Conceitos de Orientação a Objetos

Programação Orientada a Objetos - Aula 05

Professor: Hamilton Machiti da Costa

fonte: Barnes & Kölling, Programação orientada a objetos com Java - Uma introdução prática usando o BlueJ, 4a edição, ed. Prentice Hall

Introdução

- Até o momento aprendemos as estruturas básicas de programação em Java: atribuição de variáveis, operadores, desvio condicional, laços de repetição, etc.

- Porém estamos desenvolvendo tudo dentro do método **main**. Programar desta maneira nos permite lidar apenas com pequenos problemas, como os que temos resolvido até agora.

- Mas, para criar software de verdade, precisamos introduzir mais um conceito: o de **modularização**.

- Modularizar é dividir o problema em partes menores, mais fáceis de lidar, e depois construir o código para resolver cada uma destas partes.

- Conforme vamos codificando cada uma destas partes, vamos então juntando cada uma delas, de maneira bem organizada, até termos o problema todo codificado e nosso software criado.

- Mas como vamos escolher em que partes quebrar? Aí entra uma outra técnica importante, que é a **abstração**.

- Abstrair é olhar para o problema e incluir na solução somente as partes que precisam estar lá, ignorando o resto.

- Imagine um sistema controlar vagas de estacionamento: para cadastramos um veículo, precisamos saber se é carro ou moto, a marca, o modelo, a cor e a placa. Deixamos de lado uma série de informações como potência, se o motor é flex, o ano de fabricação, se o câmbio é automático ou manual, etc.

- Por que deixamos todas estas informações de lado? Porque não são importantes para a solução do problema que temos, que é saber qual veículo irá usar cada vaga.

- Juntando estes conceitos de **modularização** e **abstração**, iremos a partir de agora ensinar a vocês um jeito de programar bastante poderoso e flexível, chamado de Orientação a Objetos.

- Portanto, a partir de agora, usaremos o **main** somente para iniciar nossos programas, colocando o primeiro objeto para funcionar. O restante do código iremos sempre escrever em outra(s) classe(s) separadas e métodos separados.

Objetos

- **Paradigma**: palavra bonita que quer dizer modelo.
- **Paradigma orientado a objetos**: conjunto de técnicas que procura representar partes do mundo real em programas de computador.
- Os “representantes do mundo real” são os **objetos** e as “partes do mundo real” são chamadas **domínio**.

Objetos e Domínio

- Considere um sistema acadêmico. O domínio do problema é o ambiente da universidade, e os objetos representados no sistema serão alunos, professores, disciplinas, cursos, avaliações, etc.

- Considere agora um sistema de vendas online. Os objetos modelados serão os clientes, os produtos, o carrinho de compras, o cartão de crédito, o estoque, etc.

Classes e Objetos

- **Classes** são templates para a criação de objetos. Assim, as classes descrevem o tipo de objeto.

- Cada classe de um sistema descreve um único tipo de objeto.

- Os objetos são **instâncias** individuais das classes. Em um sistema pode haver vários objetos do mesmo tipo. Isto é, uma única classe dá origem a vários objetos do mesmo tipo.

- Veja o objeto como uma materialização da classe. A classe é uma definição no papel. O objeto é a classe em ação no sistema.

```java
public class Aluno
{

}
```

```java
public class Disciplina
{

}
```

```java
public class Turma
{

}
```

Veja acima três exemplos de classe em Java. Para criar uma classe em Java basta usar a palavra reservada **class**, colocar o nome da classe (Turma, Disciplina, Aluno) e abrir e fechar chave. Use sempre letras maiúsculas para iniciar o nome da classe. Se tiver dois nomes, use uma maiúscula para cada nome, como DisciplinaEspecial. A palavra **public** antes de class não é obrigatória, mas geralmente é usada. Observe a ordem em que as palavras são escritas. É sempre public class NomeDaClasse { } . Nunca mude isso.

