Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Programação Orientada a Objetos

Semana 06
Encapsulamento e Serialização

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Reflexão

"Não devemos acreditar na maioria que diz que apenas as pessoas livres podem ser educadas, mas sim acreditar nos filósofos que dizem que só as pessoas educadas são livres."

Epictetus (55dc-133dc)

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Encapsulamento

- Separar em partes
- Isolar
- Proteger
- Restringir acesso e conteúdo

Encapsular,
Encapsulado,
Encapsulamento,

Incluído ou encerrado em uma cápsula.
Isolamento de um corpo, substância, energia, dados...
De um determinado ambiente ou entre ambientes.

Revestimento, proteção, escudo, bloqueio... De um determinado evento, isolando-o do meio circunvizinho.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Encapsulamento

Um mecanismo que possibilita restringir o acesso a variáveis e métodos da classe (ou até a própria classe).

Permite ocultação de informações (visibilidade);

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Visibilidade

- Define quem enxerga atributos e métodos

- Tipos usados em Java:
  - package
  - public
  - private
  - protected

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Modificadores de acesso

public: um nível sem restrições, equivalente a não encapsular;

private: o nível de maior restrição em que apenas a própria classe pode ter acesso a variáveis e/ou métodos. É o tipo mais usado para implementar o encapsulamento.

protected: um nível intermediário de encapsulamento em que as variáveis e métodos podem ser acessados pela própria classe ou por suas subclasses.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Modificadores de acesso

package: nível em que a classe pode ser acessada apenas por outras classes pertencentes ao mesmo pacote.

Visibilidade	public	protected	package	private
Da mesma classe	S	S	S	S
De qualquer classe do mesmo pacote	S	S	S	N
De qualquer classe fora do pacote	S	N	N	N
De uma subclasse do mesmo pacote	S	S	S	N
De uma subclasse fora do pacote	S	S	N	N

http://slideplayer.com.br/slide/345265/

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Por que encapsular?

Garantia de acesso seguro aos dados.

Tornar transparentes as alterações em objeto.

Permite reutilizar o objeto em qualquer lugar.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Métodos set e get

Um dos métodos utilizados para usarmos os valores privados de um encapsulamento

```
// atributo privado
private double raio;
```

Variável de instância

```
// método alterar raio
public void setRaio(double r)
{
    raio = r;
}
```

Método set (altera)

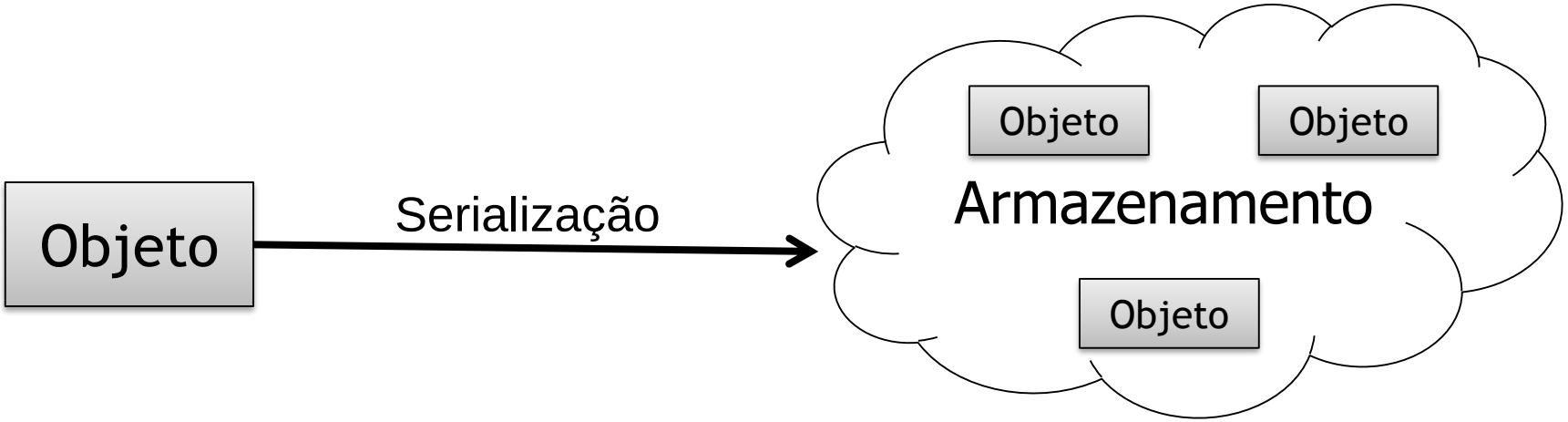
```
// método informar raio
public double getRaio()
{
    return raio;
}
```

Método get (informa)

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Persistência

Persistência de dados em Java é utilizada quando se deseja guardar objetos ou estrutura de dados por um tempo indeterminado, sendo possível recuperá-los posteriormente.



