ECM251 – Linguagens de Programação I

Aula 06 – L1/1 e L2/1

Engenharia da Computação – 3ª série

Conceitos de Orientação a Objetos – O.O.
(L1/1 – L2/1)

2024

Horário

Terça-feira: 2 x 2 aulas/semana

- L1/1 (07h40min-09h20min): *Prof. Calvetti*;
- L1/2 (09h30min-11h10min): *Prof. Calvetti*;
- L2/1 (07h40min-09h20min): *Prof. Igor Silveira*;
- L2/2 (11h20min-13h00min): *Prof. Calvetti*.

Tópico

- Introdução à Orientação a Objetos – O.O.

Introdução à O.O.

- Até o momento foram estudadas as estruturas básicas de programação em Java: atribuição de variáveis, operadores, desvio condicional, laços de repetição, etc.;
- Porém, se desenvolveu tudo dentro do método ***main( )***;
- Programar desta maneira permite lidar apenas com pequenos problemas, como os que foram resolvidos até agora;
- Mas, para criar *software* de verdade, precisa-se introduzir mais um conceito: o conceito da **modularização**;
- Modularizar é dividir o problema em partes menores, mais fáceis de lidar e, depois, construir o código para resolver cada uma destas partes.
- ;

Introdução à O.O.

- Conforme codifica-se cada uma destas partes, junta-se, então, cada uma delas, de maneira bem organizada, até se ter o problema todo codificado e o *software* criado;
- Mas como escolher em que partes quebrar o problema?
- 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, no momento;

Introdução à O.O.

- Num sistema para controlar vagas de estacionamento, deve-se cadastrar um veículo levando-se em consideração, somente, se o mesmo é um carro ou moto, sua marca, modelo, cor e placa;
- Deixa-se de lado uma série de informações, por exemplo, potência do motor, se o motor é *flex*, o ano de fabricação, se o câmbio é automático ou manual etc.

Introdução à O.O.

- Por que se deixam todas estas informações de lado?
- Porque não são importantes para a solução do problema que se tem, que é saber qual veículo irá usar cada uma das vagas;
- Juntando estes conceitos de modularização e abstração, a partir de agora, será introduzida uma forma de programação bastante poderosa e flexível, chamada de Orientação a Objetos;
- Portanto, a partir de agora, o método ***main( )*** será utilizado, apenas, para iniciar os programas, colocando o primeiro objeto para funcionar;
- O restante do código será sempre escrito em outra classe separada, uma ou muitas, com seus métodos especificados.

Definições

- Paradigma: palavra bonita que quer dizer modelo, padrão;
- Paradigma Orientado a Objetos: conjunto de técnicas que procura representar partes do mundo real em programas de computadores;
- Os “representantes do mundo real” são chamados de **objetos** e as “partes do mundo real” são chamadas de **domínio**.

Objetos e Domínios

- Considere um sistema acadêmico: o **domínio** do problema será o ambiente da universidade e os **objetos** representados no sistema serão os 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. e o **domínio** será o ambiente da loja.

Classes e Objetos

- **Classes** são *templates* para a criação de **objetos**, portanto, as **classes** descrevem os tipos de **objetos**;
- 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;
- O **objeto** pode ser entendido como uma materialização da **classe**;
- A **classe** é uma definição no "papel" e o objeto é a **classe** em ação no sistema.

Classes e Objetos

```
public class Aluno
{

}
```

```
public class Disciplina
{

}
```

```
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, suas ações;
- Os **objetos** se comunicam um com os outros chamando, ou invocando, seus **métodos**;
- Os **atributos** definem características dos seus **objetos** ou outros **objetos**, com os quais um **objeto** interage;
- Os **atributos** podem ser vistos, também, como locais onde os **objetos** armazenam seus dados para posterior utilização;
- Os **atributos** são também chamados de **variáveis de instância**.

Métodos e Atributos

