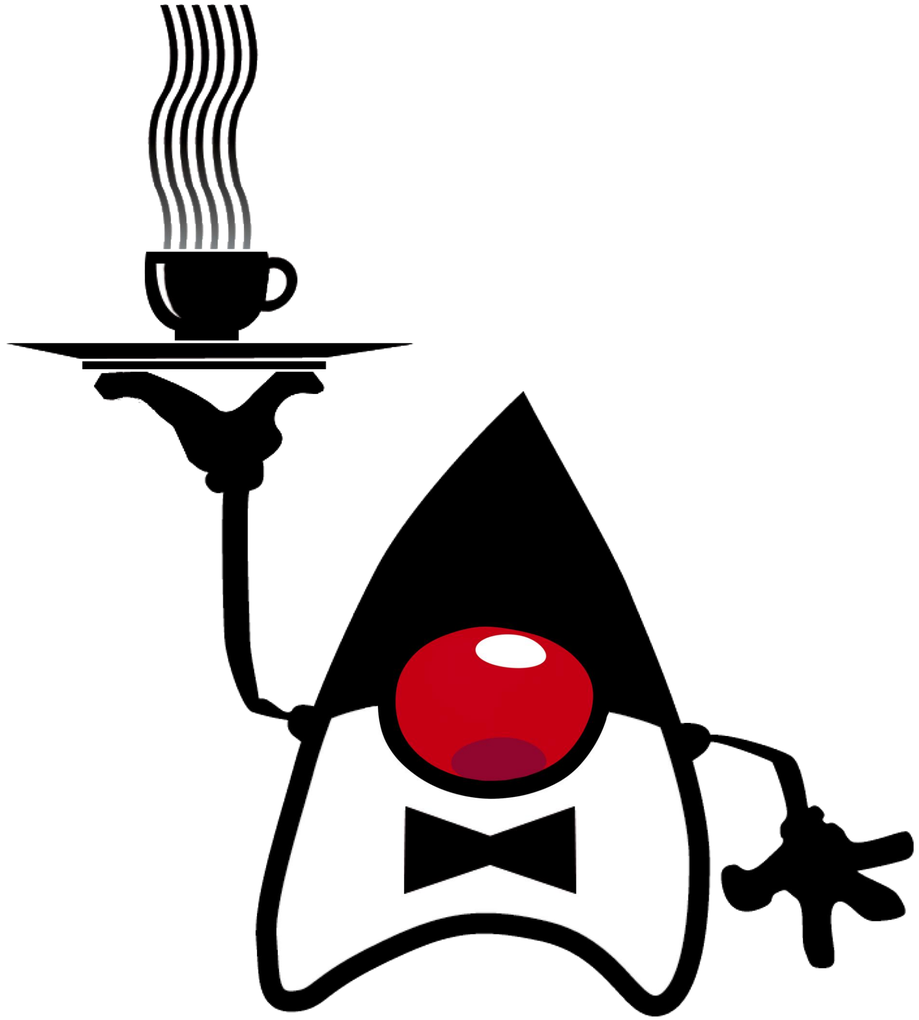
Trilha Java

Encontro 16 – Exemplo e Atividade 8



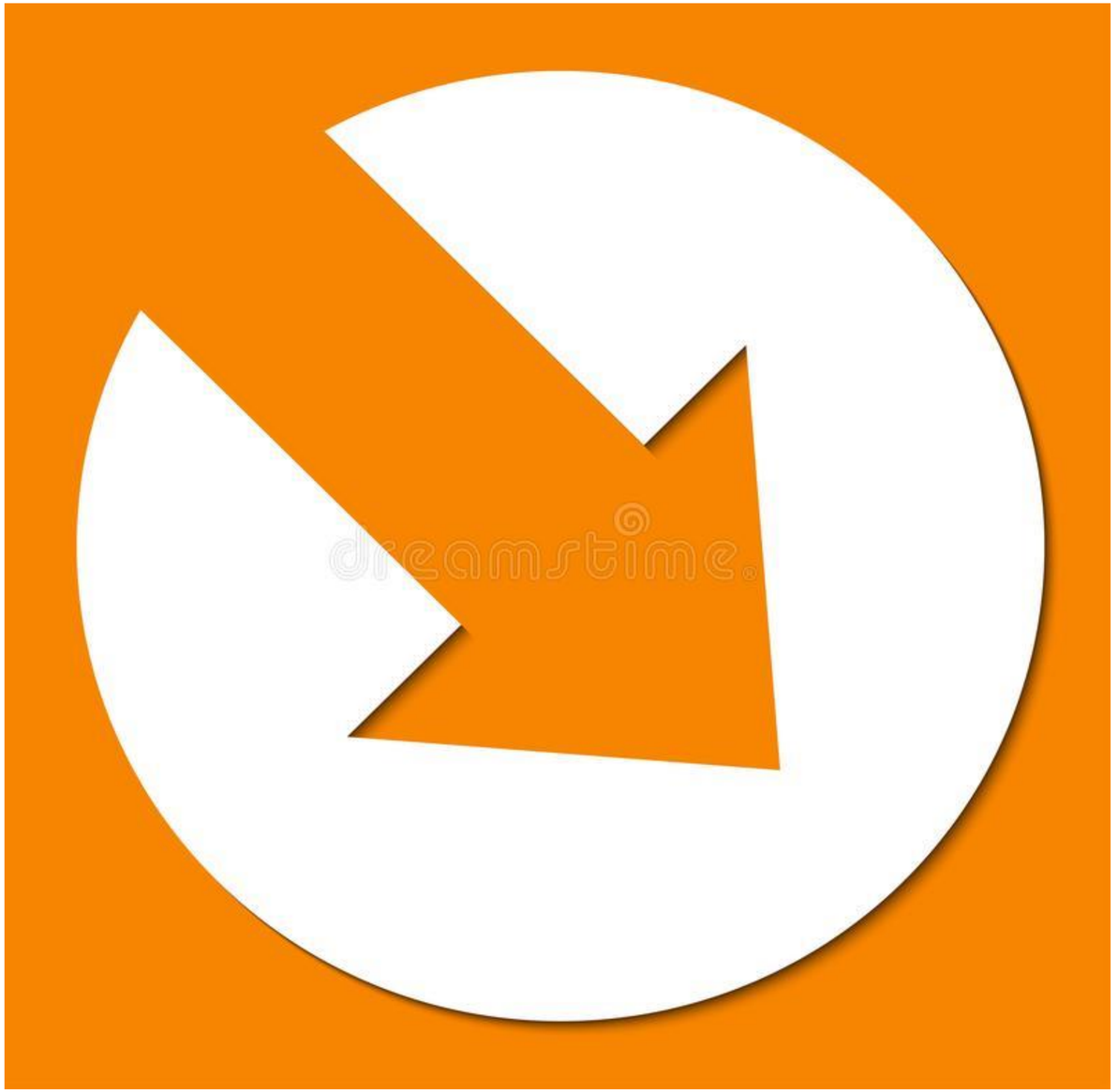
vnt/school
powered by venturus

Recapitulação

1. UML.
2. Visibilidade.

   Atributos e Métodos
3. Métodos Especiais.

   Get, Set, Construtor



vnt/school
powered by venturus

Atividade 1

Crie um programa com as seguintes características:

Crie uma classe “**ContaBanco**”.

- O “tipo” de conta só pode ser conta corrente “CC” ou conta poupança “CP”.
- Na hora de “abrirConta()” lembre-se que é necessário mudar o “status” pra verdadeiro. Se a conta não está aberta o “status” é falso.
- Suponha que se a pessoa abrir uma conta “CC” a pessoa ganha “R$50,00” e se abrir uma conta “CP” ganha “R$150,00”.

