MÓDULO PROGRAMAÇÃO JAVA

Fundamentos de Programação com Java

Professor: Marcos Brandão

Versão: 2026

APRESENTAÇÃO

Java é uma linguagem de programação utilizada no desenvolvimento de sistemas, aplicações corporativas, serviços, APIs e diversos outros tipos de software.

Neste módulo, nosso objetivo não será apenas conhecer comandos da linguagem Java.

Vamos aprender a compreender a lógica existente por trás de um programa.

Para isso, utilizaremos o Simulador de Programação Java, no qual será possível visualizar e modificar dois arquivos principais:

Main.java

Responsável principalmente pela interação com o usuário e pela interface da aplicação.

Processamento.java

Responsável pelos cálculos, decisões e regras utilizadas pelo programa.

Durante as aulas, você poderá alterar os códigos e imediatamente observar o efeito das modificações.

COMO ESTUDAR COM ESTE MÓDULO

Em praticamente todas as aulas seguiremos esta sequência:

1. CONHECER
Aprender o conceito.

2. OBSERVAR
Analisar um código pronto.

3. PREVER
Tentar descobrir o resultado antes de executar.

4. EXECUTAR
Testar o código no simulador.

5. MODIFICAR
Alterar partes do programa.

6. EXPERIMENTAR
Executar novamente e comparar os resultados.

7. PRATICAR
Resolver exercícios.

8. DESAFIAR-SE
Modificar o programa para criar um comportamento diferente.

UNIDADE 1 — INTRODUÇÃO À PROGRAMAÇÃO JAVA

1.1 O que é programação?

Programar significa criar uma sequência de instruções que serão executadas pelo computador para resolver determinado problema.

Imagine, por exemplo, que precisamos criar um programa para calcular a área de um retângulo.

O programa deverá:

1. **receber a base;**
2. **receber a altura;**
3. **multiplicar base × altura;**
4. **armazenar o resultado;**
5. **apresentar o resultado ao usuário.**

Podemos representar isso assim:

ENTRADA → PROCESSAMENTO → SAÍDA

Exemplo:

Base = 5

Altura = 4

↓

5 × 4

↓

Área = 20

Esse conceito será encontrado em praticamente todos os programas deste módulo.

1.2 O que é Java?

Java é uma linguagem de programação criada originalmente pela Sun Microsystems e posteriormente incorporada ao ecossistema da Oracle.

Um programa Java normalmente é escrito em arquivos com extensão:

.java

Exemplos:

Main.java

Processamento.java

Aluno.java

Produto.java

Calculadora.java

No desenvolvimento Java real, o código-fonte pode ser compilado para uma representação chamada bytecode, normalmente executada pela Java Virtual Machine — JVM.

Neste módulo utilizaremos um simulador didático que interpreta os recursos trabalhados nas aulas, permitindo experimentar os principais conceitos diretamente pelo navegador.

1.3 Nosso primeiro programa

Um exemplo clássico em Java é:

```
public class Main {

    public static void main(String[] args) {

        System.out.println("Olá Mundo");

    }

}
```

A instrução:

System.out.println()

é utilizada para apresentar informações no console.

Entretanto, nosso simulador utiliza também componentes gráficos Java Swing.

Por isso encontraremos instruções como:

```
JFrame frame =
    new JFrame("Calculadora Java");
```

e:

```
JOptionPane.showMessageDialog(
    frame,
    "Resultado"
);
```

1.4 Main.java e Processamento.java

No nosso simulador trabalharemos principalmente com dois arquivos.

Main.java

É onde encontraremos a criação da interface, leitura dos dados e chamadas ao processamento.

Exemplo:

```
double resultado =
    Processamento.calcular(
        v1,
        v2
    );
```

Processamento.java

É onde colocaremos cálculos e regras.

Exemplo:

```
public static double calcular(
    double valor1,
    double valor2
```

) {

double resultado =

valor1 + valor2;

return resultado;

}

Podemos visualizar a comunicação assim:

USUÁRIO

↓

Main.java

↓

Processamento.java

↓

resultado

↓

Main.java

↓

USUÁRIO

PRÁTICA 1 — CONHECENDO O SIMULADOR

Abra o exemplo:

01 – Calculadora

Observe os painéis:

- **Main.java**
- **Processamento.java**
- **Interface gráfica**
- **Resultado**
- **Variáveis**
- **Processamento**

- **Comparações/decisões**

Digite:

Valor 1: 10

Valor 2: 5

Antes de executar, responda:

Qual será o resultado?