```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 visto, os tipos de dados básicos do Java são:
  - ***int***, tipo inteiro: 10, 20, -10;
  - ***double***, tipo real: 10.0, 20.98, -10.3849 (o separador de decimais é o ponto);
  - ***String***, tipo texto: “João da Silva”, “amarelo” (os textos estão sempre entre aspas e o tipo ***String*** começando com letra maiúscula. Se escrever com minúscula não é ***String***;
  - ***boolean***, tipo lógico: vale ***true*** ou ***false***, sem aspas;
  - ***char***, tipo caractere: vale um caractere, ‘a’, ‘A’, ‘M’, sempre entre apóstrofes;

Tipos Primitivos e Objetos

- Os tipos básicos, menos a ***String***, são chamados de tipos **primitivos**;
- Eles têm 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, por exemplo, pode-se criar atributos do tipo Aluno, do tipo Disciplina e de qualquer outro objeto que se defina;
- ***String*** é um tipo **Objeto** com tratamento especial em Java.

Tipos Primitivos e Objetos

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

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

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

```
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** vêm sempre entre parênteses;
- Mesmo quando não há nenhum **parâmetro** é preciso colocar os parênteses;
- São os parênteses que indicam que se trata de um **método** e não de um **atributo**;
- A declaração de **parâmetros** segue esta regra:

tipo *nomeDoParametro,* tipo *nomeDoOutroParametro*

Retornos

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

Retornos

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

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

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

```
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 de modificadores de acesso:

  public, ***private***, ***protected*** e um ***default***, usado quando não se escreve nada.

- No momento, serão introduzidos os dois principais tipos:
  - ***public***: dá acesso para todos os outros objetos e procura-se utilizá-lo para métodos, construtores e a própria classe, evitando-se utilizá-los em atributos;
  - ***private***: dá acesso apenas para a própria classe e procura-se utilizá-lo para os atributos e em métodos que se queira acessá-los somente o próprio objeto onde está definido.

Construtores

- Os **construtores** são métodos especiais, usados para instanciar uma **classe**, "construindo" um novo objeto;
- Eles têm 2 (duas) características que os distinguem dos outros métodos:
  - Têm exatamente os mesmos nomes das suas classes;
  - Não têm retornos;
- Não é obrigatório criar um **construtor** para uma classe;
- Quando se cria um **construtor**, deve-se utilizá-lo para atribuir valores iniciais nos atributos de sua classe, isto é, configurar o **estado inicial** desse objeto.

Estado

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

Estado

```
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 pelo próprio **objeto**, os **atributos** são **encapsulados** dentro do **objeto**, evitando que outros **objetos** alterem o valor do atributo sem que haja controle;
- Para controlar o acesso aos **atributos**, o **objeto** deve fornecer os **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 sua **classe**;
- Os **métodos de acesso** têm as seguintes características:
  - Sempre recebem o nome de ***get*** + nome do **atributo** retornado, sempre com a primeira letra maiúscula;
  - Não recebem parâmetro;
  - Seu tipo de retorno é o mesmo tipo do **atributo** retornado;
  - Terminam com ***return*** do atributo.

Métodos de Acesso

```
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 o seu estado;
- Eles têm as seguintes características:
  - Sempre recebem o nome de ***set*** + nome do **atributo** alterado, sempre com a primeira letra maiúscula;
  - Sempre recebem um **parâmetro** do mesmo tipo que o **atributo** a ser alterado;
  - Seu tipo de retorno é sempre ***void***;
  - Internamente, fazem **atributo** = **parâmetro**.

Métodos Modificadores

```
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;
    }
```

```
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** , deve-se chamar o **construtor** dessa **classe**, utilizando a palavra ***new***;

Aluno aluno = new Aluno("João da Silva", 19, 72.5, 'M');

Conceitos de O.O.

Exemplo

```
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 dos Conceitos Importantes

- 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;

Resumo dos Conceitos Importantes

- Atributo: expressa características de um objeto;
- Método: define as ações de um objeto;
- Chamada de método: os objetos se comunicam chamando os 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;

Resumo dos Conceitos Importantes

- 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 têm métodos, por exemplo, *int*, *double*, *boolean*, *char*;
- Tipo objeto: são tipos complexos que têm 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;

Resumo dos Conceitos Importantes

- 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.