vnt/school
powered by venturus

Atividade 1

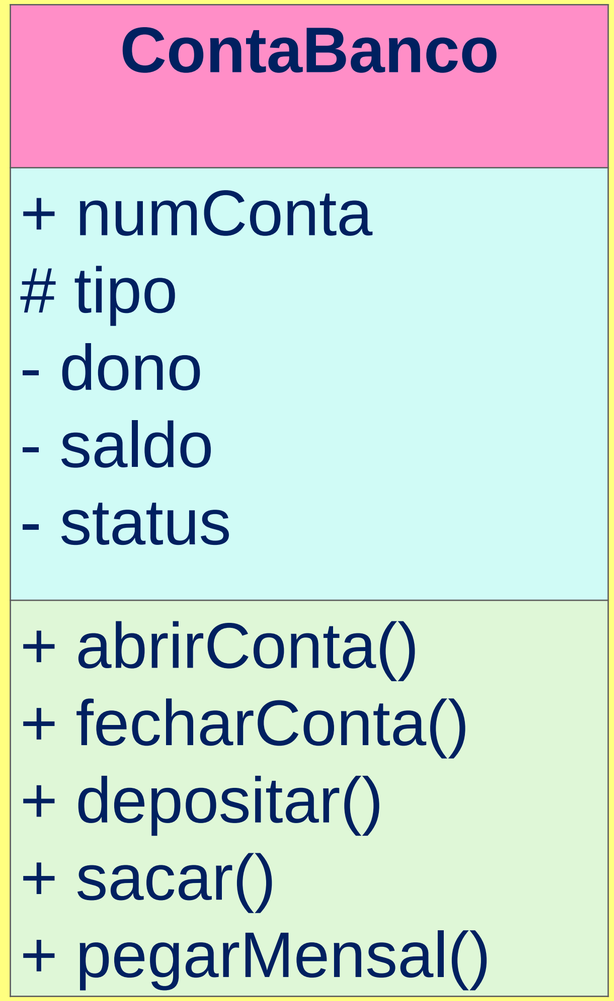
- Para o método “fecharConta()” lembre-se que a pessoa não pode ter dinheiro e nem dívida na conta.
- Para o método “depositar()” veja que só é possível se o “status” estiver “Verdadeiro”.
- E pra “sacar()” dinheiro só é possível se a conta estiver aberta, ou seja, com “status” verdadeiro e além disso, é necessário ter dinheiro. E por fim, o cliente vai pagar mensalmente uma taxa “pagarMensal()”. Cliente “CC” paga “R$12,00” e cliente “CP” paga “R$20,00”.

vnt/school
powered by venturus

Atividade 1

De acordo com o diagrama de classes, insira os atributos e métodos. Veja as visibilidades de cada atributo e método. Além disso, insira os métodos Get e Set para cada atributo.

Insira também um “construtor”, onde cada conta que é aberta já recebe o “status” como falso, ou seja está fechada ainda. E o “saldo” como “zero”.