Para persistir dados os mesmos precisam ser serializados.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Serialização

✓ Captura o estado do objeto ou a estrutura de dados, e transforma em uma cadeia de bytes.

✓ Recupera os bytes persistidos para fazer o processo inverso, e ter os dados de volta para a aplicação em execução.

✓ Serializar um objeto, em Java, só é possível caso sua classe esteja definida como ***serializável***.

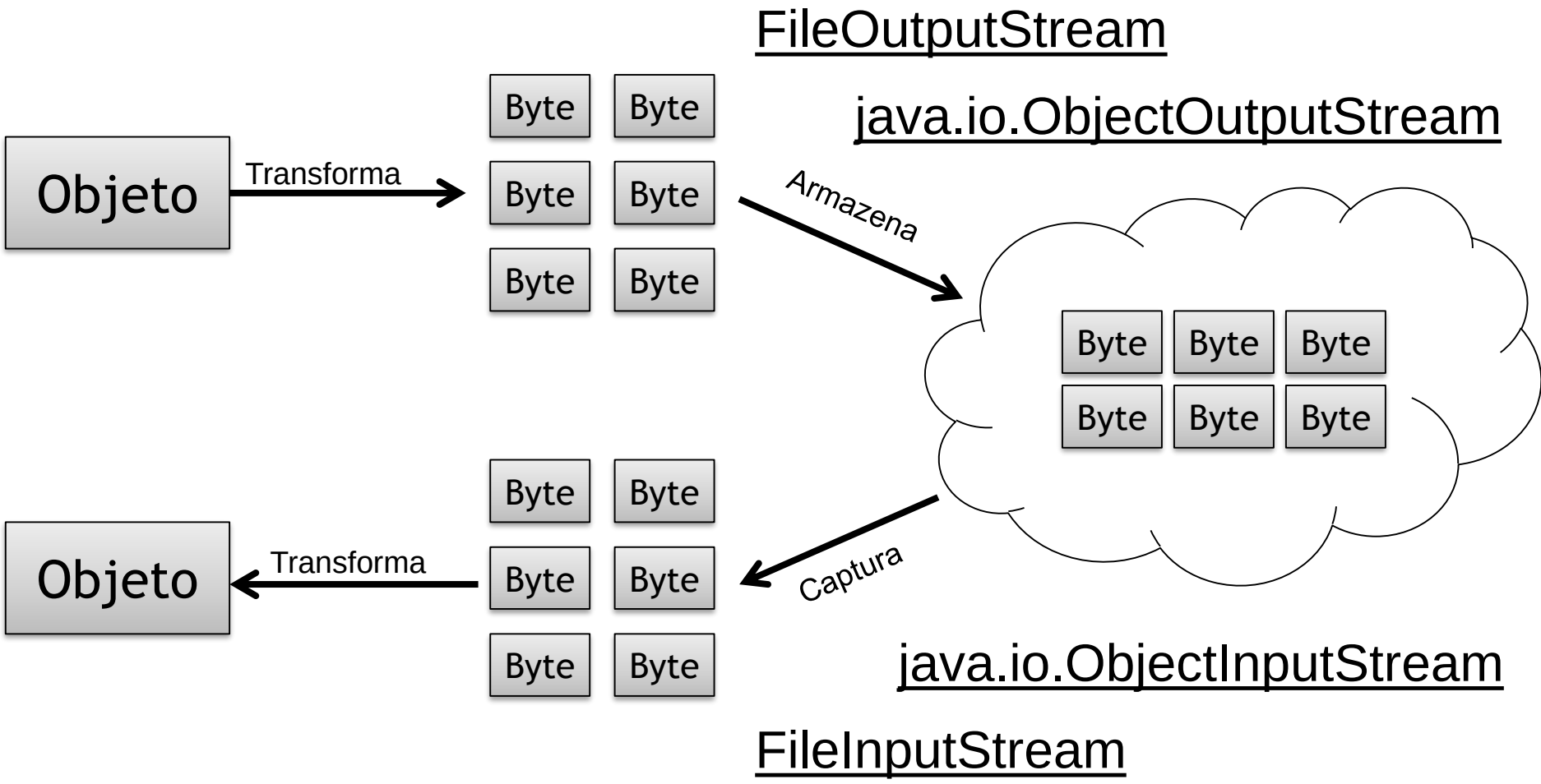
Sintaxe:

```
import java.io.Serializable;

public class NomeDaClasse implements Serializable
{...}
```

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Serialização



Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Serialização

✓ **java.io.FileOutputStream** é um fluxo de arquivo que permite a gravação em disco. Trabalham com arquivos.

