Sistemas de Informação

Desenvolvimento de Aplicativos I

Classes e Objetos – Parte I

Prof. Marciel de Liz Santos

Classes e Objetos
Classes

- Uma classe é um tipo definido pelo usuário que possui especificações (características e comportamentos) que o identifiquem.
- De uma maneira mais objetiva, podemos dizer que a classe é um molde que será usado para construir objetos que representam elementos da vida real.
- Tal molde é gerado através da observação e agrupamento de elementos que possuam as mesmas características e comportamentos sob uma mesma denominação.

Classes e Objetos
Classes

- Exemplificando, em uma loja online, identificamos um elemento chamado Produto (classe), que possui um conjunto de características tal como a identificação e o Preço (atributos).

Produto	
Identificação	Preço

Classes e Objetos
Classes

- Cada Produto poderia ser materializado com as informações abaixo:

Produto	
Identificação	Preço
Celular	R$ 200,00

Produto	
Identificação	Preço
Camiseta	R$ 15,00

Classes e Objetos
Classes

- No entanto, os **Atributos** nem sempre são úteis individualmente. Convém definirmos algumas ações e comportamentos que esse elemento possa executar dentro do ambiente modelado usando os atributos existentes na Classe.
- Esses comportamentos são denominados **Métodos** e definem as ações previstas para essa classe. Ainda no exemplo da loja, poderíamos atribuir aos Produtos algumas funcionalidades tais como aumentar/diminuir o preço e alterar a identificação.

Classes e Objetos
Classes

- Note que normalmente as ações estão associadas aos atributos da classe (Preço, Identificação).

Produto	
Identificação	Preço
Aumentar preço Diminuir preço Alterar identificação	

Classes e Objetos
Objetos

- Note que nos exemplos citados (Celular, R$ 200,00; Camiseta, R$ 15,00) representamos casos particulares da classe Produto. Eles são denominados instâncias das classes, ou objetos, e eles têm vida independente entre si, apesar de compartilharem o mesmo "molde" (Classe).
- A criação de objetos é feita através da utilização do operador *new* e então o espaço necessário de memória para o objeto é alocado. O Garbage Collector libera o espaço de memória alocado para o objeto quando ele não está mais sendo utilizado.

Classes e Objetos
Objetos

Sintaxe:

<NomeDaClasse> <nomeVariavel> = new <NomeDaClasse>();

- Para criarmos uma instância da classe Produto utilizamos o seguinte comando:

  Produto prod = new Produto();

- Para criarmos uma instância da classe Pessoa utilizamos o seguinte comando:

  Pessoa pes = new Pessoa();

Classes e Objetos
Abstração

- Como uma classe pode representar qualquer coisa em qualquer ambiente, é sugerido que a modelagem foque nos objetivos principais do negócio. Isso evita que o sistema seja muito grande e consequentemente de difícil manutenção e compreenção. A análise focada é denominada Abstração.
- No mundo real, podemos utilizar um produto que possui informações de garantia, matéria-prima, etc. Porém, conforme o tipo de aplicativo, pode ser ou não interessante colocarmos tais informações na definição do objeto.

Classes e Objetos
Abstração

- No mundo real podemos trocar produtos entre pessoas, entretanto, num cenário de e-business, não precisamos implementar este comportamento ao definirmos o Produto.

Exemplo:abstração de Produto

Qual é as características básicas (estrutura) de um produto?

- id
- preço
- Nome

Qual o comportamento desta entidade no mundo real?

- Aumentar o preço
- Aplicar desconto
- Alterar o nome

Classes e Objetos
Abstração

Exemplo:abstração de Data

Qual é as características básicas (estrutura) de um produto?

- dia
- mês
- ano

Qual o comportamento desta entidade no mundo real?

- Transformar o número do mês em um texto com o nome do mês (1:Janeiro, 2:Fevereiro)
- Transformar o número do dia em nome do dia (24:Segunda, 25:Terça)
- Devemos saber se o ano é bissexto

Classes e Objetos
Atributos

- As definições das características de um objeto são feitas de atributos da classe.
- Atributos armazenam dados do objeto.
- Declaração de atributos
  - Atributos são declarados dentro da declaração da classe, ou seja, dentro do bloco delimitado pela abertura e fechamento das chaves ({ e}).
  - Não podemos declarar atributos dentro de métodos, pois variáveis declaradas dentro de métodos são chamadas variáveis locais e tem escopo menot do que um atributo.
  - Quando declaramos um atributo em uma classe indicamos o tipo de dado que será armazenado nele.

Classes e Objetos
Atributos

- Declaração de atributos
  - Atributos podem ser de dois tipos:
    - Tipos primitivos (long, int, boolean, etc)
    - Tipos reference (Arrays ou String, Integer, Date, Cliente, Produto, Pessoa, etc)

Sintaxe para declaração:

<tipo do atributo> identificador;

Exemplo: Produto.java

```
class Produto{
    int id;
    String descricao;
    double preco;
}
```

Classes e Objetos
Atributos

- Convenção para nomes de atributos
  - Identificadores de atributos devem ser declarados com letras minúsculas
  - Quando o nome do atributo for composto por duas ou mais palavras, a separação deve ser feita com um caracter maiúsculo.

