PROGRAMAÇÃO ORIENTADA A OBJETOS

Estruturas de Controle

Prof. Emanoel Barreiros

Algoritmos

- Qualquer problema computável é resolvido executando-se uma série de ações em uma ordem específica
- A isso damos o nome de algoritmo
- A maneira mais fácil de entender o conceito de algoritmo é pensar em uma receita de bolo
  - Adicione X xícaras de farinha de trigo
  - Adicione Y ovos
  - Adiciona W gramas de manteiga
  - …
  - Misture tudo, bata numa batedeira e coloca em uma forma
  - Etc.

Estruturas de controle

- Geralmente, o programa executa linearmente, linha após linha
- No entanto, existem alguns comandos que podem ser utilizados para controlar a execução do programa, fazendo com que a próxima linha a executar não seja necessariamente a próxima linha no código
- Na década de 60, muitas linguagens permitiam que o fluxo de controle do código passasse indiscriminadamente para qualquer ponto do código através de comandos *goto*
- Identificou-se no entanto que isso causava muita dificuldades entre os programadores

Estruturas de controle

- Na década de 1970 mostrou-se que os *goto* eram realmente péssimos
- Estudos mostraram que sem o uso do goto, mais projetos passaram a ser entregues no prazo, mais projetos passaram a ser executados dentro do orçamento

Controles de seleção

- Controles de seleção executam ações de acordo com o valor de uma expressão lógica (booleana)
  - Caso a expressão seja avaliada para **true**
- Existem 3 tipos de comandos de controle de seleção
  - if
  - if … else
  - switch

O *if*

- O comando **if** é chamado também de comando de seleção simples
- Um trecho de código só é executado caso a condição de entrada seja avaliada para **true**
- Caso a expressão seja avaliada para **false**, o trecho de código dentro do **if** é ignorado
- Ex:

```
if(nota >= 7.0) {
    System.out.println("Aluno aprovado.");
}
```

O *if … else*

- O comando **if…else** é chamado também de comando de seleção dupla
- Permite que um trecho de código seja selecionado se a expressão de entrade seja avaliada para **true** e um outro trecho seja selecionado caso contrário
- Caso a expressão seja avaliada para **false**, o trecho de código dentro do **if** é ignorado
- Ex:

```
if(nota >= 7.0) {
    System.out.println("Aluno aprovado.");
} else {
    System.out.println("Aluno reprovado.");
}
```

Comandos *if…else* "aninhados"

- Algoritmos mais complexos podem ser criados se você aninhar vários if ou if…else um dentro do outro

```
if (nota >= 9.0) {
    System.out.println("A");
} else {
    if(nota >= 8.0) {
        System.out.println("B");
    } else {
        if(nota >= 7.0) {
            System.out.println("C");
        } else {
            if(nota >= 6.0) {
                System.out.println("D");
            } else {
                System.out.println("F");
            }
        }
    }
}
```

Outra possibilidade para if…else aninhados

```
if (nota >= 9.0) {
    System.out.println("A");
} else if(nota >= 8.0) {
    System.out.println("B");
} else if(nota >= 7.0) {
    System.out.println("C");
} else if (nota >= 6.0) {
    System.out.println("D");
} else {
    System.out.println("F");
}
```

Comando de repetição while

- Um comando de repetição (ou *loop*) permite que um trecho de código seja executado repetidas vezes
- O trecho de código controlado pelo comando **while** executará enquanto sua condição for **true**
- Ex:

```
int produto = 1;
while (produto <= 100) {
    produto = produto * 3;
}
```

Exercício

- Usando a classe LivroNotas vamos implementar um algoritmo um pouco mais complexo:
  - Crie um método chamado **determinarMediaTurma**()
  - Utilizando o while e um contador, obtenha 10 notas do usuário e calcule a média da turma
  - Após calcular a média, exiba o resultado
  - Imprima a quantidade de alunos que tiraram abaixo de 7.0