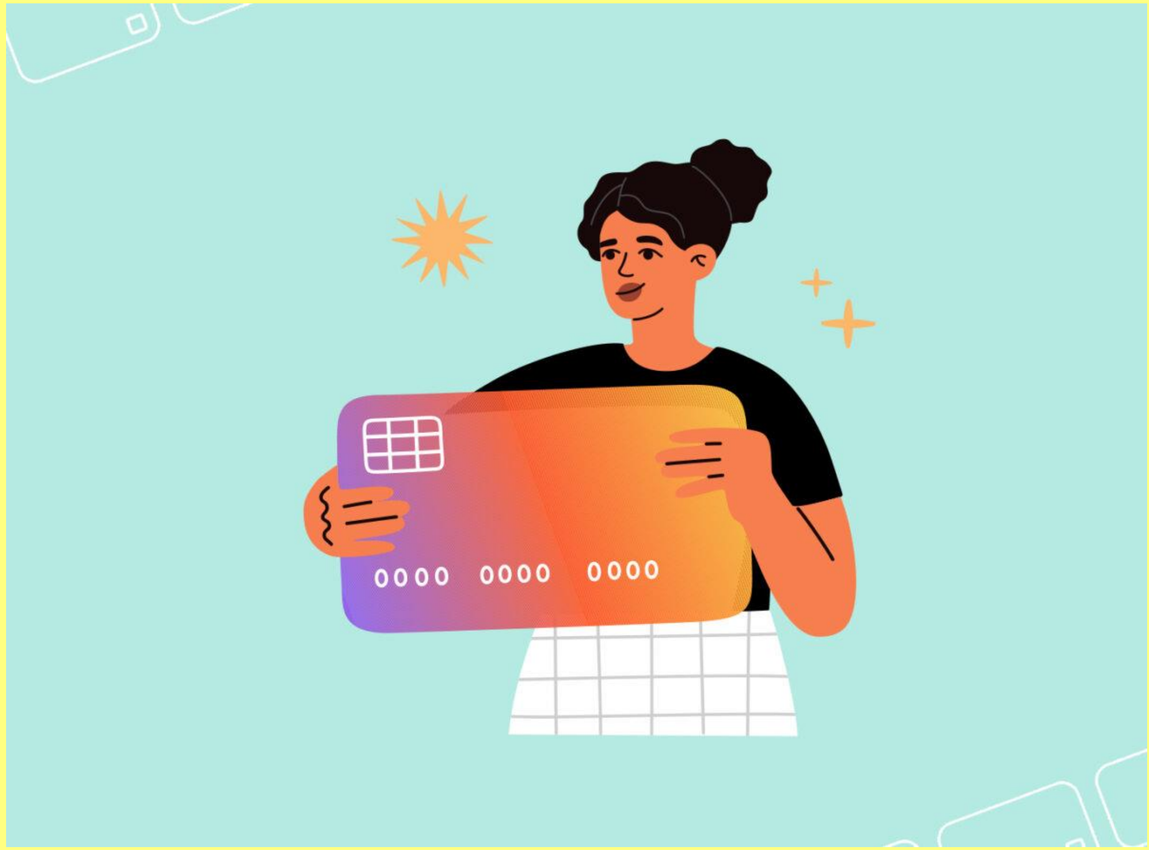
vnt/school
powered by venturus

Atividade 1

Por fim, instancie:

A conta corrente 0001 do João Silva, que vai precisar depositar R$300,00.

A conta poupança 0002 de Maria Silva que vai precisar depositar R$400,00.



vnt/school
powered by venturus

Atividade 2

Crie uma classe Carro com as seguintes características:

Use o construtor para definir o início padrão do carro. Ele deve estar desligado e marcha zero.

Insira os getters e setters.

Dentro do método desligar(), avise com uma mensagem "Carro está desligado".

Carro
+ cor + marca + ano + marcha + ligado + velocidadeAtual + velocidadeMaxima
+ ligar() + desligar() + acelegar() + pegarMarcha

vnt/school
powered by venturus

Atividade 2

Dentro do método ligar(), avise com a mensagem quando estiver ligado "O carro está ligado". E ao acelerar que apresente o valor da velocidade Atual.

Dentro do método acelerar crie a lógica necessária. Considere uma quantidade/aumento de 10km/h para cada acelerada que for incrementado.

Carro
+ cor + marca + ano + marcha + ligado + velocidadeAtual + velocidadeMaxima
+ ligar() + desligar() + acelegar() + pegarMarcha

vnt/school
powered by venturus

Atividade 2

Dentro do método acelerar crie a lógica necessária. Considere a seguinte marcha:

(-1) Ré : velocidade < 0
(0) Ponto Morto : velocidade = 0
(1) Media: 0 < velocidade < 40
(2) Alta : 40 =< velocidade =< 80
(3) qualquer valor diferente do previsto.

Carro
+ cor + marca + ano + marcha + ligado + velocidadeAtual + velocidadeMaxima
+ ligar() + desligar() + acelegar() + pegarMarcha

