4ª Aula Prática de PLC: Conceitos de Orientação a Objetos em Java

Roteiro

- Classes e Objetos
- Métodos e Atributos
- Construtores
- Sobrecarga (Overloading) de Métodos
- Troca de Mensagens

Classes e Objetos

Uma classe representa um conjunto de objetos com as mesmas características. Através da definição de uma classe, descreve-se que atributos (propriedades) que o objeto terá. Uma classe é definida pelo seu nome, pelo seu conjunto de atributos e de métodos.

Objetos são instâncias de classes. No paradigma de orientação a objetos, tudo pode ser potencialmente representado como um objeto. Um objeto é identificado pela sua referência.

Classes e Objetos

Exemplo de uma classe:

```
class Pessoa {
        String nome;
        int idade;
        char sexo;
        float altura, peso;
}
```

Exemplos de objetos:

```
Pessoa gabriel = new Pessoa( );
Pessoa paula = new Pessoa( );
Pessoa pedro = new Pessoa( );
```

Classes e Objetos

Exemplo de uma classe:

```
class Livraria {
        String[] livros;
        double[] precos;
        double lucro;
        boolean procurarLivro(String livro);
}
```

Exemplos de objetos:

```
Livraria saraiva = new Livraria( );
Livraria cultura = new Livraria( );
Livraria imperatriz = new Livraria( );
```

Métodos e Atributos

Um método nada mais é que o equivalente a um procedimento ou função (também podem ter um retorno), com a restrição que ele pode manipular apenas suas variáveis locais e os atributos que foram definidos para a classe.

Os atributos, por sua vez, são características de um objeto. Basicamente a estrutura de dados que vai representar a classe.

Métodos e Atributos

Exemplo de um método:

```
boolean procurarLivro(String livro){
	boolean encontrou=false;
	for(int i=0; i<livros.length && !encontrou; i++){
		if(livros[i].equals(livro)) encontrou=true;
	}
	return encontrou;
}
```

Exemplos de atributos:

```
String[] livros;
double lucro;
```

Construtores

O construtor da classe é um bloco declarado com o mesmo nome que a classe e é semelhante a uma rotina de inicialização que é chamada sempre que um novo objeto é criado.

O construtor não é um método, pois só é chamado uma vez para cada objeto e não retorna nada. Entretanto, o construtor pode receber argumentos e pode haver mais de um construtor por classe (desde que cada um receba parâmetros diferentes).

O construtor dá a possibilidade/obriga o usuário a passar argumentos para o objeto durante o processo de criação do mesmo.

Construtores

Exemplo de uma classe com um construtor:

```
class TV {
        int canal, volume;
        boolean ligada;
        TV(int canal, int volume){
                this.canal=canal;
                this.volume=volume;
                ligada=true;
        }
}
```

Exercícios Resolvidos

1. Crie uma classe Data com um construtor que recebe um dia, um mês e um ano. Faça, também, as validações necessárias (desconsidere anos bissextos) e crie métodos getters e setters para os atributos dessa classe.

Data(short dia, short mes, short ano);

Escreva, agora, um método vemAntes na classe Data que receba como argumento outra instância da mesma classe e retorne true se a data vier antes da passada como argumento e false caso contrário.

boolean vemAntes(Data data);

Sobrecarga (Overloading) de Métodos

Dois ou mais métodos/construtores de uma classe podem ter o mesmo nome, desde que suas assinaturas sejam diferentes. O nome desse mecanismo é sobrecarga.

Tal situação não gera conflito pois o compilador é capaz de detectar qual método deve ser escolhido a partir da análise dos tipos dos argumentos do método.

Sobrecarga (Overloading) de Métodos

Por exemplo:

```
class TV {
        int canal, volume;
        boolean ligada;
        void ligarTV( );
        void ligarTV(boolean ligada);
        void ligarTV(int canal);
        TV(int canal, int volume){
                this.canal=canal;
                this.volume=volume;
                ligada=true;
        }
}
```

Sobrecarga (Overloading) de Métodos

Onde:

```
void ligarTV( ){
        canal=3;
        volume=30;
        ligada=true;
}
void ligarTV(boolean ligada){
        canal=3;
        volume=30;
        this.ligada=ligada;
}

void ligarTV(int canal){
        this.canal=canal;
        volume=30;
        ligada=true;
}
```

Troca de Mensagens

Um programa orientado a objetos é composto por um conjunto de objetos que interagem através de trocas de mensagens. Na prática, essa troca de mensagem traduz-se na aplicação de métodos a objetos, ou seja, é simplesmente uma chamada a um objeto para invocar um de seus métodos.

```
class Main {
	static void main(String[] args){
		TV minhaTV = new TV(13, 10);
		minhaTV.ligarTV(9);
	}
}
```

Exercícios Resolvidos

2. Escreva uma classe Serie que represente a geração de séries numéricas como as usadas em testes de raciocínio, onde uma pessoa deve deduzir a regra que gerou os números e acertar os próximos números da série.

A série deve ser gerada usando três valores: inicial, multiplicador e adicional, de forma que o primeiro número da série será igual a inicial, o segundo será calculado como
(inicial + adicional) * multiplicador, o terceiro como
(segundo + adicional) * multiplicador, e assim sucessivamente.

Os valores devem ser passados como argumentos para o construtor da classe e usados por um método imprimir, que recebe como argumento o número de termos que serão impressos.

CONTINUA ->

Exercícios Resolvidos

Por exemplo, uma aplicação poderia criar duas instâncias da classe Serie e imprimir, respectivamente, os primeiros 10 e 15 termos, com o trecho de código abaixo:

```
Serie s1 = new Serie(0, -2, 2);
s1.imprimir(10);
Serie s2 = new Serie(1, 2, 0);
s2.imprimir(15);
```

O resultado da execução do trecho de código acima é mostrado abaixo.

```
0 -4 4 -12 20 -44 84 -172 340 -684
1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384
```

Exercício Prático

1. Implemente a classe Texto que é a representação de um texto qualquer com algumas operações básicas.
A forma de implementar essa classe ficará a sua escolha (desde que você respeite os princípios da programação orientada a objetos, principalmente os de encapsulamento), entretanto a classe deverá possuir métodos para imprimir o texto completo na tela do console, informar o número total de palavras do texto, informar o número de uma determinada palavra dentro do texto e substituir todas as ocorrências de uma palavra no texto por uma outra palavra qualquer.
Crie, também, uma aplicação simples que utilize ao menos uma vez cada um dos métodos propostos para essa classe.

Exercício Prático

O exercício prático deve ser realizados individualmente e enviado para **monitoria-if686-ec-l@cin.ufpe.br** com o assunto "**[IF686EC] EXERCÍCIOS PRÁTICOS 04**" até as 23:59 de hoje (16.05.2017).