

Prática 05 – Estrutura de uma Classe

por Frederico C. G. Pereira

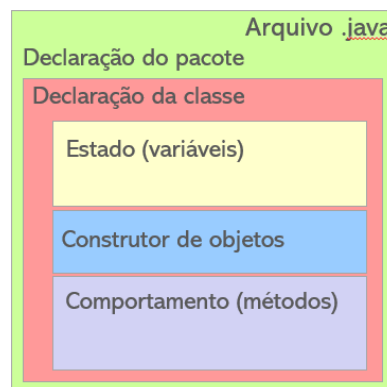
© Copyright 2025

Objetivos

- Praticar a declaração de classes
- Definir variáveis de instância
- Definir construtores
- Definir variáveis de classe
- Definir métodos de classe

I. Estrutura de uma Classe

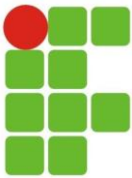
Uma classe em Java possui a seguinte estrutura sintática:



Uma classe descreve uma categoria de objetos que compartilham uma estrutura (variáveis) e operações (métodos) em comum. A classe funciona como um gabarito para criar vários objetos distintos. Cada vez que se usa a classe para instanciar um objeto é reservada na memória área suficiente para que todas as variáveis sejam alocadas. Assim, se uma classe possui uma variável do tipo **int** (4 bytes), uma **boolean** (1 byte) e uma **double** (8 bytes), quando ele for instanciado, sua área de memória compreenderá 13 bytes no total. Se instanciarmos 10 destes objetos, a memória alocada para objetos será de 130 bytes. Estas variáveis são exclusivas de cada objeto, assim, se mudarmos o valor de uma delas, isto afeta apenas aquele objeto. Por isto que estas variáveis são chamadas de **variáveis de instância**. Cada instância (ou objeto) possui sua cópia.

Em contrapartida, as **variáveis de classe** pertencem à classe, ou seja, existe apenas uma cópia delas na memória (porque a classe também é carregada para a memória e somente uma vez cada classe). As variáveis de classe surgem para evitar que uma variável que seja compartilhada por todos os objetos daquela classe não seja repetida (com o mesmo valor) em todos os objetos dela. Assim, uma classe que modelasse um celular **SamsungS24** teria seu número e seu IMEI representado por variáveis de instância (cada celular tem seu próprio número e IMEI), mas o nome do fabricante (uma **string** "Samsung") seria uma variável de classe, pois todos os celulares possuem o mesmo valor para este dado.


Construtores são blocos de código especiais que servem para atribuir valores às variáveis de um objeto (em geral as de instância) logo após área de memória ser alocada para elas através do operador **new**. O construtor possui um nome que é exatamente o mesmo nome da classe, uma lista opcional de parâmetros e um corpo onde está o bloco de código que executa a inicialização do objeto. Não possuem tipo de retorno, nem mesmo **void**. Se o programador não declara um construtor em uma classe, o compilador Java provê um construtor chamado de padrão que não possui parâmetros e que também não faz nada... Ele existe apenas para manter a regularidade da sintaxe de instanciação de objetos Java que é sempre na seguinte forma:

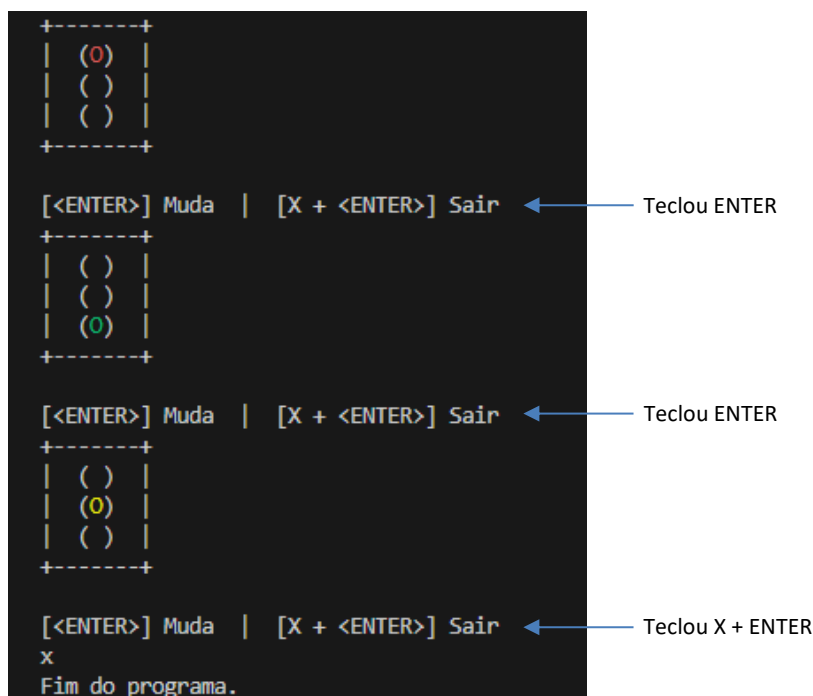


```
ClasseTipo referencia = new ClasseTipo();
```

Caso o programador implemente um construtor com uma lista de parâmetros, Java não fornece mais o construtor padrão sem parâmetros. Se o programador precisar usar o seu construtor com parâmetros, mas também o sem parâmetro, deverá implementar os dois na classe. Esta característica de Java de permitir que construtores com diferentes implementações e diferentes listas de parâmetros possuam o mesmo nome é chamado de **sobrecarga** ou ainda **polimorfismo paramétrico**. Do grego Polýs (muitas) + Morphé (forma).