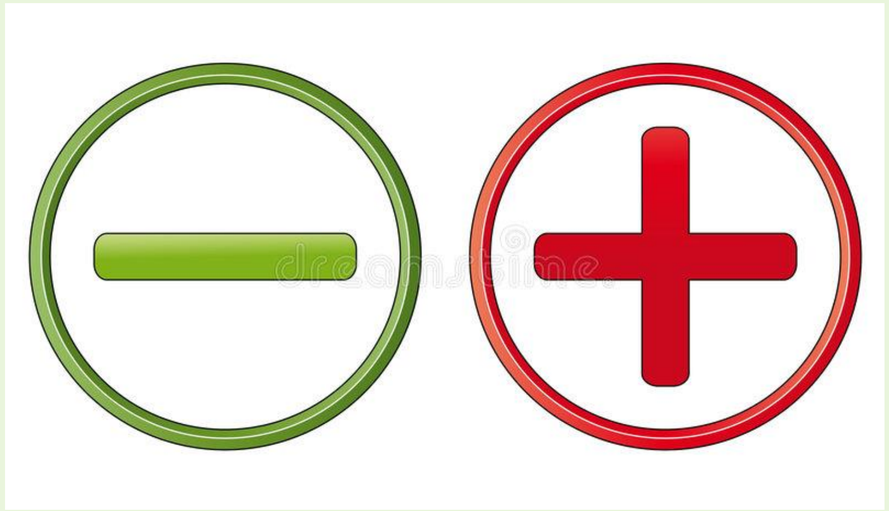
Atividade 2

Por fim, instancie dois objetos carro. Teste os carros com características e comportamentos diferentes.

vnt/school
powered by venturus

Atividade 3

Faça uma programa para ler um número inteiro (N), e depois dizer se este número é negativo ou não. Além disso dizer também se este número é par ou ímpar. Crie o programa principal e crie uma classe **operaçãoMatematica** com dois métodos, **ParOuImpar()** e **NegativoOuPositivo()**, onde deve conter a lógica do problema. Considere os métodos como públicos. E Considere a variável N como protegido. Depois tente trocar para privado e discuta o resultado.



vnt/school
