte fracionária em uma divisão de inteiros é descartada, não contendo nenhum arredondamento.

O módulo (%) fornece o resto da divisão, na expressão “x % y”, o resultado é o restante depois que x é dividido por y. Sendo assim, na expressão “7 % 4”, o resultado é 3 e “17 % 5” o resultado produz 2. Esse operador é mais utilizado com operandos inteiros, mas também pode ser utilizado com outros tipos.

Precedência de operadores

Os operadores possuem regras que são aplicadas nas expressões aritméticas do Java, que são as mesmas seguidas em álgebra. Quando dizemos que os operadores são aplicados da esquerda para a direita, estamos nos referindo à sua associatividade.

Operadores de multiplicação, divisão e módulo são aplicadas primeiro. Por exemplo, quando aparecer uma expressão com várias dessas operações, elas serão aplicadas da esquerda para a direita.

As operações de adição e subtração são aplicadas em seguida.

Abaixo uma tabela de referência dos operadores e suas ordens de avaliação.

Operador	**Operação**	**Ordem de avaliação(precedência)**
* / %	Multiplicação Divisão Resto	Avaliado primeiro. Se houver vários operadores desse tipo serão avaliados da esquerda para a direita
+ -	Adição Subtração	Avaliado em seguida. Se houver vários operadores desse tipo, serão avaliados da esquerda para a direita.
=	Atribuição	Avaliado por último

Listagem: Avaliação da precedência dos operadores

```
public class Avalia_Precedencia {
   public static void main(String[] args) {
      int a = 30;
      int b = 5;
      int c = 10;
      int total = (a + b + c) / 10;
      System.out.println("O resultado = "+total);
   }
}
```

1.2. Operadores

Os operadores de igualdade verificam se o valor ou o resultado da expressão lógica à esquerda é igual (“==”) ou diferente (“!=”) ao da direita, retornando um valor booleano.

Veja o exemplo:

```
int idadeJoao=20;
int idadeMaria=22;

if(idadeJoao == idadeMaria){
        System.out.println("idades iguais");
}else{
        System.out.println("idades diferentes");
}
```

Esse código verifica se duas variáveis contêm o mesmo valor e imprimem o resultado. Uma vez que as variáveis IdadeJoao e IdadeMaria possuem valores diferentes, o trecho de código presente no else será executado.

A tabela abaixo apresenta os **operadores de igualdade** do Java:

==	Utilizado quando desejamos verificar se uma variável é igual a outra.
!=	Utilizado quando desejamos verificar se uma variável é diferente de outra.

Operadores relacionais

Os operadores relacionais, assim como os de igualdade, avaliam dois operandos. Nesse caso, mais precisamente, definem se o operando à esquerda é menor, menor ou igual, maior ou maior ou igual ao da direita, retornando um valor booleano.

Veja o exemplo:

```
int idadeJoao = 20;
int idadeMaria = 22;

if (idadeJoao > idadeMaria) {
        System.out.println(“maior”);
}
```

```
if (idadeJoao >= idadeMaria) {
        System.out.println(“maior ou igual”);
}

if (idadeJoao < idadeMaria) {
        System.out.println(“menor”);
}

if (idadeJoao <= idadeMaria) {
        System.out.println(“menor ou igual”);
}
```

Esse código realiza uma série de comparações entre duas variáveis para determinar o que será impresso no console. Uma vez que o valor da variável idadeJoao é menor que idadeMaria serão impressas as mensagens “menor” e “menor ou igual”.

Opções de operadores relacionais

A tabela abaixo apresenta os operadores relacionais do Java:

>	Utilizado quando desejamos verificar se uma variável é maior que outra.
>=	Utilizado quando desejamos verificar se uma variável é maior ou igual a outra
<	Utilizado quando desejamos verificar se uma variável é menor que outra.
<=	Utilizado quando desejamos verificar se uma variável é menor ou igual a outra.

Operadores lógicos

Os operadores lógicos representam o recurso que nos permite criar expressões lógicas maiores a partir da junção de duas ou mais expressões. Para isso, aplicamos as operações lógicas E (representado por “&&”) e OU (representado por “||”).