Agora execute.

Observe principalmente o painel:

⚙️ Processamento

Localize:

```
double resultado =
    valor1 + valor2;
```

Altere:

+

para:

*****

Execute novamente.

Pergunta

Por que o programa agora apresenta outro resultado mesmo sem modificarmos o Main.java?

UNIDADE 2 — VARIÁVEIS E TIPOS DE DADOS

2.1 O que é uma variável?

Uma variável é um espaço utilizado pelo programa para armazenar temporariamente um valor.

Exemplo:

```
int idade = 25;
```

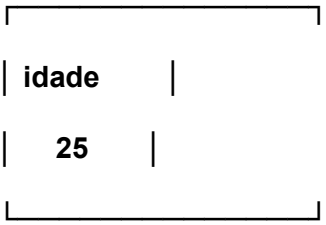
Temos:

int → tipo

idade → nome da variável

25 → valor

Podemos imaginar:



2.2 Principais tipos básicos

Alguns tipos frequentemente encontrados em Java são:

Tipo	Utilização	Exemplo
int	números inteiros	10
double	números decimais	7.5
float	números decimais	10.5
long	inteiros maiores	100000
boolean	verdadeiro ou falso	true
char	um caractere	'A'

String	textos	"Marcos "

Exemplo:

```
String nome = "Carlos";

int idade = 20;

double altura = 1.75;

boolean aprovado = true;
```

PRÁTICA 2 — OBSERVANDO VARIÁVEIS

Abra:

08 – Área do retângulo

Observe:

```
double b =
    Double.parseDouble(
        base.getText()
    );
```

O programa recebe o conteúdo digitado no campo `base` e o transforma em um número.

Depois encontramos:

```
double resultado =
    Processamento.calcularArea(
        b,
        a
    );
```

No Processamento.java:

```
double area =
    base * altura;
```

Teste:

Base: 5

Altura: 4

Previsão:

Área = ?

Execute e observe o painel Variáveis.

Depois teste:

Base: 10

Altura: 3

ATIVIDADE 1

Sem executar inicialmente, determine os resultados:

Base = 7

Altura = 2

Base = 8

Altura = 5

Base = 12

Altura = 10

Depois utilize o simulador para conferir suas respostas.

DESAFIO

Transforme o programa de área do retângulo em um cálculo diferente.

Por exemplo:

```
double area =
    base * altura;
```

pode ser alterado experimentalmente para:

```
double resultado =
    (base + altura) * 2;
```

Observe o que acontece.

Qual cálculo essa nova expressão está realizando?

UNIDADE 3 — OPERADORES ARITMÉTICOS

Java possui operadores para realização de cálculos.

Operador	Operação
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão

Exemplo:

```
double resultado =
    valor1 + valor2;
```

Para multiplicação:

double resultado =

valor1 * valor2;

Para divisão:

double resultado =

valor1 / valor2;

3.1 O operador %

O operador % retorna o resto de uma divisão.

Exemplo:

10 ÷ 2

resto = 0

Logo:

10 % 2

resulta em:

0

Agora:

7 ÷ 2

resto = 1

Portanto:

7 % 2

resulta em:

1

Isso permite descobrir se determinado número é par ou ímpar.

PRÁTICA 3 — PAR OU ÍMPAR

Abra:

05 – Par ou Ímpar

Observe:

```
int resto =
    numero % 2;
```

Depois:

```
if (resto == 0) {

    return "Número PAR";

} else {

    return "Número ÍMPAR";
}
```

Teste:

8

Depois:

7

Observe os painéis Variáveis e Comparações/decisões.

ATIVIDADE 2

Antes de executar, determine se os números serão classificados como PAR ou ÍMPAR:

10

17

24

31

100

Depois confira no simulador.

UNIDADE 4 — ENTRADA E SAÍDA DE DADOS

Programas frequentemente precisam receber informações do usuário.

No simulador encontramos componentes como:

```
JTextField nome =
    new JTextField();
```

Esse componente representa um campo de entrada.

Para recuperar seu conteúdo:

```
nome.getText()
```

4.1 Convertendo textos em números

Um campo de texto fornece originalmente um texto.

Para transformar o conteúdo em número decimal:

```
Double.parseDouble(
    peso.getText()
);
```

Para inteiro:

```
Integer.parseInt(
    idade.getText()
);
```

4.2 Saída de dados

Podemos apresentar uma mensagem utilizando:

```
JOptionPane.showMessageDialog(
    frame,
    "Olá!"
);
```

Também podemos apresentar variáveis:

```
JOptionPane.showMessageDialog(
    frame,
    "Resultado: " + resultado
);
```

O operador +, nesse caso, pode realizar a concatenação entre texto e valor.