powered by venturus

Atividade 4

Baseado na tabela ao lado, escreva um programa que leia o código de um item e a quantidade deste item. A seguir, calcule e mostre o valor da conta a pagar. Crie o programa principal para instanciar os objetos e crie a classe **Produtos** para criar lógica do problema. Implemente as variáveis/atributos como protegido e os métodos como píblico.

Código	Produto	Preço
1	Cachorro-Quente	7.00
2	X-Salada	9.00
3	X-Bacon	11.00
4	Torrada	5.00
5	Refrigerante	4.00

Tratamento de Exceções

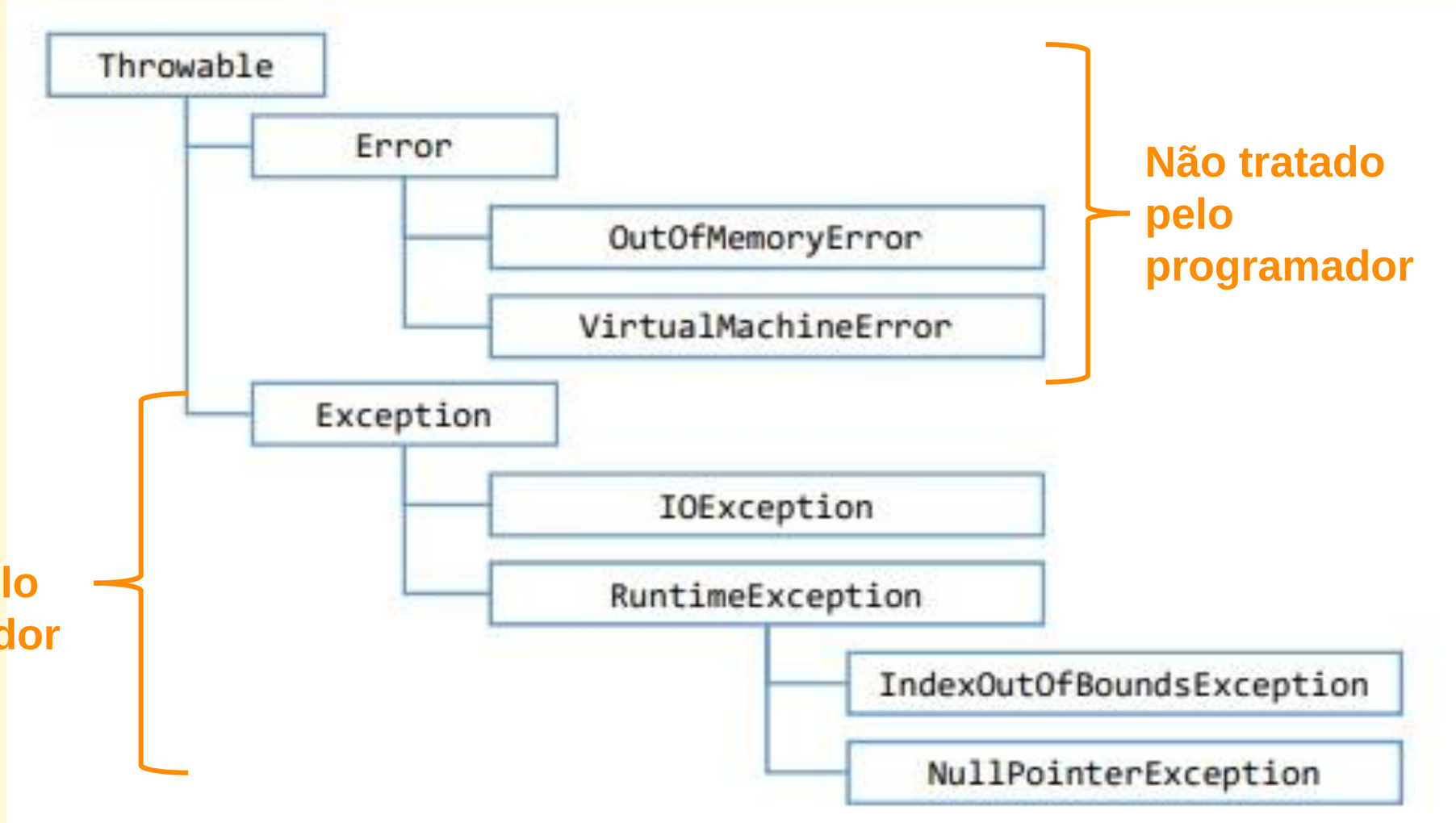
Uma **exceção** é qualquer condição de erro ou comportamento inesperado encontrado por um programa em execução.

