

Programa o Estruturada

Aula 12 - Registros e enumera es

Apresentação

Nesta aula, você vai aprender sobre o uso de **registros e enumerações** na linguagem Java. Você aprenderá que informações manipuladas por um sistema, e que estejam relacionadas de alguma forma, podem ser agrupadas através de estruturas de dados chamadas de registros, mesmo que essas informações sejam de tipos de dados diferentes (*inteiros*, *float*, *Strings*, etc.).

Você também irá ter contato com as enumerações, que são novos tipos de dados que você pode definir e cujos valores correspondentes são constantes e passíveis de serem representados por nomes bem intuitivos. Como você verá nesta aula, tanto o uso de registros como o de enumerações irá facilitar o desenvolvimento de programas.

Sendo assim, faça uma boa leitura desse assunto!



Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você será capaz de:

- Conhecer conceitos relacionados aos registros e enumerações.
- Saber trabalhar com registros nos programas.
- Utilizar e aplicar enumerações nos programas.

1. Introdução

Para iniciar a aula de hoje, precisamos definir alguns conceitos. Você já sabe que um conjunto homogêneo de dados é composto por variáveis do mesmo tipo, como nos vetores. Mas, e se tivermos um conjunto em que os elementos não são do mesmo tipo? Por exemplo, os dados de um aluno (nome do tipo *String*, idade do tipo *int*, etc.). Temos, assim, um conjunto heterogêneo de dados, os quais não podem ser armazenados em um mesmo vetor, já que são valores com tipos de dados diferentes. Para esses casos, usamos um tipo de estrutura de dados chamada de **registro** (às vezes chamadas simplesmente de estrutura, dado o nome *struct*, usado na linguagem C, bastante conhecida no mercado).

Um registro é uma das principais formas de construir e organizar os dados no programa, visando facilitar o agrupamento de variáveis de tipos diferentes, mas que possuem uma relação lógica entre eles.

2. O que são Registros?

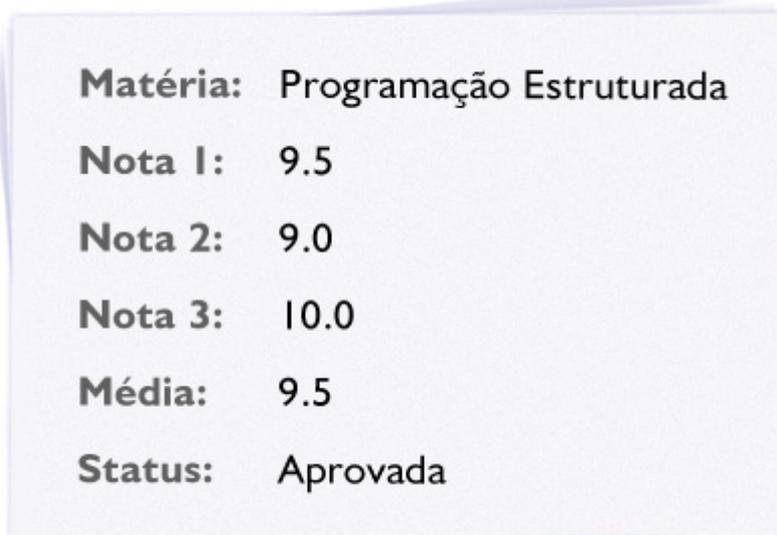
Registros são estruturas de dados definidas como um conjunto de uma ou mais variáveis relacionadas, que podem ser ou não de tipos diferentes, e que são agrupadas sobre um único nome. O fato de variáveis agrupadas em um registro poderem ser referenciadas por um único nome facilita a manipulação dos dados armazenados nessas estruturas.

Como exemplo, imagine uma estrutura de dados para armazenar as diversas notas que um aluno tirou em uma matéria. As notas junto com a informação sobre a matéria formam um conjunto de informações logicamente relacionadas, porém, de tipos diferentes, tais como:

- | | |
|---|--|
| 1 | Nome da matéria (tipo <i>String</i>), |
| 2 | Notas do aluno (sequência de números <i>float</i>), |
| 3 | Média (tipo <i>float</i>), |

Essas informações não podem ser agrupadas em um vetor, por serem de tipos de dados diferentes, mas podem ser agrupadas em um registro. A seguir, apresentamos na Figura 1, as informações em um registro criado para armazenar as informações de notas de um aluno.