PRÁTICA 4 — CADASTRO

Abra:

06 – Cadastro

Digite:

Nome: Ana

Idade: 25

Observe:

String n =

nome.getText();

e:

int i =

Integer.parseInt(

idade.getText()

);

O primeiro dado é tratado como texto.

O segundo é transformado em número inteiro.

ATIVIDADE 3

No Processamento.java localize:

String resultado =

"Nome: " + nome +

"\nIdade: " + idade;

Modifique para:

String resultado =

"Aluno: " + nome +

"\nIdade informada: " + idade;

Execute novamente.

Desafio

Tente modificar as mensagens apresentadas sem alterar os cálculos do programa.

UNIDADE 5 — OPERADORES RELACIONAIS

Programas também precisam comparar valores.

Para isso utilizamos operadores relacionais.

Operador	Significado
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual
==	igual
!=	diferente

Exemplo:

idade >= 18

A expressão pergunta:

idade é maior ou igual a 18?

O resultado será:

true

ou:

false

UNIDADE 6 — ESTRUTURAS DE DECISÃO

Uma das estruturas mais importantes da programação é:

if

Ela permite que o programa tome decisões.

Exemplo:

```
if (idade >= 18) {

    return "Maior de idade";

} else {

    return "Menor de idade";
}
```

Podemos ler isso como:

SE idade for maior ou igual a 18

    Maior de idade

SENÃO

    Menor de idade

PRÁTICA 5 — IDADE

Abra:

04 – Maior ou menor de idade

Teste:

17

Depois:

18

Depois:

25

Observe o painel:

🔎 Comparações / decisões

Ele permitirá visualizar algo semelhante a:

idade >= 18 → FALSO

ou:

idade >= 18 → VERDADEIRO

EXPERIMENTO

Altere:

if (idade >= 18)

para:

if (idade >= 21)

Teste novamente:

18

20

21

25

Pergunta

O que mudou na regra do sistema?

UNIDADE 7 — IF, ELSE IF E ELSE

Nem sempre existem apenas duas possibilidades.

Considere a classificação do IMC.

Podemos ter:

Abaixo do peso

Peso normal

Sobrepeso

Obesidade

Precisamos, portanto, verificar várias condições.

Java permite utilizar:

if

else if

else

PRÁTICA 6 — IMC

Abra:

03 – Calculadora de IMC

Observe:

```
if (imc < 18.5) {

    return "Abaixo do peso";

} else if (imc < 25) {
```

```
    return "Peso normal";

} else if (imc < 30) {

    return "Sobrepeso";

} else {

    return "Obesidade";
}
```

O programa verifica as condições em sequência.

PREVISÃO

Digite:

Peso: 69.8

Altura: 1.60

Antes de executar, tente prever:

IMC = ?

Classificação = ?

Agora execute.

Observe as comparações realizadas.

ATIVIDADE 4 — ALTERANDO REGRAS

Localize:

else if (imc < 25)

Altere temporariamente para:

else if (imc < 27)

Execute novamente utilizando os mesmos dados.

Responda

1. **O cálculo do IMC mudou?**
2. **A classificação mudou?**
3. **Qual parte do sistema foi modificada?**
4. **Por que alterar uma condição pode modificar o comportamento do sistema?**

Depois restaure o exemplo original.

UNIDADE 8 — MÉTODOS

Métodos são blocos de código criados para executar determinada tarefa.

Observe:

```
public static double calcularMedia(
   double nota1,
   double nota2
) {

   double media =
      (nota1 + nota2) / 2;

   return media;
}
```

O método se chama:

calcularMedia

Ele recebe:

nota1

nota2

realiza:

(nota1 + nota2) / 2

e retorna:

media

8.1 Parâmetros

Em:

calcularMedia(

double nota1,

double nota2

)

`nota1` **e** `nota2` **são parâmetros.**

Eles permitem que diferentes valores sejam enviados ao método.

Por exemplo:

calcularMedia(8, 6)

produzirá:

7

Enquanto:

calcularMedia(10, 8)

produzirá:

9

O método é o mesmo.

Os dados são diferentes.

8.2 Return

Observe:

return media;

`return` devolve um resultado para quem chamou o método.