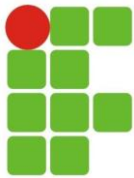
O objetivo desta prática é fazer você modelar uma classe a partir de uma declaração de um problema e escrever um pequeno programa para usá-la.

- 1)  Implemente uma classe que modele um semáforo de trânsito! Que propriedades e métodos você implementaria? Sua classe deve ter constantes, construtores e, talvez, até métodos de classe ou métodos privados (não é garantia, depende da sua modelagem). Além da classe Semaforo, implemente também uma classe de aplicação SemaforoApp que permita ao usuário dizer quando quer que o semáforo mude de estado. Exemplo de telas que esta aplicação deve apresentar ao usuário (para ficar mais interessante o resultado, limpe a tela sempre que o usuário pressionar algo na entrada):



Dicas para embelezamento da interface com o usuário:

- Para limpar a tela: `System.out.println("\033[H\033[2J");`
- `System.out.println("\033[31mTexto Vermelho\033[0m");`
- `System.out.println("\033[32mTexto Verde\033[0m");`
- `System.out.println("\033[33mTexto Amarelo\033[0m");`



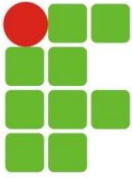
- 2) Um banco deseja ter um sistema simples para representar contas de clientes. Cada conta deve guardar informações essenciais, como o número da conta e o nome do titular. Ela também precisa manter o valor de suas movimentações. Além disso, toda conta pertence a uma mesma instituição bancária, que não muda. É necessário poder criar contas já com um valor inicial para seu número e titular ou começar com valores "zerados". As operações esperadas para a conta são: **depositar valores, retirar valores e consultar o saldo, imprimir o extrato, mudar o nome do correntista, converter conta em String**.
- a) Implemente uma classe que represente essa entidade
 - b) Implemente uma classe com um main que instancie uma conta, realize operações diversas sobre ela (com dados literais no main)
 - c) Implemente uma outra classe de aplicação (com método main) que tenha uma interface via console para pedir ao usuário os dados e exibir os resultados. O programa deve iniciar pedindo ao usuário dados (número e titular) para criar uma conta. Depois ele mostra um menu com as opções 1-dados da conta, 2-depositar, 3-retirar, 4-consultar saldo, 5-extrato, X-sair. O programa deve realizar na conta a respectiva operação ou encerrar sua execução.
- 3) Uma escola precisa representar uma turma de alunos. Toda turma possui um nome (ex: "TSI-POO") identificador e o nome do professor responsável. Ela também deve ter a **capacidade máxima de alunos** que pode comportar. Os nomes dos alunos devem ser armazenados em uma lista fixa (um array de String de tamanho igual à capacidade). No momento de criar a turma, deve ser possível apenas indicar o nome e o professor, ou ainda informar também a capacidade máxima de alunos. A escola precisa ter operações para: **adicionar alunos na turma, listar os nomes dos alunos já cadastrados e informar quantas vagas ainda restam**.
- a) Implemente a classe que represente essa entidade.
 - b) Crie uma classe com um programa main que a utilize. Esta classe deve pedir os dados ao usuário.
- 4) Um sistema de rede precisa representar endereços numéricos no padrão IPv4. Cada endereço é formado por quatro octetos (valores numéricos inteiros de 0 a 255). É importante que o endereço possa ser criado a partir dos quatro números ou, alternativamente, a partir de uma única string no formato "192.168.0.1". O sistema também deve garantir que os valores usados estejam dentro dos limites válidos, rejeitando números inválidos. Uma vez criado, o endereço deve permitir:
- a) mostrar o endereço em formato de string no padrão usual;
 - b) calcular o endereço de rede de um IP, dada uma máscara de sub-rede;

Exemplo:

IP: 10.3.2.121
Máscara: 255.255.0.0
Resultado: 10.3.0.0

```
IPv4 ip = new IPv4("10.3.2.121");  
IPv4 mask = new IPv4("255.255.0.0");  
IPv4 netaddr = ip.subnet(mask);    //netaddr ← 10.3.0.0
```

- c) Escreva uma classe de aplicação que peça ao usuário para digitar o IP, a máscara e exiba o endereço de subrede como resultado.
- d) Dicas:



- O que é uma subrede? <https://www.cbtnuggets.com/blog/technology/networking/networking-basics-what-is-ipv4-subnetting>
- O operador Java '&' faz um AND bit-a-bit entre dois bytes. Exemplo:

```
byte mask      = 0xF0;    //em binário 11110000
byte octet     = 0xAA;    //em binário 10101010
byte subnetaddr = a & b;  //c =      10100000 ou 0xA0
```

Perceba, onde a máscara tem 1, deixa passar o valor equivalente do octeto, onde a máscara é zero, anula o bit do octeto, seja qual for. A máscara acima preservou no resultado os mesmos bits do octeto para os 4 bits mais significativos (1111) e anulou os bits do octeto para o 4 menos (0000).

11110000	<div style="border: 1px solid black; padding: 5px; display: inline-block;">& (AND)</div>
10101010	
10100000	

Diagram illustrating the bit-wise AND operation. The first row shows the mask (11110000) and the octet (10101010). The second row shows the result (10100000) after applying the AND operation. Blue arrows point from the mask bits to the result bits, indicating that only the bits where both mask and octet are 1 are preserved.