Métodos e Atributos

- Os objetos são definidos, nas classes, em termos de **métodos** e **atributos**.
- Os métodos definem o que o objeto faz.
- Os objetos se comunicam um com os outros chamando seus métodos.
- Os atributos definem características dos objetos ou outros objetos com os quais um objeto interage.
- Podem ser vistos também como locais onde os objetos armazenam dados para utilizar.
- Os atributos são também chamados de variáveis de instância.

```java
public class Aluno
{
    private String nome;
    private int idade;
    private double peso;
    private boolean formando;
    private char sexo;
}
```

```java
public class Disciplina
{
    private String nome;
    private int qtdAlunosMatriculados;

    public void nomear(String n){
        nome = n;
    }
    public int qtdAlunos(){
        return qtdAlunosMatriculados;
    }
}
```

```java
public class Turma
{
    public void matricula(Aluno aluno, Disciplina disciplina){
    }
}
```

Uma classe pode ter só atributos, como Aluno, só métodos, como Turma, ou atributos e métodos, como a classe Disciplina. Os nomes de métodos e de atributos começam sempre com minúsculas, como idade em Aluno e inscrever em Disciplina. Se houver dois ou mais nomes, o primeiro começa com minúscula e os demais com maiúscula, como em qtdAlunosMatriculados. Vamos ver tudo por partes. Primeiro, os tipos de dados.

Tipos de Dados

- Como já vimos, os tipos de dados básicos do Java são
  - int, tipo inteiro: 10, 20, -10
  - double, tipo real: 10.0, 20.98, -10.3849 (note que o separador de decimais é o ponto)
  - String, tipo texto: “João da Silva”, “amarelo” (note que os textos estão sempre entre aspas duplas e que o tipo String começa com letra maiúscula; se escrever com minúscula não é String)
  - boolean, tipo lógico: vale true ou false (verdadeiro ou falso, sem aspas)
  - char, tipo caractere: vale um caractere, ‘a’, ‘A’, ‘M’ (sempre entre aspas simples)

Tipos Primitivos e Objetos

- Os tipos básicos, menos a String, são chamados de **tipos primitivos**. Tem este nome pois é a partir deles que os tipos Objetos são criados.

- Os tipos primitivos não tem métodos.

- Qualquer **objeto** pode ser um tipo de dado. Portanto podemos criar atributos do tipo Aluno, do tipo Disciplina e de qualquer outro objeto que definirmos.

- A String é um tipo Objeto com tratamento especial em Java. Mais sobre isso mais tarde.

```java
public class Aluno
{
    private String nome;
    private int idade;
    private double peso;
    private boolean formando;
    private char sexo;
}
```

```java
public class Disciplina
{
    private String nome;
    private int qtdAlunosMatriculados;

    public void nomear(String n){
        nome = n;
    }
    public int qtdAlunos(){
        return qtdAlunosMatriculados;
    }
}
```

```java
public class Turma
{
    public void matricula(Aluno aluno, Disciplina disciplina){
    }
}
```

A classe Aluno tem atributos de todos os tipos básicos (primitivos + String). Veja que, na definição de um atributo, o tipo sempre vem antes do nome do atributo e depois da palavra private. Na classe Disciplina, o método nomear recebe um parâmetro do tipo String e o método qtdAlunos tem um retorno do tipo int. E na classe Turma, o método matricula recebe dois parâmetros objeto, dos tipos Aluno e Disciplina. Mas o que são parâmetros e retornos?

Parâmetros

- São valores passados para os métodos que fornecem informações adicionais a eles e alteram seu comportamento. Um método pode receber zero, um, dois ou mais **parâmetros**.

- Os parâmetros vem sempre entre parênteses. Mesmo quando não há nenhum parâmetro é preciso colocar os parênteses. São os parênteses que indicam que aquilo é um método e não um atributo.

- A declaração de parâmetros é sempre assim:

  tipo nomeDoParametro, tipo nomeDoOutroParametro

Retornos

- Os métodos quase sempre retornam algum valor
- Os tipos destes **retornos** são todos os tipos do Java, primitivos ou objetos
- Se o método não for retornar nada usamos a palavra **void**
- O tipo de retorno vem sempre antes do nome do método e depois da palavra public

```java
public class Aluno
{
    private String nome;
    private int idade;
    private double peso;
    private boolean formando;
    private char sexo;
}
```

```java
public class Disciplina
{
    private String nome;
    private int qtdAlunosMatriculados;

    public void nomear(String n){
        nome = n;
    }
    public int qtdAlunos(){
        return qtdAlunosMatriculados;
    }
}
```

```java
public class Turma
{
    public void matricula(Aluno aluno, Disciplina disciplina){
    }
}
```