Fluxo:

Main.java

calcularMedia(8, 6)

↓

Processamento.java

(8 + 6) / 2

↓

return 7

↓

Main.java

media = 7

PRÁTICA 7 — MÉTODOS

Abra:

02 – Média do aluno

Digite:

Nota 1: 8

Nota 2: 6

Antes de executar:

Qual será a média?

Qual será a situação?

Execute.

Observe que dois métodos são utilizados:

Processamento.calcularMedia()

e:

Processamento.verificarSituacao()

O primeiro calcula.

O segundo toma uma decisão.

EXPERIMENTO

Altere:

if (media >= 7)

para:

if (media >= 6)

Teste:

Nota 1: 6

Nota 2: 6

Depois:

Nota 1: 5

Nota 2: 7

Explique o resultado.

UNIDADE 9 — SEPARAÇÃO ENTRE INTERFACE E PROCESSAMENTO

Uma prática importante no desenvolvimento de sistemas é separar responsabilidades.

No nosso simulador:

Main.java

fica principalmente responsável por:

- **criar a interface;**
- **receber informações;**
- **chamar métodos;**
- **apresentar resultados.**

Enquanto:

Processamento.java

fica responsável por:

- **cálculos;**
- **validações;**
- **decisões;**
- **regras.**

9.1 Exemplo

Main.java:

double resultado =

```
    Processamento.calcular(
        v1,
        v2
    );
```

Processamento.java:

```
public static double calcular(
    double valor1,
    double valor2
) {

    double resultado =
        valor1 + valor2;

    return resultado;
}
```

O Main não precisa saber detalhadamente como o cálculo é realizado.

Ele solicita:

calcular(v1, v2)

e recebe a resposta.

PRÁTICA 8 — FRONT-END E BACK-END

Abra:

01 – Calculadora

Não altere o Main.java.

Modifique somente:

Processamento.java

Troque:

valor1 + valor2

por:

valor1 - valor2

Execute.

Depois:

valor1 * valor2

Execute.

Depois:

valor1 / valor2

Execute.

Responda

Foi necessário modificar a interface para mudar a regra do programa?

Explique por que separar interface e processamento pode facilitar a manutenção de um sistema.

UNIDADE 10 — EXPRESSÕES MATEMÁTICAS

Observe o programa de temperatura:

```
double fahrenheit =
    (celsius * 9 / 5) + 32;
```

Uma expressão pode combinar:

- **variáveis;**
- **números;**
- **operadores;**
- **parênteses.**

Os parênteses ajudam a determinar a ordem das operações.

PRÁTICA 9 — TEMPERATURA

Abra:

07 – Conversão de temperatura

Teste:

0

Previsão:

Fahrenheit = ?

Execute.

Depois:

100

O resultado esperado é:

212 °F

DESAFIO

Observe:

(celsius * 9 / 5) + 32

Explique com suas próprias palavras o que acontece em cada parte da expressão.

UNIDADE 11 — INTERFACE GRÁFICA COM SWING

Nos exemplos utilizamos componentes da biblioteca Java Swing.

Entre eles:

JFrame

JLabel

JTextField

JButton

JOptionPane

JFrame

Representa uma janela.

```
JFrame frame =
    new JFrame("Calculadora");
```

JLabel

Apresenta um texto.

```
JLabel label =
    new JLabel("Digite sua idade:");
```

JTextField

Cria um campo para digitação.

```
JTextField idade =
    new JTextField();
```

JButton

Representa um botão.

```
JButton verificar =
    new JButton("Verificar");
```

JOptionPane

Pode apresentar mensagens.

```
JOptionPane.showMessageDialog(
    frame,
```

"Resultado"

);

PRÁTICA 10 — MODIFICANDO A INTERFACE

Abra:

01 – Calculadora

No Main.java localize:

new JFrame("Calculadora Java");

Troque para:

new JFrame("Minha Calculadora");

Observe a interface.

Depois altere:

new JLabel("Valor 1:");

para:

new JLabel("Primeiro número:");

E:

new JButton("Calcular");

para:

new JButton("Executar cálculo");

Observe como o Main.java controla elementos da interface.

UNIDADE 12 — LEITURA DO FLUXO DE UM PROGRAMA

Agora podemos compreender um programa completo.

Considere a calculadora.