Figura 01 - Informações de notas de um aluno



| | |
|-----------------|-------------------------|
| Matéria: | Programação Estruturada |
| Nota 1: | 9.5 |
| Nota 2: | 9.0 |
| Nota 3: | 10.0 |
| Média: | 9.5 |
| Status: | Aprovada |

As informações inseridas nesse registro são de tipos diferentes, mas todas elas estão relacionadas a um único aluno (o nome dele poderia estar no registro) e a uma única matéria. Esse agrupamento estabelece a função lógica de uma espécie de boletim do aluno naquela matéria. Isso irá facilitar, como veremos mais adiante, a manipulação das informações pelo programa.

Nas próximas seções, iremos aprender a manipular registros.

3. Declarando Registros

Conforme aprendemos em capítulos anteriores, para declarar uma variável é necessário informar o seu tipo e dar a ela um nome. Além disso, ela precisa ser definida em uma classe. Nas aulas anteriores você estava criando apenas classes que representavam os programas criados em nossas atividades de aula.

No caso dos registros, devemos colocá-los de forma separada do programa, em uma classe a parte. Para declarar um registro, precisamos declarar uma nova classe (representando o registro) e declarar dentro dela as variáveis que o registro irá agrupar. Crie um novo arquivo e coloque o registro nele, como já estamos acostumados a fazer. É importante notar se o registro e o programa estão no mesmo pacote, na aba Projects do NetBeans!

Vamos ver um exemplo. Imagine que precisamos criar uma estrutura de dados para armazenar o endereço de uma pessoa. Essa estrutura de dados, que contém tipos de dados diferentes, consegue ser bem representada por um registro, como mostrado no código a seguir:

```
1 public class Endereco {  
2     public String rua;  
3     public int numero;  
4     public String bairro;  
5     public String cidade;  
6     public String estado;  
7     public String cep;  
8 }
```

Note a criação da classe de nome “Endereco”. Essa classe tem como objetivo representar o registro, então seu nome deve representar o significado das informações armazenadas nele, ou seja, o endereço. As variáveis declaradas dentro do registro são justamente os componentes de um endereço (rua, número, bairro, cidade, estado e CEP).

Observe também que não estamos utilizando a palavra chave *static* na definição das variáveis do registro “Endereco”. Isto se deve ao fato de que uma variável declarada com o *static* é única e só representa um valor por vez. Quando não usamos o *static*, isso muda, mas esse assunto será visto um pouco mais adiante.

Por fim, o uso do *public* na definição dos campos do registro “Endereco” será posteriormente substituído por *private*. Mas isso você só vai aprender no módulo de Programação Orientada a Objetos, ao ser apresentado aos conceitos de tipos abstratos de dados e encapsulamento.

Como um segundo exemplo do uso de registros, vamos criar uma estrutura que contém os dados pessoais de uma pessoa:

```
1 public class Pessoa {
2     public String nome;
3     public String telefone;
4     public Endereco enderecoResidencia;
5 }
```

Agora nós temos uma novidade! De acordo com o exemplo anterior, podemos ver que o registro “Pessoa” está utilizando o outro registro “Endereco”. Sim! Isso é possível e correto. Já que uma pessoa possui um endereço de residência, nada mais intuitivo do que o registro “Pessoa” ter um registro “Endereco”.

E é por esse motivo que a estrutura nos dá uma grande flexibilidade de armazenamento e gerenciamento dos dados.

Atividade 01

1. Defina uma estrutura de dados para agrupar e representar os dados de um automóvel (marca do fabricante, modelo, ano de fabricação e quantidade máxima de passageiros).
2. Considerando que toda pessoa possua um carro, que alteração no registro "Pessoa" mostrado na aula nós precisamos fazer para representar esse relacionamento?