O método nomear da classe Disciplina recebe o parâmetro String n e não retorna nada. Por isso é void. O método qtdAlunos não recebe nada como parâmetro e retorna int. Veja que para retornar usamos a palavra return e que o método está retornando o atributo qtdAlunosMatriculados, que é int. Por isso retorna int. Veja que um método pode retornar double, boolean, Aluno, Disciplina, isto é, qualquer tipo válido em Java.

Modificadores de Acesso

- O Java tem 4 tipos public, private, protected e um default, usado quando não se escreve nada.
- No momento vamos nos preocupar com apenas dois deles:
  - **public**: dá acesso para todos os outros objetos; use apenas para métodos, construtores e a própria classe; nunca use em atributos
  - **private**: dá acesso apenas para a própria classe; use sempre nos atributos e algumas vezes em métodos que você queira que somente o próprio objeto use

Construtores

- Os **construtores** são métodos especiais usados para instanciar uma classe, “construindo” um novo objeto
- Tem 2 características que o distinguem dos outros métodos
  - Tem exatamente o mesmo nome da classe
  - Não tem retorno
- Não é obrigatório criar um construtor para uma classe
- Quando criamos devemos usá-lo para atribuir valores iniciais para os atributos da classe, isto é, configurar o **estado** inicial do objeto

Estado

- O estado de um objeto é representado pelos valores dos seus atributos em um determinado momento
- A mudança de um atributo muda o estado
- Classes não tem estado, só objetos, pois somente os objetos podem receber valores

```java
public class Aluno
{
    private String nome;
    private int idade;
    private double peso;
    private boolean formando;
    private char sexo;

    public Aluno(String n, int i, double p, char s){
        nome = n;
        idade = i;
        peso = p;
        sexo = s;
        formando = false;
    }
}
```

Exemplo de construtor da classe Aluno. Note que não há o tipo de retorno entre o public e o nome do método. Somente o construtor pode ser assim, nenhum outro método. Veja que o construtor é usado para inicializar as variáveis com os parâmetros recebidos ou com valores padrão, com o caso do atributo formando. O construtor sempre é public.

Encapsulamento

- É um conceito muito importante em orientação a objetos.
- Ao tornar os atributos private, com acesso somente pela próprio objeto, você os encapsula dentro do objeto, evitando que outros objetos alterem o seu valor sem que você saiba.
- Para controlar o acesso aos atributos, o objeto de fornecer métodos para isso, chamados de métodos de **acesso** e métodos **modificadores**.

Métodos de Acesso

- São métodos que retornam o valor de um determinado atributo da classe
- Tem as seguintes características
  - sempre se chamam get + nome do atributo retornado com a primeira letra maiúscula
  - não recebem parâmetro
  - seu tipo de retorno é o mesmo do atributo retornado
  - terminam com um return atributo

```java
public class Aluno
{
    private String nome;
    private int idade;
    private double peso;
    private boolean formando;
    private char sexo;

    public Aluno(String n, int i, double p, char s){
        nome = n;
        idade = i;
        peso = p;
        sexo = s;
        formando = false;
    }

    public String getNome(){
        return nome;
    }
    public int getIdade(){
        return idade;
    }
    public double getPeso(){
        return peso;
    }
    public boolean getFormando(){
        return formando;
    }
    public char getSexo(){
        return sexo;
    }
}
```

O exemplo acima ilustra todos os métodos de acesso da classe Aluno. Veja que todos são public, enquanto os atributos são private. Note que o atributo nome é retornado pelo método getNome e, como nome é String, o tipo de retorno do método é String. A idade é retornada por getIdade, que é int. O formando é retornado por getFormando, que é boolean. E assim por diante.