ETAPA 1 — Entrada

O usuário digita:

10

5

ETAPA 2 — Main

O programa recupera:

```
double v1 =
    Double.parseDouble(
        valor1.getText()
    );
```

ETAPA 3 — Chamada

```
Processamento.calcular(
    v1,
    v2
);
```

ETAPA 4 — Processamento

```
double resultado =
    valor1 + valor2;
```

ETAPA 5 — Retorno

```
return resultado;
```

ETAPA 6 — Saída

```
JOptionPane.showMessageDialog(
    frame,
    "Resultado: " + resultado
);
```

Portanto:

ENTRADA

↓

VARIÁVEIS

↓

MÉTODO

↓

PROCESSAMENTO

↓

RETURN

↓

RESULTADO

ATIVIDADE DE ANÁLISE

Abra o exemplo:

02 – Média do aluno

Sem executar, identifique:

1. **Quais são os campos de entrada?**
2. **Quais variáveis recebem os valores?**
3. **Qual método calcula a média?**
4. **Quais parâmetros são enviados?**
5. **Qual expressão matemática é utilizada?**
6. **Qual método verifica a situação?**
7. **Qual condição é utilizada para aprovação?**
8. **O que acontece quando a condição é falsa?**
9. **Qual informação é apresentada ao usuário?**

Depois execute e confira suas respostas utilizando os painéis do simulador.

UNIDADE 13 — DEPURAÇÃO E TESTES

Programar não significa apenas escrever código.

Também precisamos testar.

Um programa pode possuir erros que aparecem somente em determinadas situações.

Por exemplo:

valor1 / valor2

O que acontece matematicamente se:

valor2 = 0

Por isso devemos experimentar diferentes entradas.

13.1 Teste de limite

Considere:

if (idade >= 18)

Valores interessantes para testar:

17

18

19

Eles verificam exatamente a fronteira da condição.

Para:

if (media >= 7)

podemos testar:

6.9

7

7.1

PRÁTICA 11 — TESTANDO LIMITES

Utilize:

02 – Média do aluno

Crie combinações de notas que produzam:

6.9

7.0

7.1

Registre para cada teste:

Entradas:

Média:

Condição:

Resultado:

UNIDADE 14 — ERROS COMUNS EM PROGRAMAÇÃO

Durante o desenvolvimento, erros são normais.

Alguns exemplos:

Operador incorreto

valor1 - valor2

quando deveria ser:

valor1 + valor2

Condição incorreta

media > 7

quando a regra deveria permitir exatamente 7:

media >= 7

Fórmula incorreta

Errado:

peso / altura

Para o cálculo trabalhado no exemplo do IMC:

peso / (altura * altura)

Tipo inadequado

int altura = 1.75;

não é apropriado para armazenar esse valor.

Um tipo decimal é necessário:

double altura = 1.75;

ATIVIDADE — CAÇA AO ERRO

O professor poderá modificar um dos exemplos do simulador propositalmente.

O aluno deverá:

1. **executar;**
2. **identificar o resultado incorreto;**
3. **localizar o possível erro;**
4. **corrigir o código;**
5. **executar novamente;**

6. **explicar a correção.**

UNIDADE 15 — PROJETO INTEGRADOR

DESAFIO: SISTEMA DE SITUAÇÃO DO ALUNO

Utilize como base:

02 – Média do aluno

O programa atual possui:

APROVADO

REPROVADO

Vamos criar três situações:

APROVADO

RECUPERAÇÃO

REPROVADO

Regras:

Média >= 7

APROVADO

Média >= 5

RECUPERAÇÃO

Média < 5

REPROVADO

O Processamento.java deverá utilizar a estrutura:

```
if (...) {

   return "...";
```

} else if (...) {

return "...";

} else {

return "...";

}

ETAPA 1 — PREVISÃO

Antes de programar, complete:

Média	Situação
8.0	
7.0	
6.5	
5.0	
4.9	
2.0	

ETAPA 2 — IMPLEMENTAÇÃO

Modifique:

verificarSituacao()

para implementar as três situações.

ETAPA 3 — TESTES

Teste pelo menos:

8

7

6

5

4

Observe principalmente:

🔎 Comparações / decisões

ETAPA 4 — PERSONALIZAÇÃO

No Main.java altere:

- **título da janela;**
- **textos dos campos;**
- **texto do botão;**
- **mensagem final.**

Crie sua própria versão da aplicação.

ETAPA 5 — APRESENTAÇÃO

O aluno deverá explicar:

1. **quais são as entradas;**
2. **quais variáveis foram utilizadas;**
3. **como a média é calculada;**
4. **quais métodos existem;**
5. **quais parâmetros são utilizados;**

6. **como funciona o `return`;**
7. **quais condições foram implementadas;**
8. **como Main.java e Processamento.java se comunicam.**

DESAFIOS EXTRAS

Após concluir o projeto principal, tente modificar os exemplos existentes.