4. Utilizando Registros

Agora vamos mostrar como utilizar registros. Vamos criar um programa que lê os dados de um endereço. Considerando a classe “Registro” já criada, analise o seguinte código:

```

1 import java.util.Scanner;
2
3 public class ProgramaLeitorEndereco {
4     public static void main(String[] args) {
5         Scanner leitor = new Scanner(System.in);
6         Endereco end = new Endereco();
7         System.out.println("Digite a rua onde você mora:");
8         end.rua = leitor.nextLine();
9         System.out.println("Digite o número da residência:");
10        end.numero = leitor.nextInt();
11        leitor.nextLine();
12        System.out.println("Digite o bairro:");
13        end.bairro = leitor.nextLine();
14        System.out.println("Digite a cidade:");
15        end.cidade = leitor.nextLine();
16        System.out.println("Digite o estado:");
17        end.estado = leitor.nextLine();
18        System.out.println("Digite o CEP:");
19        end.cep = leitor.nextLine();
20        System.out.println("----- Seu endereço é -----");
21        System.out.println(end.rua + ", " + end.numero);
22        System.out.println(end.bairro + ", " + end.cidade);
23        System.out.println("CEP" + end.cep + ", " + end.estado);
24    }
25 }

```

Nesse programa, a novidade está no seguinte comando:

```

1 Endereco end = new Endereco();

```

Esse comando declara uma variável *end* do tipo “Endereco”. Note que variáveis podem ser não só de tipo primitivo (*int*, *float*, etc.), mas também do tipo registro. E a novidade está na forma que inicializamos esse registro. Usamos a palavra chave *new* para informar ao ambiente de execução do programa que é necessário alocar espaço na memória para caber todas as informações contidas em um registro do tipo “Endereco”.

Se as variáveis que armazenam as informações do endereço no registro (rua, número, bairro, cidade, etc.) tivessem sido declaradas com o *static*, não precisaríamos usar a palavra chave *new*. Usamos, nesse caso, justamente porque a alocação do espaço em memória é dinâmica, e não estática. A vantagem de alocar dinamicamente o espaço para os registros é que você pode alocar espaço para quantos registros quiser! Veremos isso através de um exemplo mais adiante.

Outra novidade é a forma de acessar as variáveis do registro. Basta colocar o nome da variável que representa o registro (*end*) e o nome do campo do registro que queremos acessar (rua, número, etc.) separado por um ponto. Dessa forma, para ler ou escrever o nome da rua, usamos a expressão "*end.rua*" para termos acesso de escrita ou leitura a esse espaço de memória.

Bom, agora que você já deve ter entendido bem o código do programa mostrado, vamos fazer uma pequena variação no código criando um novo programa:

```
1 import java.util.Scanner;
2
3 public class ProgramaLeitorEndereco {
4     public static void main(String[] args) {
5         Scanner leitor = new Scanner(System.in);
6         System.out.println("-- Digite seu primeiro endereço --");
7         Endereco end1 = lerEndereco(leitor);
8         System.out.println("-- Digite seu segundo endereço --");
9         Endereco end2 = lerEndereco(leitor);
10        System.out.println("----- Seu primeiro endereço é -----");
11        imprimirEndereco(end1);
12        System.out.println("----- Seu segundo endereço é -----");
13        imprimirEndereco(end2);
14    }
15
16    public static Endereco lerEndereco(Scanner leitor) {
17        Endereco end = new Endereco();
18        System.out.println("Digite a rua onde você mora:");
19        end.rua = leitor.nextLine();
20        System.out.println("Digite o número da residência:");
21        end.numero = leitor.nextInt();
22        leitor.nextLine();
23        System.out.println("Digite o bairro:");
24        end.bairro = leitor.nextLine();
25        System.out.println("Digite a cidade:");
26        end.cidade = leitor.nextLine();
27        System.out.println("Digite o estado:");
28        end.estado = leitor.nextLine();
29        System.out.println("Digite o CEP:");
30        end.cep = leitor.nextLine();
31        return end;
32    }
33    public static void imprimirEndereco(Endereco end) {
34        System.out.println(end.rua + ", " + end.numero);
35        System.out.println(end.bairro + ", " + end.cidade);
36        System.out.println("CEP " + end.cep + ", " + end.estado);
37    }
38 }
```

Observe que parte do código da rotina “main()” anteriormente mostrada foi deslocado para duas novas rotinas, o “lerEndereco” e o “imprimirEndereco”. A função “lerEndereco” lê um endereço do teclado e armazena os dados em um novo registro do tipo “Endereco”, retornando ao registro criado. Já o procedimento “imprimirEndereco” recebe como parâmetro um registro do tipo endereço e imprime seus dados na tela do computador.