Métodos Modificadores

- São métodos que alteram o valor de um determinado atributo da classe, mudando seu estado
- Tem as seguintes características
  - sempre se chamam set + nome do atributo alterado com a primeira letra maiúscula
  - sempre recebem parâmetro do mesmo tipo que o atributo alterado
  - seu tipo de retorno é sempre void
  - fazem atributo = parâmetro

```java
public class Aluno
{
    private String nome;
    private int idade;
    private double peso;
    private boolean formando;
    private char sexo;

    public Aluno(String n, int i, double p, char s){
        nome = n;
        idade = i;
        peso = p;
        sexo = s;
        formando = false;
    }
    //modificadores
    public void setNome(String n){
        nome = n;
    }
```

```java
public void setIdade(int i){
    idade = i;
}
public void setPeso(double p){
    peso = p;
}
public void setFormando(boolean f){
    formando = f;
}
public void setSexo(char s){
    sexo = s;
}
```

Estes são os métodos modificadora da classe Aluno. Veja que, assim como os de acesso, todos são public, enquanto os atributos são private. Note que o atributo nome é alterado pelo método setNome e, como nome é String, o parâmetro s recebido pelo método é String. A idade é mudada por setIdade, que é int. Por isso o parâmetro é int. O formando é mudado por setFormando, que é boolean. Então o parâmetro é boolean. E assim por diante.

Instanciação de Objetos

- Para se criar uma nova instância de uma classe chama-se o construtor desta classe com a palavra new

Aluno aluno = new Aluno(“João da Silva”, 19, 72.5, ‘M’);

```java
import javax.swing.JOptionPane;

public class TesteAluno {

    // cadastrar um novo aluno no metodo main
    public static void main(String[] args) {

        // coletando os dados do aluno a ser cadastrado

        String nome = JOptionPane.showInputDialog("Nome");
        int idade = Integer.parseInt(JOptionPane.showInputDialog("Idade"));
        double peso = Double.parseDouble(JOptionPane.showInputDialog("Peso"));
        // pega o primeiro caractere da String e retorna como char
        char sexo = JOptionPane.showInputDialog("Sexo M/F").charAt(0);

        // cria um objeto aluno
        Aluno aluno = new Aluno(nome, idade, peso, sexo);

        // monta a String de saida chamando os metodos de acesso do aluno
        String msg = "Nome: " + aluno.getNome() + "\nIdade: "
        + aluno.getIdade() + " anos" + "\nPeso: " + aluno.getPeso()
        + " kg";
        if(aluno.getFormando()){
            msg += "\nFormando: sim";
        } else {
            msg += "\nFormando: nao";
        }
        if(aluno.getSexo() == 'M'){
            msg += "\nsexo: masculino";
        } else {
            msg += "\nsexo: feminino";
        }
        // mostra o aluno
        JOptionPane.showMessageDialog(null, msg);
    }
}
```

Resumo de Conceitos

- **Abstração**: capacidade de ignorar detalhes de partes para focalizar a atenção em um nível mais elevado de um problema.
- **Modularização**: processo de dividir um todo em partes bem definidas, que podem ser construídas e examinadas separadamente e que interagem de maneiras bem definidas.
- **Orientação a Objetos**: forma de modelar o mundo real em um sistema.
- **Objeto**: elemento do mundo real representado em um sistema.
- **Classe**: definição dos objetos em termos de atributos e métodos.

- **Atributo**: expressa características de um objeto.
- **Método**: define as ações de um objeto.
- **Chamada de método**: os objetos se comunicam chamando métodos uns dos outros.
- **Assinatura**: é o cabeçalho de um método que fornece informações sobre como chamá-lo.
- **Retorno**: os métodos retornam alguma tipo de dado ou void se não retornam nada.

- **Tipo de dado**: indica o conjunto de valores que um atributo ou parâmetro podem assumir ou que um método pode retornar.

- **Tipo primitivo**: são tipos básicos que não tem métodos. Ex: int, double, boolean, char.

- **Tipo objeto**: são tipos complexos que tem métodos, criados a partir dos tipos primitivos e de outros tipos objeto.

- **Estado**: conjunto de valores dos atributos de um objeto em um determinado momento.

- **Método Modificador**: altera o valor de um atributo, mudando o estado do objeto.
- **Método de Acesso**: retorna o valor de um atributo.
- **Construtor**: instancia um objeto, geralmente atribuindo valores para seus atributos.
- **Parâmetro**: valores passados para os métodos como informações adicionais