Exercícios

- Extras, propostos pelo professor em aula, utilizando os conceitos abordados neste material...

Bibliografia Básica

- MILETTO, Evandro M.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP (Tekne). Porto Alegre: Bookman, 2014. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788582601969

- WINDER, Russel; GRAHAM, Roberts. Desenvolvendo Software em Java, 3ª edição. Rio de Janeiro: LTC, 2009. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/978-85-216-1994-9

- DEITEL, Paul; DEITEL, Harvey. Java: how to program early objects. Hoboken, N. J: Pearson, c2018. 1234 p. ISBN 9780134743356.

Continua…

Bibliografia Básica (*continuação*)

- HORSTMANN, Cay S; CORNELL, Gary. Core Java. SCHAFRANSKI, Carlos (Trad.), FURMANKIEWICZ, Edson (Trad.). 8. ed. São Paulo: Pearson, 2010. v. 1. 383 p. ISBN 9788576053576.
- LIANG, Y. Daniel. Introduction to Java: programming and data structures comprehensive version. 11. ed. New York: Pearson, c2015. 1210 p. ISBN 9780134670942.
- TURINI, Rodrigo. Desbravando Java e orientação a objetos: um guia para o iniciante da linguagem. São Paulo: Casa do Código, [2017]. 222 p. (Caelum).

Bibliografia Complementar

- HORSTMANN, Cay. Conceitos de Computação com Java. Porto Alegre: Bookman, 2009. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788577804078

- MACHADO, Rodrigo P.; FRANCO, Márcia H. I.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software III: programação de sistemas web orientada a objetos em java (Tekne). Porto Alegre: Bookman, 2016. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788582603710

- BARRY, Paul. Use a cabeça! Python. Rio de Janeiro: Alta Books, 2012. 458 p.
  ISBN 9788576087434.

Continua...

Bibliografia Complementar (continuação)

- LECHETA, Ricardo R. Web Services RESTful: aprenda a criar Web Services RESTfulem Java na nuvem do Google. São Paulo: Novatec, c2015. 431 p.
  ISBN 9788575224540.

- SILVA, Maurício Samy. JQuery: a biblioteca do programador. 3. ed. rev. e ampl. São Paulo: Novatec, 2014. 544 p.
  ISBN 9788575223871.

- SUMMERFIELD, Mark. Programação em Python 3: uma introdução completa à linguagem Phython. Rio de Janeiro: Alta Books, 2012. 506 p.
  ISBN 9788576083849.

Continua...

Bibliografia Complementar (continuação)

- YING, Bai. Practical database programming with Java. New Jersey: John Wiley & Sons, c2011. 918 p.
- ZAKAS, Nicholas C. The principles of object-oriented JavaScript. San Francisco, CA: No Starch Press, c2014. 97 p. ISBN 9781593275402.

FIM

Engenharia da Computação – 3ª série

Laços de Repetição
(L1/2 – L2/2)

2024

Horário

Terça-feira: 2 x 2 aulas/semana

- L1/1 (07h40min-09h20min): *Prof. Calvetti*;
- L1/2 (09h30min-11h10min): *Prof. Calvetti*;
- L2/1 (07h40min-09h20min): *Prof. Igor Silveira*;
- L2/2 (11h20min-13h00min): *Prof. Calvetti*.

