

Material de Apoio Aula 5

Controle de fluxo de execução: Seleção simples

Dentro de um método, o fluxo de execução das instruções é sequencial. Isto quer dizer que as instruções são executadas uma a uma, respeitando a ordem em que elas aparecem no código. Note que, a cada instrução executada, uma posição de memória pode ser alterada – esta memória pode ser representada por algum atributo no estado interno do objeto ou por alguma variável ou objeto local ao próprio método. Em um caso prático, a implementação de um método deve considerar diversas situações, as quais refletem em diferentes seqüências de execução de instruções, ou ainda, seleção de qual conjunto instruções deve ser executado, considerando cada uma das situações. O comando `if` (ou `if/else`) permite realizar uma forma de seleção, considerando uma condição única.

Sintaxe do comando `if` e `if/else`:

```
// Exemplo 1: caso a condição seja satisfeita, executa a Instrução_B antes de executar a
//           Instrução_D. Se a condição não for satisfeita, após a Instrução_A executa a
//           Instrução_D (neste caso, a Instrução_B não irá ser executada.

Instrução_A;
if ( condição )
    Instrução_B;
Instrução_D;

// Exemplo 2: caso a condição seja satisfeita, executa a Instrução_B antes de executar a
//           Instrução_D (a Instrução_C não será executada). Se a condição não for satisfeita,
//           após a Instrução_A é executada a Instrução_C e após a Instrução_D.

Instrução_A;
if ( condição )
    Instrução_B;
else
    Instrução_C;
Instrução_D;
```

Observe que nos dois exemplos acima, tanto a Instrução_B como a Instrução_C podem ser substituídas por um bloco de instruções. Neste caso, o bloco deverá ser delimitado por um par de abre e fecha chaves `{ }`.

```
// Exemplo 3: caso a condição seja satisfeita, executa as instruções do Bloco_B1, antes de
//           executar a Instrução_D. Se a condição não for satisfeita, após a Instrução_A
//           executa a Instrução_D (neste caso, nenhuma instrução do Bloco_B será executada.

Instrução_A;
if ( condição ) { // início do Bloco_B
    Instrução_B1;
    Instrução_B2;
    ...
} // fim do Bloco_B
Instrução_D;

// Exemplo 4 caso a condição seja satisfeita, executa as instruções do Bloco_B antes de executar
//           a Instrução_D (o Bloco_C não será executado). Se a condição não for satisfeita,
//           após a Instrução_A são executadas as instruções do Bloco_C e após a Instrução_D.

Instrução_A;
if ( condição ) { // início do Bloco_B
    Instrução_B1;
    Instrução_B2;
    ...
} // fim do Bloco_B
else{ // início do Bloco_C
    Instrução_B1;
    Instrução_B2;
    ...
} // fim do Bloco_C
Instrução_D;
```

Exercícios

1. Implemente uma classe `Primo`. Esta classe recebe no seu construtor um número `n` e deve implementar um único método: `ehPrimo`, o qual deve retornar `true` ou `false`, conforme o número for ou não primo.
2. Implemente a classe `MaiorDeTodos`. Esta classe possui um atributo inteiro e deve conter, pelo menos, os seguintes métodos:
 - a) Construtor sem parâmetro, que inicializa o estado interno com o menor número negativo (-2147483648);
 - b) Construtor com parâmetro, que inicializa o estado interno com o parâmetro informado;
 - c) Um método para registrar um novo valor;
 - d) Retornar o maior valor registrado.
3. Recupere a classe `Cofrinho`, desenvolvida em exercícios anteriores.
 - a) Reimplemente o código do método `sacar` para que não seja possível sacar um valor maior daquele que disponível em `saldo`. Experimente este código no BlueJ.
 - b) Repense o código acima de forma que o método retorne um valor booleano (`boolean`). O método deve retornar `true` no caso de sucesso na retirada, `false` no caso contrário.
 - c) Em linha de comando, implemente uma classe com um método `main` que utilize o método desenvolvido no Item 3.b). Deve ser impressa uma mensagem informando sucesso ou insucesso na operação de saque.
4. Reimplemente a classe `Funcionario` (Material Apoio 4, Exercício 5). Implemente para esta classe:
 - a) Um método para calcular o salário do final do mês. Como entrada, este método deve receber o número de horas trabalhadas no mês e não tem nenhum retorno. O resultado é a alteração de um atributo do objeto indicando quanto ele deve receber como pagamento.
 - b) Um método para calcular um bônus a ser pago para cada funcionário. A regra do bônus é atribuir um percentual sobre o salário para cada funcionário que receber até R\$ 320. Caso o funcionário deva receber mais do que R\$ 320, atribuir um valor (em reais) fixo. Este método deve receber dois parâmetros, o percentual do bônus e o valor fixo, para usar em quando o salário for maior que R\$ 320.
 - c) Use sobrecarga de método para implementar um novo método para bônus que aplique o mesmo bônus para todos os funcionários, independente de salário.
5. Reimplemente a classe para calcular o desempenho de uma turma (Material Apoio 4, Exercício 6), impedindo que sejam inseridas notas abaixo de 0 e acima de 10.
6. Reimplemente a classe para simular o comportamento de um carro (Material Apoio 4, Exercício 7). Nesta reimplementação, o método para fazer o deslocamento do veículo deve avaliar se a distância solicitada pode ser percorrida ou não. Caso possa, este método deve imprimir uma mensagem dizendo que o carro se deslocou e a quantidade de combustível consumida e quanto resta no tanque. Caso não seja possível o deslocamento, informar que o carro ficou parado, a quantidade de combustível que seria necessária ao deslocamento e a quantidade de combustível disponível no momento.
7. A média de um aluno, na Unisinos, é composta pelo somatório ponderado do Grau A com o Grau B ($GF = 0,33 * GA + 0,67 * GB$). O aluno é considerado aprovado caso obtenha como grau final uma nota igual ou superior a 6 ($GF \geq 6,0$). Caso não esteja aprovado, o aluno tem direito a realizar o Grau C. A nota do GC substitui uma das notas já obtida (GA ou GB), conforme escolha do aluno. Como o GA e o GB têm pesos diferentes, a escolha não é óbvia e necessita um cálculo para verificação. Implemente uma classe `Aluno`, que, além do nome e do número de matrícula do aluno, mantenha em seu estado interno informações sobre as notas dos diferentes graus. Esta classe deve prover um método para calcular a média (GF) e outro para indicar se o aluno deve fazer o GC para substituir o GA ou o GB. Esta classe deve também implementar métodos que informem:
 - O nome do aluno;
 - Se o aluno foi ou não aprovado na disciplina;
 - Se o aluno necessita fazer GC;
8. Implementar uma classe `Compara`. Esta classe não possui nenhum estado interno e apenas um método público. Este método recebe dois parâmetros, os dois objetos da classe `Aluno` (Exercício 7). O método deve informar o nome do aluno que possui maior GF. Caso as notas sejam iguais, isto deve ser informado.
9. Um investimento pré-fixado é um tipo de investimento em que o cliente aplica um certo valor, por uma taxa média mensal pré-fixada, por um certo número de meses. Ao final do período obtém um rendimento. O rendimento vai sendo capitalizado a cada mês. Crie uma classe de nome `Investimento`. Atributos: valor inicial