A rotina “main()” criada, chama agora duas vezes as rotinas “lerEndereco” e “imprimirEndereco”. Isso quer dizer que serão criados, lidos e impressos na tela dois registros do tipo “Endereco”!

Note que registros podem ser passados como parâmetro para procedimentos e funções, bem como retornados por funções. Se o registro “Endereco” possui 6 campos (rua, número, bairro, etc.), isto quer dizer que evitamos de declarar 6 parâmetros na função “imprimirEndereco”, passando a apenas um, que é um registro que agrupa todos os 6 campos. Outra vantagem é que se surgir um sétimo campo em “Endereco”, automaticamente a função “imprimirEndereco” já estará apta a manipular esse novo campo, já que ela recebe a estrutura “Endereco” como um todo.



Vídeo 02 - Imprimir Dados dos Usuários

Os registros são recursos muito interessantes, não são?

Atividade 02

1. Escreva um programa que armazene os dados relacionados a um filme (nome, autor, ano, preço) em uma estrutura de dados do tipo registro. Crie rotinas para ler e para imprimir os dados desse tipo de registro. Crie uma rotina *main* para ler e imprimir os dados de 3 diferentes filmes.

5. Enumerações

Para entender a motivação das enumerações, vamos primeiro imaginar que precisamos declarar uma variável para representar o dia da semana no qual ocorreu a venda de um produto. Qual seria o tipo de dados que você utilizaria para declarar essa variável? Uma opção seria declará-la como do tipo *int*, usando os números de 1 a 7 para representar os dias da semana (domingo, segunda-feira, ..., sexta-feira). Outra opção seria usar caracteres ou *Strings*.

Pensando na variável declarada como do tipo *int*, temos os seguintes problemas:

- Programadores podem se confundir na manutenção do programa para saber se o valor 2 é segunda-feira ou terça-feira, por exemplo. Afinal, a semana começa no domingo ou na segunda-feira? Isso é uma decisão de quem programou originalmente o programa.
- Um valor inválido, como 8, pode ser atribuído a essa variável. Isto porque sendo do tipo *int*, a variável pode receber qualquer valor do tipo inteiro suportado pelo computador.

Para contornar esses problemas, existe um recurso interessante que algumas linguagens fornecem: as enumerações. De forma geral, uma enumeração é um tipo de dados cujos valores são determinados por um conjunto fixo de constantes. Por exemplo, podemos declarar a seguinte enumeração em Java para representar os dias da semana:

```
1 public enum EnumDiaSemana {  
2     DOMINGO, SEGUNDA_FEIRA, TERCA_FEIRA,  
3     QUARTA_FEIRA, QUINTA_FEIRA, SEXTA_FEIRA, SABADO  
4 }
```

Assim como as classes, enumerações em Java precisam ser criadas em arquivos separados. Observe o uso de *public enum* ao invés de *public class* na primeira linha da declaração da enumeração. Muitos ambientes de desenvolvimento Java possuem uma opção de menu para criar uma nova enumeração, assim como acontece com a criação de classes.

Ao invés de conterem variáveis e rotinas, as enumerações em Java contêm uma lista de constantes que representam os possíveis valores da enumeração. No caso do exemplo mostrado, existe uma constante para cada dia da semana. Note que as constantes não são declaradas como valores 1, 2, 3, etc., mas sim como nomes intuitivos sobre o que cada uma das constantes representa.