Exemplo

Crie as classes conforme a descrição abaixo:

a. Crie a classe **Aluno** com os atributos privados: **nome**, do tipo *String*, **idade**, do tipo *int*, **peso**, do tipo *double*, **formando**, do tipo *boolean* e **sexo**, do tipo *char*;

b. Crie o construtor da classe **Aluno** que recebe parâmetros para inicializar os atributos **nome**, **idade**, **peso** e **sexo**. O atributo **formando** deve ser inicializado automaticamente com ***falso***;

c. Crie os métodos de acesso para todos os atributos;

d. Crie os métodos modificadores para todos os atributos;

Exemplo

e. Crie a classe **TesteAluno** com um método ***main***;

f. No método ***main***, use o método ***JOptionPane.showInputDialog***, para ler os valores de **nome**, **idade**, **peso** e **sexo**;

g. Depois, ainda no ***main***, instancie um objeto **Aluno** e passe como parâmetros do construtor os valores lidos;

h. Depois, ainda no ***main***, imprima todos os atributos do objeto **Aluno** usando ***JOptionPane.showMessageDialog*** e os métodos de acesso de **Aluno**;

Exemplo

i. Depois, ainda no ***main***, pergunte novamente ao usuário os valores da **idade**, do **peso** e se o aluno é **formando**;

j. Altere estes atributos com os parâmetros informados por meio dos métodos modificadores;

k. Finalmente, ainda no ***main***, imprima novamente os valores dos atributos, como fez no item *h* desta lista.

Conceitos de O.O.

Exemplo

Solução: Classe Aluno

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

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

    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;
    }
}
```

Exemplo

Solução: Classe TesteAluno

```
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";
        }
```

Conceitos de O.O.

Exemplo

```
if (aluno.getSexo() == 'M') {
    msg += "\nsexo: masculino";
} else {
    msg += "\nsexo: feminino";
}
// mostra o aluno
JOptionPane.showMessageDialog(null, msg);

// altera informacoes; nao precisa criar todas as variaveis novamente
idade = Integer.parseInt(JOptionPane.showInputDialog("Idade"));
peso = Double.parseDouble(JOptionPane.showInputDialog("Peso"));
// tem que digitar true ou false
boolean formando = Boolean.parseBoolean(JOptionPane
        .showInputDialog("E' formando?true/false"));

// muda usando os metodo modificadores
aluno.setIdade(idade);
aluno.setPeso(peso);
aluno.setFormando(formando);

// mostra novamente o cadastro do aluno
// monta a String de saida chamando os metodos de acesso do aluno
msg = "Nome: " + aluno.getNome() + "\nIdade: " + aluno.getIdade()
        + " anos" + "\nPeso: " + aluno.getPeso() + " kg";
```

Conceitos de O.O.

Exemplo

```
        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);
    }
}
```

Exercícios

1. Crie a classe Turma com seu construtor, métodos de acesso e modificadores e os atributos privados nome, do tipo String, curso, do tipo String, quantidadeDeAlunos, do tipo int, serie, do tipo int;
2. Crie a classe TesteTurma com o método main. De modo análogo ao exemplo, peça para o usuário entrar com os valores necessários para criar uma turma, instancie um objeto Turma e depois exiba os dados da turma criada. Depois, peça para o usuário uma nova quantidade de alunos, altere o valor do atributo e exiba os dados novamente;

Exercícios

3. Crie a classe Produto com seu construtor, métodos de acesso e modificadores e os atributos privados nome, do tipo String, preço, do tipo double, quantidade, do tipo int. Crie uma classe TesteProduto com um método main, onde você vai ler valores de um produto, instanciar um produto e exibir o produto criado;
4. Crie a classe Disciplina com seu construtor, métodos de acesso e modificadores e os atributos privados nome, do tipo String, professor, do tipo String, semestre, do tipo int, ofertada, do tipo boolean. Crie a classe TesteDisciplina com o método main, leia dos dados necessários para criar uma disciplina, crie uma disciplina e exiba a disciplina criada;

Atividade

- Individualmente, resolver os exercícios propostos e apresentar à sala, explicando-a, na próxima aula L1/2 e L2/2, a solução daquele solicitado pelo professor.

Bibliografia (apoio)

- LOPES, ANITA. GARCIA, GUTO. Introdução à Programação: 500 algoritmos resolvidos. Rio de Janeiro: Elsevier, 2002.
- DEITEL, P. DEITEL, H. Java: como programar. 8 Ed. São Paulo: Prentice-Hall (Pearson), 2010.

FIM