  Exemplo: telefoneComercial, enderecoDeEntrega, contratoPessoaJuridica.
- Acessando atributos a partir de outras classes
  - A partir de uma instância de classe, podemos acessar seus atributos com dois objetivos, ler o valor do atributo ou alterar o valor do atributo.
  - Para tal, utilizamos ponto e mais o nome do atributo

  Exemplo: nomeClasse.nomeAtributo

Classes e Objetos

Exemplo do uso da classe Produto

- Primeiro criamos a classe Produto.java

```
class Produto {
        int id;
        String descricao;
        double preco;
}
```

Classes e Objetos

- Depois criamos a classe TesteProduto.java

```
class TesteProduto {
    public static void main(String[] args) {
        //Criamos a instância da classe Produto
        Produto prod = new Produto();
        //Acessamos os atributos da classe Produto para definir valores
        prod.id = 11;
        prod.descricao = "Celular";
        prod.preco = 200;
        //Acessando os atributos para leitura
        System.out.println("Id do produto: " + prod.id);
        System.out.println("Descricao do produto: " + prod.descricao);
        System.out.println("Preço do produto: " + prod.preco);
    }
}
```

Classes e Objetos
Métodos

- Através dos métodos definimos as operações que podem ser executadas com ou sobre um objeto. Popularmente diz-se que os métodos definem o comportamento da classe.
- Declaração de métodos
  - Métodos são declarados dentro do corpo da classe, ou seja, dentro do bloco de código da definição da classe, que é definido com abre e fecha chaves.

  Um método é dividido em três partes:
  - retorno do método
  - nome do método
  - parâmetros do método

Classes e Objetos
Métodos

- Sintaxe para declaração de métodos:

  <tipo do retorno> nomeDoMetodo (<parametrosDeMetodos>){ }

- Sintaxe para declaração de <parametrosDeMetodos>:

  (<tipo> identificador, <tipo> identificador...)

- Convenções para nomes de métodos
  - Identificadores de métodos devem ser declarados com letras minúsculas
  - Quando o nome do método for composto por duas ou mais palavras, a separação deve ser feita com um caracter maiúsculo
  - Uma sugestão seria o uso de verbos no infinitivo e na voz passiva

Classes e Objetos
Métodos

- Retorno de métodos
  - Para retornar o valor para quem chamou o método utilizamos a instrução *return*
  - O tipo do valor retornado deve ser compatível com o daquele indicado na assinatura do método
  - O valor de retorno de um método pode ser uma literal
  - O retorno de um método pode ser obtido através da chamada a um método ou expressão
  - Variáveis que recebem valores retornados nos métodos devem obrigatoriamente ser de tipo compatível com o declarado na assinatura do método
  - Os valores retornados por métodos não precisam ser armazenados em variáveis
  - Quando um método não retorna valor nenhum, indicamos em sua declaração que seu tipo de retorno é *void*

Classes e Objetos
Métodos

Exemplo: ExemploMetodos.java

```
class ExemploMetodos {
    //void indica que o metodo nao retorna nada
    void imprime(){
        System.out.println("Este metodo nao retorna nada!");
    }
    //o metodo DEVE retornar um int ou tipo primritivo compativel, caso contrario
    //haveria um erro de compilacao
    int calculaFrete(){
        return 19;
    }
    //o metodo DEVE retornar um objeto da classe String, ou seja, um tipo reference
    String getNome(){
        return "Este metodo retorna uma String!";
    }
}
```

Classes e Objetos
Métodos

- Passagem de parâmetros
  - Os parâmetros de um método devem ser declarados separadamente usando-se vírgula entre os parênteses da assinatura do método
  - Em chamadas a métodos, é obrigatório passar os parâmetros usando os mesmos <tipos> definidos em sua assinatura, caso contrário, o compilador acusará um erro
  - Variáveis são SEMPRE passadas através de uma cópia de seu valor, diferente de outras linguagens nas quais a própria variável é passada

Classes e Objetos
Métodos

Exemplo de passagem de parâmetros: Calculadora.java

```
class Calculadora {
    //a passagem de dois valores do tipo int (ou tipo compativel) e obrigatoria
    int soma (int x, int y){
        return x + y;
    }
    //a passagem de dois valores do tipo double (ou tipo compativel) e orbigatoria
    double multiplicacao(double d1, double d2){
        double resultado = d1 * d2;
        return resultado;
    }
```

Continua...

Classes e Objetos
Métodos

Exemplo de passagem de parâmetros: Calculadora.java

Continuação...

```
    //a passagem de dois valores do tipo int e orbigatoria
    boolean maior(int num1, int num2){
        if (num1 > num2){
            return true;
        }else{
            return false;
        }
    }
    //a passagem de um parametro do tipo String e obrigatoria
    void print(String texto){
        System.out.println("Texto: " + texto);
    }
}
```

Classes e Objetos
Métodos

- Acessando métodos a partir de outras classes
  - Para acessar os métodos de uma classe a partir de outra classe, é necessário criar o objeto. A partir desta instância, podemos acessá-los, usando a seguinte sintaxe:

nomeDoObjeto.nomeDoMetodo(<argumentos>);

  - Convém observar que a passagem de argumentos é opcional e depende da assinatura do método.

Classes e Objetos
Métodos

Exemplo acessando métodos de outra classe:
TesteCalculadora.java

```
class TesteCalculadora {
    public static void main(String[] args) {
        Calculadora calc = new Calculadora();
        calc.print("Vamos testar a calculadora");
        int resultado1 = calc.soma(10, 10);
        System.out.println("10 + 10 " + resultado1);
        double resultado2 = calc.multiplicacao(10, 10);
        System.out.println("10 * 10 " + resultado2);
        boolean resultado3 = calc.maior(20, 100);
        System.out.println("20 > 100 " + resultado3);
    }
}
```