Em Java, uma exceção é um objeto herdado da classe:
java.lang.Exception - o compilador obriga a tratar ou propagar
java.lang.RuntimeException - o compilador não obriga a tratar ou propagar

Quando lançada, uma exceção é propagada na pilha de chamadas de métodos em execução, até que seja capturada (tratada) ou o programa seja encerrado

Hierarquia de Exceções

[java.lang Class Hierarchy (Java SE 10 & JDK 10) (oracle.com)]



vnt/school
powered by venturus

Porque Exceções

O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas.

Vantagens:
Delega a lógica do erro para a classe responsável por conhecer as regras que podem ocasionar o erro.

Trata de forma organizada (inclusive hierárquica) exceções de tipos diferentes.

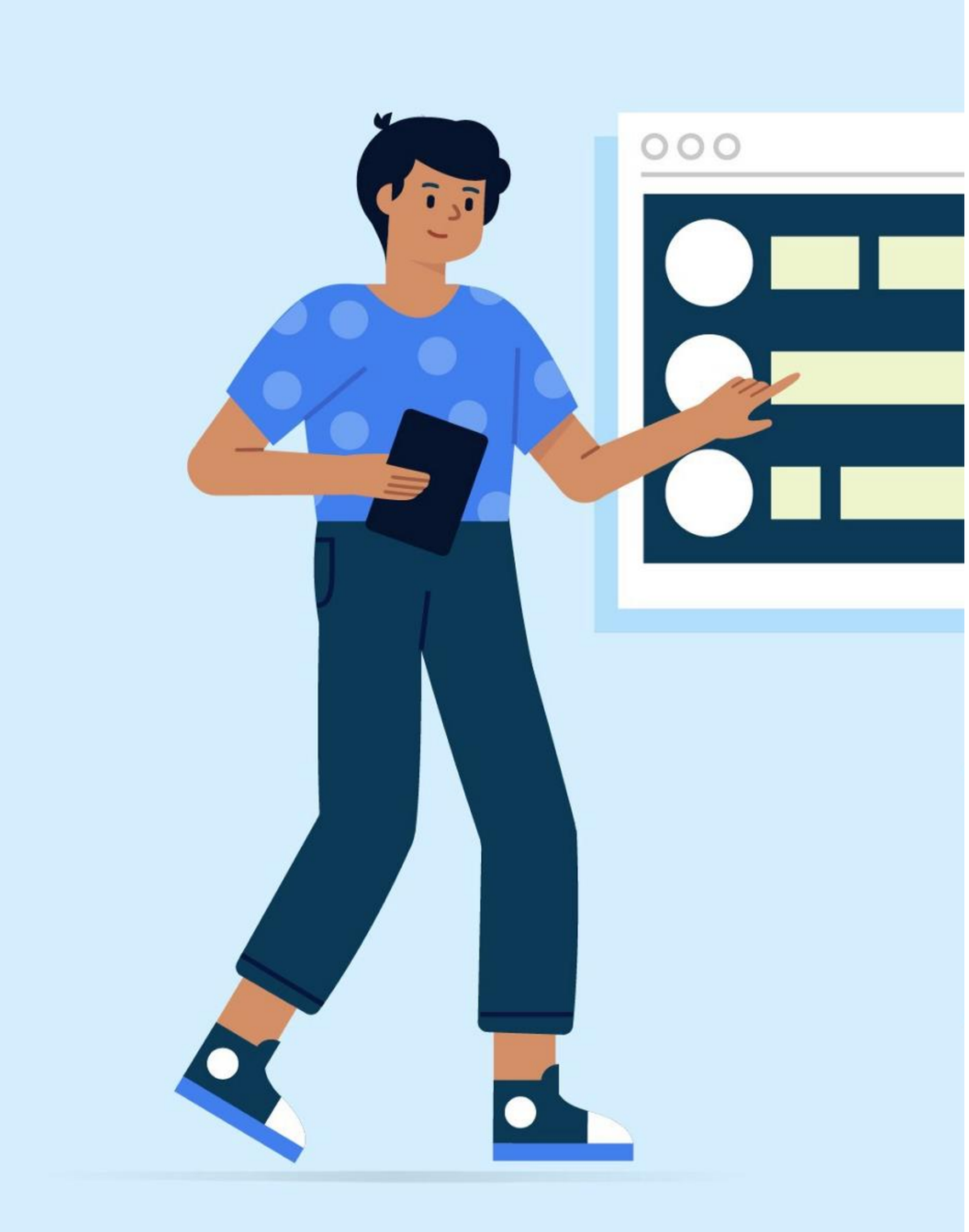
A exceção pode carregar dados quaisquer.

Tratamento Exceções

Nas próximas aulas 18 e 20, serão apresentados **Estruturas de tratamento** como:

Estrutura Try-Catch.
Pilha de Chamadas de Métodos.
Bloco finally.
Exceções Personalizadas

vnt/school
powered by venturus



Comunidade VNT

Dica de hoje

Toda classe em Java é uma subclasse da classe **Object.**
Object possui os seguintes **métodos**:

getClass - retorna o tipo do objeto
equals - compara se o objeto é igual a outro
hashCode - retorna um código hash do objeto
toString - converte o objeto para string

Pesquisa um pouco mais sobre!!
Boa pesquisa!!

Referências

[1] A. Goldman, F. Kon, Paulo J. S. Silva; Introdução à Ciência da Computação com Java e Orientação a Objetos (USP). 2006. Ed. USP.
[2] Algoritmo e lógica de programação. Acessado julho/2022: https://visualg3.com.br/
[3] G. Silveira; Algoritmos em Java; Ed. Casa do Código.
[4] M. T. Goodrich, R. Tamassia; Estrutura de dados e algoritmos em Java. Ed Bookman. 2007.
[5] Algoritmo e lógica de programação. Acessado julho/2022: https://www.cursoemvideo.com/
[6] P. Silveira, R. Turini; Java 8 Pratico: lambdas, streams e os novos recursos da linguagem. Ed. Casa do Código.
[7] Linguagem Java: Curso acessado em agosto/2022: https://www.udemy.com/
[8] Linguagem Java: Curso acessado em setembro/2022: https://www.cursoemvideo.com/