do investimento, taxa mensal de juros contratada, prazo (em meses) durante o qual o dinheiro permanece investido e o rendimento obtido durante esse tempo. Programe os seguintes métodos:

- Construtor para atribuir valores aos três primeiros atributos;
- Métodos para retornar o valor de cada um dos atributos;
- Método para calcular o rendimento a partir do que foi contratado.
- Método que retorne a quantidade de meses necessária para fazer dobrar o valor investido, se aplicado à mesma taxa contratada.

10. Implemente a classe Data, conforme especificação abaixo.

```
public class Data {
    private int dia, mes, ano;
    public Data(int d, int m, int a);
    public Data(int amd); // data no formato de um número inteiro AAAAMDD
                          // (ex: 20060322)
    public String obtemDataPadrao(); //devolve a data no formato padrão DD/MM/AAAA
                                    // (ex: 22/03/2006)
    public int obtemDataInvertida(); //devolve a data na forma de um número inteiro
}
```

11. Escreva uma classe Pessoa com dois atributos: nome e data de nascimento, sendo que este último é do tipo Data (pode ser declarado private Data dataDeNascimento;). Crie um construtor com quatro parâmetros para inicializar o nome e a data de nascimento. Crie os seguintes métodos:

- Método que devolve a idade da pessoa, calculada a partir da data de hoje, que deve ser fornecida como parâmetro;
- Método que devolve o nome.
- Implementar um novo método na classe Compara (Exercício 8) que receba dois objetos da classe Pessoa e informe quem é a mais velha (caso as duas tenham a mesma idade, dizer que os dois tem a mesma idade).

12. Crie uma classe de nome Atleta com os seguintes campos: nome, data de nascimento, peso e categoria.

- Crie um construtor que entre com valores para os três primeiros campos. A data de nascimento (conforme Exercício 10).
- Faça um método que retorne a idade do atleta a partir da data de nascimento.
- Escreva um método que determine a categoria do atleta de acordo com a seguinte tabela:

Idade	Peso	Categoria
12 anos	–	Infantil
13 a 16 anos	Até 40 Kg Acima de 40 Kg	Juvenil Leve Juvenil Pesado
17 a 24 anos	Até 45 Kg de 45,001 a 60 Kg acima de 60 Kg	Sênior leve Sênior médio Sênior pesado
Acima de 24 anos	–	Veterano

13. Uma loja de eletrodomésticos estabeleceu as seguintes modalidades de pagamento para as suas vendas:

```
À vista (1 vez)..... desconto de 2,5% sobre o preço de tabela;
De 2 até 5 vezes ..... preço de tabela, sem desconto ou acréscimo;
De 6 até 10 vezes ..... juros de 6% sobre o preço de tabela;
De 11 até 15 vezes ..... juros de 13% sobre o preço de tabela.
```

Exemplo:

preço de tabela = R\$ 100,00, para pagamento em 8 vezes;
preço total = $100,00 + 100 \cdot 0,06$ (6% de 100,00) = 106,00;
cada parcela = $106,00 / 8 = R\$ 13,25$.

- Escreva uma classe de nome Venda com os atributos preço de tabela, número de parcelas em que será paga a compra, preço total (após cálculo de descontos ou juros) e valor de cada parcela (preço total dividido pelo número de parcelas).
- Escreva um construtor que entra com o preço de tabela e o número de vezes em que será feito o pagamento.
- Escreva um método para calcular o preço total e o valor de cada parcela.