

Material de Apoio 4

Súmary

Este material reforça os conceitos de classe, objeto, métodos e atributos e discute a visibilidade dos membros (atributos e métodos) dos objetos. O estudo é conduzido através do acompanhamento de um exemplo desenvolvido em Java. Outros assuntos relacionados à aula: métodos construtores e sobre-carga de métodos. Revisões necessárias: noção de escopo, encapsulamento, variável e referência. Como tópico extra, a aula apresenta a estrutura de arquivos de Java (não consta neste material). Os exercícios propostos auxiliam na absorção dos conceitos apresentados; a prática deve ser obtida através da utilização de um ambiente de programação Java: sugere-se o uso do BlueJ.

Classe e Objetos

As classes, ou tipos abstratos de dados, são definidas pelo programador em tempo de construção de seu programa. É possível que classes relacionem-se entre si, alguns destes relacionamentos, como a agregação e a herança, serão discutidos nas próximas aulas. Uma classe consiste em uma especificação de um conjunto de objetos que possuem a mesma estrutura, a qual deve ser representada em termos de seu estado interno (atributos) e de seu conjunto de serviços (métodos).

Os objetos são instanciados explicitamente a partir de uma classe específica durante a execução do programa. Cada objeto reproduz a estrutura definida pela classe. É importante observar que dois objetos criados de uma mesma classe possuem a mesma estrutura mas cada um possui uma área de memória própria para armazenar seus atributos. Portanto, a execução de um método em um objeto tem acesso somente ao estado interno do próprio objeto. Assim, caso dois objetos criados da mesma classe venham a receber uma mesma mensagem, a resposta não é necessariamente a mesma.

Exercícios

- De uma definição para os seguintes termos:
 - Atributo: _____

 - Método: _____

 - Estado interno: _____

 - Instanciação: _____

 - Encapsulamento: _____

 - Invocação: _____

 - Mensagem: _____

- Por que uma mesma mensagem (invocação do mesmo método passando o mesmo conjunto de parâmetros) enviada para dois objetos criados da mesma classe não retornam, necessariamente, o mesmo resultado?
- Suponha a implementação de um procedimento ou uma função em uma linguagem do paradigma imperativo. Neste caso, a invocação de um subprograma depende de quais informações (dados)? Considerando este fato, a invocação de um procedimento e/ou função com o mesmo conjunto de dados deve, obrigatoriamente, retornar o mesmo conjunto de dados? Isto seria desejável?
- A resultado da execução de um método pode ter seu resultado influenciado pela execução anterior de um outro método no contexto do próprio objeto? Caso positivo, diga como. Caso negativo, justifique.

Método construtor

Uma classe consiste em um tipo abstrato de dados. Tal como um tipo de dado qualquer, devem ser definidos os valores iniciais para sua representação de memória interna. Ou seja, o estado interno do objeto deve ser inicializado no momento em que o objeto é criado atribuindo valores para os atributos. Esta inicialização é efetuada por um método particular, denominado *método construtor*. O método construtor de um objeto é executado implicitamente no momento em que um objeto é criado e executa no escopo deste objeto. Em Java e em C++ este método é identificado pelo próprio nome da classe e, diferentemente dos demais métodos, não é previsto nenhum tipo de retorno para este método.

A Figura 1 ilustra o corpo de uma classe em Java.

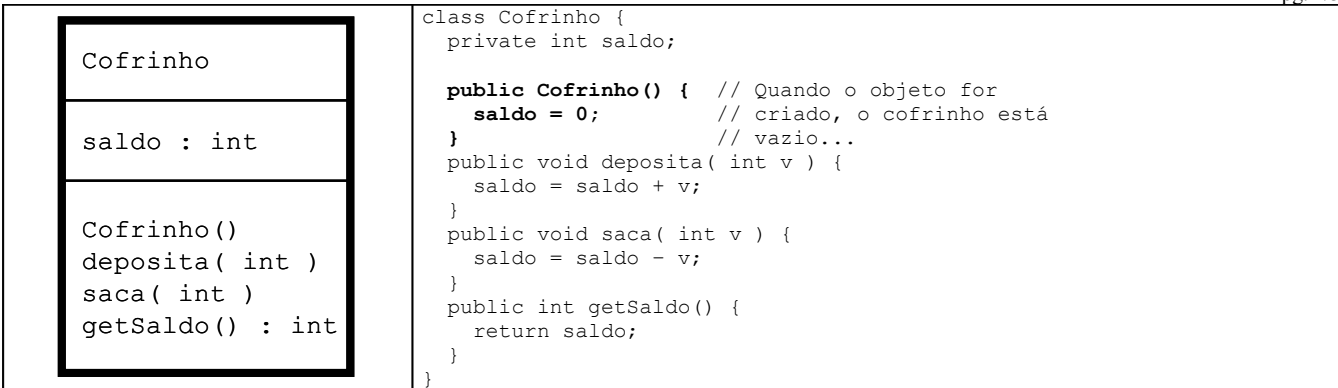


Figura 1: Representação da classe Cofrinho

Exercícios

5. Represente a evolução da memória após a execução de cada uma das linhas do trecho de código abaixo.

```
1. Cofrinho c1,
2.     c2;

3. c1 = new Cofrinho();
4. c2 = new Cofrinho();

5. c1.deposita(10);
6. c2.deposita(20);
7. System.out.println( c1.getSaldo() );
8. System.out.println( c2.getSaldo() );

9. c1.saca(5);
10. System.out.println( c1.getSaldo() );
11. System.out.println( c2.getSaldo() );
```

6. Represente a evolução da memória após a execução de cada uma das linhas do trecho de código apresentado acima.

Escopo de visibilidade

Quando da definição de uma classe, o programador define os membros que deveram compor a estrutura dos objetos que a partir dela forem criados, notadamente os seus atributos e métodos. Temos visto em nossos exemplos que alguns destes membros tem sido definidos como `private` e alguns como `public`. Isto significa que os referidos membros ou podem ser acessados somente no escopo do objeto (privado) ou pertencem a interface do objeto (público). Tipicamente métodos são públicos e atributos são privados.