Desafio 1 — Calculadora de desconto

Utilize dois valores:

Preço

Percentual de desconto

Fórmula:

desconto = preço × percentual / 100

Depois:

valorFinal = preço - desconto

Desafio 2 — Conversor de quilômetros

Receba:

quilômetros

e transforme em metros.

Sabendo que:

1 km = 1000 metros

Desafio 3 — Dobro de um número

Entrada:

Número

Processamento:

numero * 2

Saída:

Dobro = resultado

Desafio 4 — Verificador de nota

Receba uma nota.

Se:

nota >= 7

retorne:

Aprovado

Caso contrário:

Reprovado

Desafio 5 — Classificação de temperatura

Crie as seguintes regras:

temperatura < 20

FRIO

temperatura < 30

AGRADÁVEL

demais casos

QUENTE

Utilize:

if

else if

else

EXERCÍCIOS DE REVISÃO

1. O que é uma variável?

2. Qual a diferença básica entre `int`, `double` e `String`?

3. Para que serve `Double.parseDouble()`?

4. Qual a função do operador `%`?

5. Explique:

idade >= 18

6. Qual a diferença entre:

=

e:

==

7. Para que serve `if`?

8. Quando utilizamos `else`?

9. Quando podemos utilizar `else if`?

10. O que é um método?

11. O que são parâmetros?

12. Para que serve `return`?

13. Qual é a responsabilidade principal do Main.java nos exemplos estudados?

14. Qual é a responsabilidade do Processamento.java?

15. Explique o fluxo:

Entrada

↓

Main.java

↓

Processamento.java

↓

return

↓

Main.java

↓

Saída

CHECKLIST DE APRENDIZAGEM

Ao concluir este módulo, o aluno deverá conseguir:

☐ **compreender o conceito básico de programa;**

☐ **identificar entrada, processamento e saída;**

☐ **compreender a estrutura básica de um programa Java;**

☐ **declarar e utilizar variáveis;**

☐ **diferenciar `int`, `double`, `boolean` e `String`;**

☐ **utilizar operadores aritméticos;**

☐ **compreender operadores relacionais;**

☐ **utilizar estruturas `if`, `else if` e `else`;**

☐ **compreender métodos;**

☐ **compreender parâmetros;**

☐ **compreender o funcionamento de `return`;**

☐ **acompanhar valores pelo painel de variáveis;**

☐ **acompanhar decisões pelo painel de comparações;**

☐ **modificar cálculos existentes;**

☐ **modificar regras de negócio;**

☐ **modificar elementos simples da interface;**

☐ **interpretar a comunicação entre Main.java e Processamento.java;**

☐ **testar um programa utilizando diferentes entradas;**

☐ **identificar e corrigir erros simples;**

☐ **adaptar um exemplo existente para resolver um novo problema.**

CONCLUSÃO

Aprender programação não significa decorar comandos.

O objetivo principal é aprender a compreender como os dados percorrem um programa.

Durante este módulo trabalhamos constantemente com:

ENTRADA

↓

VARIÁVEIS

↓

PROCESSAMENTO

↓

DECISÃO

↓

RESULTADO

O simulador permite visualizar esse processo acontecendo.

Quando você modifica:

valor1 + valor2

para:

valor1 * valor2

não está apenas trocando um símbolo.

Está modificando a regra executada pelo programa.

Quando modifica:

media >= 7

para:

media >= 6

está modificando uma regra de decisão do sistema.

E quando cria um método:

public static double calcular(...)

está separando uma tarefa do sistema em um bloco reutilizável e organizado.

Essa capacidade de observar um problema, transformá-lo em regras e fazer o computador executar essas regras é um dos fundamentos da programação.

PRÓXIMOS PASSOS

Depois de dominar os conteúdos deste módulo, os próximos assuntos naturais são:

- **estruturas de repetição `for` e `while`;**
- **arrays;**
- **classes e objetos;**
- **atributos;**
- **construtores;**
- **encapsulamento;**
- **orientação a objetos;**
- **tratamento de exceções;**
- **coleções;**
- **arquivos;**
- **banco de dados;**
- **desenvolvimento de aplicações Java mais completas.**

O mais importante, porém, é dominar primeiro os fundamentos.

Leia o código. Faça uma previsão. Execute. Observe. Modifique. Execute novamente.

É assim que este módulo transforma o simulador em um verdadeiro laboratório de programação Java.