✓ **java.io.FileInputStream** é justamente o contrário, permitindo a leitura de um arquivo em disco. Trabalham com arquivos

✓ **java.io.ObjectOutputStream** semelhante ao **FileOutputStream** e trabalham com objetos.

✓ **java.io.ObjectInputStream** semelhante ao **FileInputStream** e trabalham com objetos.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Exercícios

1. Associe os termos ao seu significado em relação ao encapsulamento.
    a) public
    b) private
    c) protected
    d) Package

I) Nível em que a classe pode ser acessada apenas por outras classes pertencentes ao mesmo pacote.
II) Um nível intermediário em que as variáveis e métodos podem ser acessados pela própria classe ou por suas subclasses.
III) Um nível sem restrições, equivalente a não encapsular.
IV) O nível de maior restrição em que apenas a própria classe pode ter acesso a variáveis e/ou métodos. É o tipo mais usado para implementar o encapsulamento.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Exercícios

2. Faça uma classe de nome Cilindro com os atributos raioBase (double) e altura (double). Faça o Encapsulamento desta Classe por meio dos métodos "set" e "get“. Os atributos devem ser privados. Elabore um método que retorna o valor do volume do cilindro.

3. Crie uma classe “UsaCilindro" (com o método main) que:

a) instancia a classe "Cilindro" (da questão 2)

b) inclui valores nos seus atributos por meio dos métodos "set".

c) Apresente em tela o valor do volume.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Exercícios

4. Tendo como base a classe Funcionario abaixo:

```
3  public class Funcionario {
4      public String nome;
5      public double salario;
6      public Funcionario(String nome, double salario) {
7          this.nome = nome;
8          this.salario = salario;
9      }
10     public void mostra() {
11         System.out.println(nome + ", ganha " + salario + " reais");
12     }
13     public void aumentaSalario(double aumento) {
14         salario *= 1 + aumento / 100;
15     }
16 }
```

```
3  public class UsaFuncionario {
4      public static void main(String[] args) {
5          Funcionario motorista = new Funcionario("Emmanuel Ferreira", 2590);
6          motorista.aumentaSalario(10);
7          motorista.mostra();
8      }
9  }
```

Altere o modificador de acesso dos atributos e do construtor de "public" para "private" e descreva o que ocorre na classe UsaFuncionario.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Exercícios

5. Para solucionar o problema do exercício anterior crie os métodos set() e get() para cada variável.

6. Modifique a classe UsaFuncionario para fazer as modificações nos valores das variáveis usando os métodos set().

7. Usando os conceitos de encapsulamento, elabore uma classe chamada aluno contendo os atributos nota1 e nota2. Esses só podem aceitar valores entre 0 e 10. Elabore também um método chamado getMedia que retorna a média aritmética entre as duas notas.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Exercícios

8. Faça uma aplicação que implemente uma classe chamada Conta (conta bancária) que contenha um atributo numérico chamado saldo. Encapsule esse atributo de maneira que uma classe externa não tenha acesso direto a ele. A seguir, elabore três métodos públicos chamados depositar, sacar e consultar. Esses métodos permitirão manipular o valor do saldo, conforme a descrição seguinte:

    a) depositar: recebe um valor e o adiciona ao saldo. O depósito não pode ser realizado entre 0h00 e 6h00.
    b) sacar: recebe um valor e o subtrai do saldo; O saque não pode ser superior ao saldo. O saldo nunca pode ser negativo.
    c) consultar: apresenta o valor do saldo em tela.

Elabore outra classe para testar as funcionalidades da classe Conta.

Prof. Dr. Sérgio Furgeri: sergio.furgeri@fatec.sp.gov.br

Exercícios

9. Faça uma classe chamada Cliente que implemente Serializable que contenha os atributos privados nome, sexo e cpf. Implemente nessa classe:

a) Métodos get e set;

b) Construtor com todos os atributos;

c) Método toString para retornar os dados de todos os atributos.

d) Um método para persistir os dados no disco

e) Um método para ler os dados do disco

10. Crie outra classe para gerar manipular um objeto serializado testando as funcionalidades do exercício anterior.