Uma vez declaradas as constantes da enumeração, fazemos uso delas da seguinte forma:

```
1 import java.util.Scanner;
2
3 public class ProgramaEnum {
4     public static void main(String[] args) {
5         Scanner leitor = new Scanner(System.in);
6         System.out.println("Digite o dia da semana (1 a 7, onde 1 representa o domingo:");
7         EnumDiaSemana diaSemana = pegarDiaSemana(leitor.nextInt());
8         // aqui o programa faz o que ele precisa fazer
9         // ...
10        System.out.println("Você selecionou o dia da semana de "+ nomeDiaSemana(diaSemana));
11    }
12
13    public static EnumDiaSemana pegarDiaSemana(int diaSemanaInt) {
14        EnumDiaSemana diaSemana = EnumDiaSemana.SABADO;
15        switch (diaSemanaInt) {
16            case 1:
17                diaSemana = EnumDiaSemana.DOMINGO;
18                break;
19            case 2:
20                diaSemana = EnumDiaSemana.SEGUNDA_FEIRA;
21                break;
22            case 3:
23                diaSemana = EnumDiaSemana.TERCA_FEIRA;
24                break;
25            case 4:
26                diaSemana = EnumDiaSemana.QUARTA_FEIRA;
27                break;
28            case 5:
29                diaSemana = EnumDiaSemana.QUINTA_FEIRA;
30                break;
31            case 6:
32                diaSemana = EnumDiaSemana.SEXTA_FEIRA;
33        }
34        return diaSemana;
35    }
36
37    public static String nomeDiaSemana(EnumDiaSemana diaSemana) {
38        String nome = "Sábado";
39        switch (diaSemana) {
```

```

40     case DOMINGO:
41         nome = "Domingo";
42         break;
43     case SEGUNDA_FEIRA:
44         nome = "Segunda-feira";
45         break;
46     case TERCA_FEIRA:
47         nome = "Terça-feira";
48         break;
49     case QUARTA_FEIRA:
50         nome = "Quarta-feira";
51         break;
52     case QUINTA_FEIRA:
53         nome = "Quinta-feira";
54         break;
55     case SEXTA_FEIRA:
56         nome = "Sexta-feira";
57     }
58     return nome;
59 }
60 }

```

Note a declaração da variável “diaSemana” na rotina “main()” como do tipo “EnumDiaSemana”. Para lermos um valor para essa variável do teclado, temos que usar algum artifício como o mostrado pela função “pegarDiaSemana()”. Do teclado, nós lemos um número e depois convertemos seu valor para uma das opções da enumeração, como mostrado dentro do código da função “pegarDiaSemana()”.

Com a variável inicializada, podemos criar programas para manipular esse valor. Por fim, quando precisarmos imprimir essa informação na tela, faremos novamente uma conversão. Isto é mostrado pela função “nomeDiaSemana()”. Essa função recebe um valor do tipo “EnumDiaSemana” e converte para uma *String* que melhor representa o valor recebido. Na rotina “main()”, essa *String* é então utilizada para apresentar a informação para o usuário.



Vídeo 03 - Imprimir Dados Usuários (Enumeração)



Vídeo 04 - Dia Semana

Atividade 03

1. Considere que um programa precisa manipular os meses (janeiro a dezembro). Defina uma enumeração para representar esses valores.
2. Considerando a atividade anterior, crie agora funções para: ler valores dessa enumeração (meses do ano) a partir do teclado; escrever esses valores na tela.

6. Resumo

Nesta aula, você aprendeu a utilizar estruturas de dados chamadas de registros e as enumerações. Você viu que os registros são um agrupamento de variáveis que podem ser de tipos diferentes, mas que tem uma relação lógica entre elas. Por serem formados por elementos possivelmente de diferentes tipos, os registros são chamados de estruturas de dados heterogêneas e cada elemento agrupado pelo registro é chamado de campo. Você aprendeu também a utilizar as enumerações nos programas para definir tipos de dados que representem valores específicos, como os dias da semana, meses do ano, situações de alunos em um curso (matriculado, trancado, aprovado, reprovado, etc.). Com todo o conhecimento adquirido nesta aula, você está dando os primeiros passos para elaborar programas de maior complexidade! Na próxima aula, veremos outro assunto importante para a construção de programas mais complexos: a manipulação de arquivos. Até lá!

7. Autoavaliação

1. Explique o que são estruturas de dados do tipo registro, indicando em quais situações elas são úteis.
2. Indique quais são os possíveis benefícios de utilizarmos registros.
3. Explique o que são enumerações, indicando em quais situações elas são úteis na criação de programas de computador.
4. Indique quais são os possíveis benefícios de utilizarmos enumerações.

8. Referências

ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. **Fundamentos da programação de computadores:** algoritmos, Pascal, C/C++ e Java. São Paulo: Editora Pearson, 2008.

THE JAVA tutorials. Disponível em:
<http://download.oracle.com/javase/tutorial/>. Acesso em: 6 dez. 2011.