Noções sobre Objetos e Classes

Prof. Marcelo Cohen

1. Elementos de programação

- Revisão de programação
  - variáveis, tipos de dados
  - expressões e operadores
  - cadeias de caracteres
  - escopo de variáveis
- Revisão do conceito classe-objeto
  - os objetos em Java são criados a partir de **classes**
  - então, para se usar um objeto em um programa, é necessário primeiramente instanciá-lo. Como fazer isso ???

- Tipos de dados
  - inteiros: ***byte, short, int, long***
  - reais: ***float, double***
  - caractere: ***char***
  - booleano: ***boolean*** (assume valores ***true*** ou ***false***)
- Variáveis
  - mesma sintaxe usada em C:
    - ***int valor;***
    - ***double taxa, percentual;***
    - ***int idade = 25;***
    - ***char letra = 'X';***
  - convenção para nomes de variáveis em Java
    - ***int valorAtual;***
    - ***double taxaCrescAnual;***
- Comentários
  - mesma sintaxe usada em C/C++
    - *// isto é um comentário de uma linha*
    - */* Isto é um comentário de múltipas linhas ... */*

- Expressões e operadores
  - aritméticos:
    - `+, -`
    - `*, /, %`
  - relacionais:
    - `>, >=`
    - `<, <=`
    - `==, !=`
  - lógicos
    - `&, && - and`
    - `|, || - or`
    - `! - not`
    - `^ - xor`
  - relacionais:
    - `>, >=`
    - `<, <=`
    - `==, !=`
  - funções matemáticas
    - classe **Math**
    - `double raiz = Math.sqrt(25);`

Ativação do método **sqrt(double)**, na classe Math

- Constantes
  - `final double PI = 3.141592;`
  - `final int idade = 28;`

- Cadeias de caracteres: strings são implementadas através de uma classe
  - `String nome;`
  - `nome = “Fulano”;`
- atribuição simultânea
  - `String nome = “Fulano”;`
  - `String sobrenome = “de Tal”;`
- aritmética de strings:
  - `String nomeCompleto = nome + “ “ + sobrenome;`
- Se uma expressão tiver uma *String*, Java converte tudo para strings:
  - `int idade = 25;`
  - `String nomeIdade = nome + “tem “ + idade + “anos”;`
- também funciona o operador +=
  - `String nomeCompleto+=“ Jr.”;`

- Escopo de variáveis
  - As variáveis (e os métodos também) podem possuir um dos três seguintes atributos de escopo:
    - **private - visível apenas na classe atual**
    - **public - visível para qualquer classe externa**
    - *protected - a ser comentado posteriormente...*

O escopo de variáveis é uma das formas de garantir o encapsulamento correto!

2. Revisão do Conceito Classe-Objeto

- Objetos em Java são criados a partir de classes
- Lembre-se que as classes servem para “fabricar” objetos
- Então, para se usar um objeto em um programa, é necessário primeiramente instanciá-lo. Como fazer isso ???

Exemplo: classe Pessoa

- Criaremos uma classe para representar alguns dados sobre pesssoas:
- A classe deve representar o nome, idade e sexo de alguém
- A classe deve possibilitar a exibição desses dados na tela

Classe Pessoa
Dados: nome, idade e sexo
Operações: imprimir nome, imprimir idade, imprimir sexo

3. Olhando a classe de perto

```
class Pessoa
{
   private String nome;
   private int idade;
   private char sexo;

...

} // fim da classe
```

- Observe a utilização do modificador private, pois as variáveis só devem ser acessadas **dentro da classe**!

```
...
   public void imprimeNome()  {
     System.out.println(nome);
   }

   public void imprimeIdade() {
     System.out.println(idade);
   }

   public void imprimeSexo() {
     System.out.println(sexo);
   }
} // fim da classe
```

- O que é *`System.out.println(...)`* ???

- É a chamada para o método `println(...)`, que exibe uma string na tela
- Observe que funciona também para qualquer tipo de dado primitivo (int, char, float, double, boolean)

- Atenção: a utilização desse método pode criar um programa que não seja 100% portável, pois a exibição de caracteres pode variar dependendo do tipo de terminal!

- Pergunta: quando instanciarmos objetos a partir desta classe, como informaremos os valores que estes deverão armazenar ?
- Solução: quando um objeto em Java é criado, um método especial é chamado no momento exato da criação, denominado **construtor**.
- Esse método possui o mesmo nome da classe, e não tem tipo de retorno:

```
class Pessoa
{  ...
   public Pessoa(String n,int i,char s) {
      nome = n;
      idade = i;
      sexo = s;
   } // fim do construtor
```

4. Instanciando objetos

- Para instanciar um objeto em java, utiliza-se o operador **new**
- Exemplo: criação de dois objetos a partir da classe Pessoa:

```
Pessoa pessoa1, pessoa2;
pessoa1 = new Pessoa(“Fulano”,25,’M’);
pessoa2 = new Pessoa(“Ciclana”,18,’F’);
```

...

5. Como fazer um programa completo ???

- Uma vez que os programas em Java são compostos por diversas instâncias e classes, como saber onde inica a execução ?
- Definiu-se um método especial, denominado **main**, que indica para o compilador Java onde o programa inicia, e deve ter a seguinte forma

```
public static void main(String args[])
{
  Pessoa pessoa1;
  pessoa1 = new Pessoa(“Fulano”,25,’M’);
   ...
}
```

- Observe a utilização do construtor criado anteriormente!

- A partir dos objetos já criados, é possível chamar qualquer método que seja public:

```
public static void main(String args[])
{
  Pessoa pessoa1,pessoa2;
  pessoa1 = new Pessoa("Fulano",25,'M');
  pessoa2 = new Pessoa("Ciclana",18,'F');

  pessoa1.imprimeNome();
  pessoa1.imprimeIdade();
  pessoa2.imprimeNome();
  pessoa2.imprimeSexo();
  ...
}
```

5. Exercícios

- Escreva uma classe "Contador", que apresente métodos para informar o valor inicial, incrementar, decrementar e exibir o valor atual na tela.

- Altere a classe, para que funcione como um relógio, armazenando e contando horas, minutos e segundos (cada incremento/decremento) corresponde a um segundo.

- Exemplifique a utilização desta classe, criando uma nova classe que crie duas ou mais instâncias da primeira e use os métodos definidos. Exemplo: crie três relógios e chame alguns métodos para incrementar/decrementar o tempo. No final, exiba o horário de cada um.