Exercícios

7. É possível que um atributo seja definido como `public`, no entanto isto não é interessante. Justifique esta afirmação.
8. Apresente, caso exista, um exemplo de situação onde seja interessante o uso de um atributo público.
9. É possível que um método seja definido como `private`, e o uso deste recurso pode interessante em determinados casos. Justifique esta afirmação.
10. Caracterize uma situação em que se faça interessante o uso de um método privado.
11. Implemente uma classe `Aluno`. Esta classe deve possuir como atributos o nome (`String`), as notas de GA e GB (`double`), e a frequência (`int`). Além do construtor, objetos desta classe devem contar com os métodos `setGA`, `setGB` e `setFrequencia`, de forma a introduzir os respectivos valores a medida que o semestre evolui, e o método `getMedia`, que retorna a média do individuo.
12. Reimplemente a classe `Aluno` introduzindo um novo método: `situacao`. Este método deve retornar `true` se o aluno foi aprovado (média igual ou superior a 6.0 e frequência mínima de 15 aulas) e `false` caso contrário.
13. Implemente uma classe `Carro`. Objetos desta classe possuem alguns atributos, tais como: matrícula, consumo por litro, contador de quilometragem e quantidade de litros de combustível no tanque (defina demais atributos conforme sua necessidade especifica). Esta classe deve conter, no mínimo, métodos para realizar os seguintes serviços:
 - Abastecer o carro;
 - Fazer o carro andar um número determinado de quilômetros;
 - Fazer o carro andar uma quantidade de quilômetros definida aleatoriamente (`random` da classe `Math`);
 - Mostrar quantidade de combustível no tanque;
 - Mostrar o status do veículo: quantidade de combustível, quilômetros percorridos e quantidade de quilômetros que podem ser percorridos com a quantidade de combustível disponível.

Sobrecarga de método

Um método é reconhecido pela sua assinatura. A assinatura de um método é identificada pelo nome do método e o tipo dos parâmetros que ele recebe. No exemplo da Figura 1, temos a assinatura de quatro métodos:

- O método construtor sem parâmetros;
- O método `deposita` que recebe um valor inteiro;
- O método `saca` que recebe um valor inteiro;
- O método `getSaldo` sem parâmetros.

Quando uma mensagem é enviada a um objeto, o serviço (método) acionado é aquele que responde satisfatoriamente a mensagem recebida. Desta forma um interessante recurso é disponibilizado ao programador: a sobrecarga de método. Alguns autores nomeiam este recurso de programação como polimorfismo estático – nós não utilizaremos esta nomenclatura. A sobrecarga de método permite identificar diferentes serviços pelo mesmo nome de método, diferenciando-os pelo número e tipo dos parâmetros.

Um exemplo de uso deste recurso é apresentado na Figura 2, com duas implementações possíveis para a construção de objetos da classe `Cofrinho`, considerando depósitos iniciais.

```
class Cofrinho {
    private int saldo;

    public Cofrinho() { // Quando o objeto for
        saldo = 0;    // criado, o cofrinho está
    }                // vazio...
    public Cofrinho( int aux ) { // Recebe um deposito inicial
        saldo = aux;    // o cofrinho e´ criado com valor inicial
    }
    // restante da classe
}
```

Figura 2: Exemplo de sobrecarga de método no construtor da classe `Cofrinho`

Exercícios

14. Construa uma classe `ContaEmBanco`. Atributos mínimos necessário: nome do cliente e saldo. Na construção, no mínimo nome do cliente deve ser informado, opcionalmente com um depósito inicial.
15. Defina uma classe para conter informações sobre um funcionário de uma empresa (classe `Funcionario`). Quais são os atributos desta classe? Inclua entre eles, o salário que o funcionário deve receber por hora trabalhada. Implemente, para esta classe, pelo menos três métodos construtores: um que receba apenas o nome do funcionário e assuma valores default para os demais atributos (assuma que o funcionário deve receber 2 pilas por hora trabalhada); o segundo construtor deve receber, além do nome, o valor que o referido trabalhador deve receber por hora trabalhada. Identifique e implemente demais métodos que achar conveniente para um objeto da classe `Funcionario`.
16. Leia páginas 71 a 73 do livro: *Conceitos de Computação com o Essencial de Javam*, de Cay Horstmann, 3ª Ed., Porto Alegre, Bookman, 2005 – Seção “Como Fazer 2.1, Projetando e Implementando Classes”.
17. Implemente, com auxílio do BlueJ as classes desenvolvidas neste material.
18. Comente o programa abaixo, linha a linha.

```
public class teste {
    public static void main( String args [] ) {
        String nome;
        BufferedReader keyboard;

        keyboard = new BufferedReader(new InputStreamReader(System.in));
        try{
            nome = keyboard.readLine();
        }
        catch(IOException e){
            System.out.println(e);
            return;
        }

        nome.toUpperCase();

        System.out.println("o nome " + nome + " tem " + nome.length () + "
caracteres. ");
    }
}
```