Exemplo de uso:

```
if((1 == (2 -1)) && (2 == (1 + 1))){
   System.out.println(“Ambas as expressões são verdadeiras”);
}
```

Opções de operadores de lógicos

A tabela abaixo apresenta os operadores lógicos do Java:

&&	Utilizado quando desejamos que as duas expressões sejam verdadeiras.
||	Utilizado quando precisamos que pelo menos um das expressões seja verdadeira.

1.3. Estrutura condicional IF()

As estruturas de condição possibilitam ao programa tomar decisões e alterar o seu fluxo de execução. É por meio delas que podemos dizer ao sistema: “execute a instrução A caso a expressão X seja verdadeira; caso contrário, execute a instrução B”.

A estrutura condicional if/else permite ao programa avaliar uma expressão como sendo verdadeira ou falsa e, de acordo com o resultado dessa verificação, executar uma ou outra rotina.

Na linguagem Java, o tipo resultante dessa expressão deve ser sempre um boolean, pois diferentemente das demais, o Java não converte null ou inteiros como 0 e 1 para os valores true ou false.

Sintaxe do if/else:

```
if (expressão booleana) {
   // bloco de código 1
} else {
   // bloco de código 2
}
```

As instruções presentes no bloco de código 1 serão executadas caso a expressão booleana seja verdadeira. Do contrário, serão executadas as instruções presentes no bloco de código 2.

O Java utiliza as chaves como delimitadores de bloco, sendo que elas têm a função de agrupar um conjunto de instruções. Apesar do uso desses delimitadores ser opcional caso haja apenas uma linha de código, ele é recomendado, pois facilita a leitura e manutenção do código, tornando-o mais legível.

Complementar ao if/else temos o operador else if. Esse recurso possibilita adicionar uma nova condição à estrutura de decisão para atender a lógica sendo implementada.

Sintaxe do if/else com else if:

```
if (expressão booleana 1) {
   // bloco de código 1
} else if (expressão booleana 2) {
   // bloco de código 2
} else {
   // bloco de código 3
}
```

Dessa forma, se a expressão booleana 1 for verdadeira, o bloco de código 1 será executado. Caso seja falsa, o bloco de código 1 será ignorado e será testada a expressão booleana 2. Se ela for verdadeira, o bloco de código 2 será executado. Caso contrário, o programa vai ignorar esse bloco de código e executar o bloco 3, declarado dentro do else.

Podemos utilizar quantos else if forem necessários. Entretanto, o else deve ser adicionado apenas uma vez, como alternativa ao caso de todos os testes terem falhado.

Exemplo de condicional IF simples:

```
int qtdeVenda;
int qtdeEstoque;
int estoqueMinimo;

estoqueMinimo=50;
qtdeVenda=140;
qtdeEstoque=170;

int total=qtdeEstoque - qtdeVenda;
System.out.println("Quantidade atual é de: "+total);
if(total>estoqueMinimo){
 System.out.println("Repor estoque mínimo de 50 unidades");
}else{
 System.out.println("Estoque em dia");
}
```

No exemplo acima, foram criadas três variáveis, quantidade de venda, quantidade no estoque e estoque mínimo, uma outra variável foi criada para calcular a diferença entre a quantidade vendida e o que tem no estoque.

Uma condição foi criada para verificar se a quantidade vendida baixou o estoque. Se isso aconteceu, uma mensagem informa que o mínimo é de 50 unidades, caso contrário, o estoque está em dia.

Criando um exemplo utilizando o operador lógico “&&”.

No exemplo abaixo, estaremos fazendo um teste em um cadastro de revendedor, onde o usuário deve morar na cidade de Montenegro e ser maior de idade para ser aprovado no sistema.

```
String cidade;
Integer idade;
EditText resultado;

resultado=(EditText) findViewById(R.id.resultado);
cidade="Montenegro";
idade=17;

if(cidade=="Montenegro" && idade>17){
  resultado.setText("Dados confirmados;
}else{
  resultado.setText("Cidade ou idade não conferem com o exigido");
}
```

Para o exemplo acima funcionar, deve ser inserido um componente do tipo Plain Text.