



PROGRAMAÇÃO ORIENTADA A OBJETOS

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br

<https://github.com/josenalde/apds>

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

OBJETIVOS

O(a) discente compreenderá o paradigma de programação orientada a objetos e como utilizá-lo para organizar soluções de software modularizado, a partir da modelagem em diagramas de classes

CONTEÚDO – 60H (4 CR)

Conceitos básicos do paradigma de orientação a objetos; Classes e objetos: estado, comportamento e identidade; Atributos, métodos e construtores; Pilares da orientação a objetos: encapsulamento e modificadores de acesso, herança, polimorfismo, classes abstratas e interfaces; Tipos enumerados; Agrupamento de classes (vetores, listas, conjuntos, etc.); Biblioteca de objetos; Introdução ao diagrama de classes UML: composição, agregação e herança; Tratamento de Exceções; Fluxo de dados, InputStream e outputStream; Projeto e implementação de uma aplicação OO (API de interface gráfica e conexão com banco de dados – Model View Controller).

PLANO DE CURSO

- Referências

[1] DEITEL, P.; DEITEL, H.; **Java como programar**, 10ª edição. Pearson, 2016.

[2] SIERRA, K.; BATES, B.; GEE, T. **Java: guia do aprendiz para programação no mundo real**, 3. ed. Série Use a Cabeça. O'Reilly (Alta Books), 2024.

[3] Material didático curso Informática para Internet@IMD: **Programação Orientada a Objetos**. Disponível em <<https://materialpublic.imd.ufrn.br/curso/disciplina/3/8>>.

[4] SHVETS, A. **Mergulho nos Padrões de Projeto**, 2022. Disponível (\$) em: <refactoring-guru/pt-br/design-patterns/book>

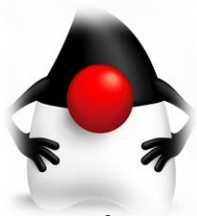
[5] RANGEL, P.; CARVALHO Jr., J.G. **Sistemas Orientados a Objetos: teoria e prática com UML e Java**. Rio de Janeiro: Brasport, 2021.

PLANO DE CURSO

- Software usado nas aulas
 - **Visual Studio Code** com extensão EXTENSION PACK FOR JAVA (Microsoft)
 - *Ctrl + Shift + P (Java: Create Java Project...) – Inicialmente com No Build Tools*
 - **Java Development Kit** ([ORACLE](#) ou [OpenJDK](#))
 - *Configurar variável de ambiente JAVA_HOME (caso não seja config. automaticamente durante instalação)*
 - *Outras IDEs de uso profissional em Java (Eclipse, Netbeans, IntelliJ, ...)*



OpenJDK



Duke

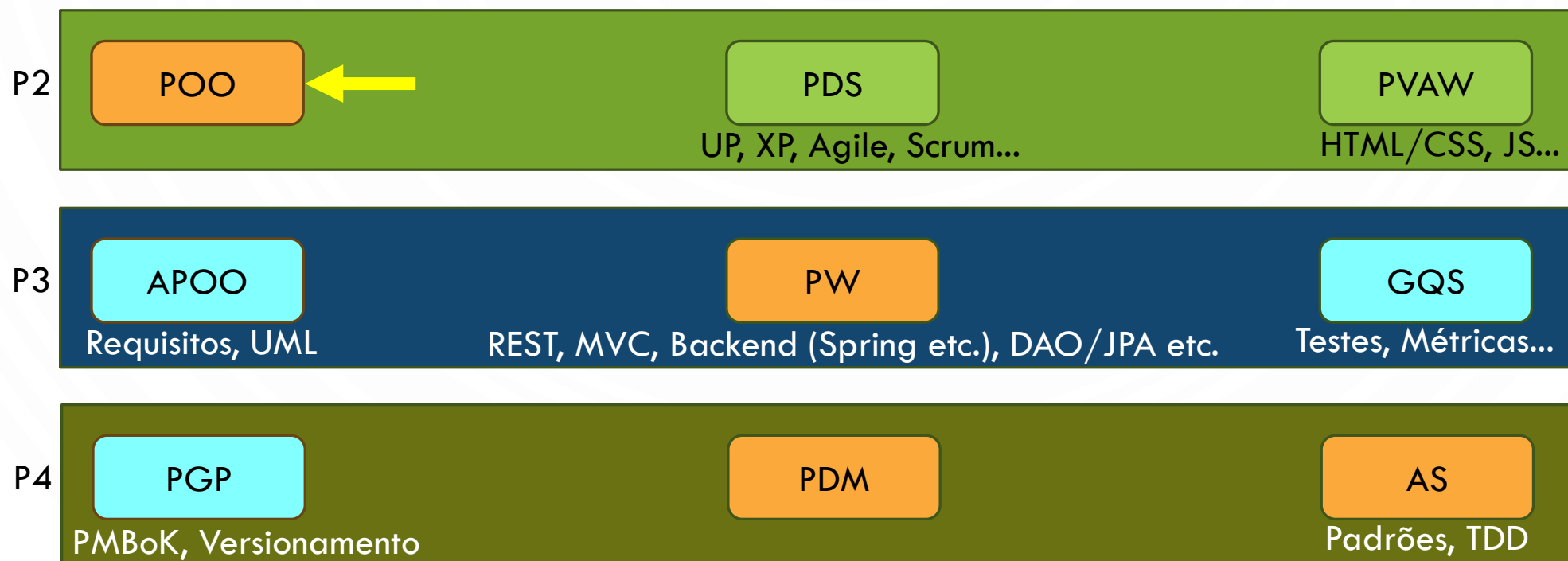
OBS: o site REPLIT.COM mudou sua política e agora cobra a partir de N repls e algumas linguagens (Java) já pede cartão de crédito direto....

PLANO DE CURSO

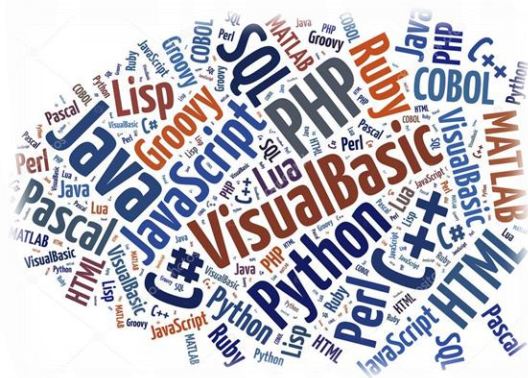
- Avaliações (Semestre 2024.2: 16.set.24 – 01.fev.25)
 - *Feriados nas aulas de POO de 03.out (estadual) e 20.nov (nacional)*
 - Unidade I: 24.out.2024
 - Unidade II: 05.dez.2024 (*Suspensão das atividades acadêmicas de 23/12-03/01*)
 - *Retorno às aulas de POO em 2025 dia 08.jan*
 - Unidade III: 23.jan.2025
 - Prova reposição: 29.jan.2025 (FIM)

ONDE ESTAMOS NO CURSO?

- A importância de Algoritmos: dependem TAD0009 - PROGRAMACAO ORIENTADA A OBJETOS 2. nível; TAD0018 - PROCESSAMENTO DIGITAL DE IMAGENS 4. nível; TAD0020 - ESTRUTURAS DE DADOS 3. nível; TAD0025 - INTELIGÊNCIA COMPUTACIONAL 3. nível; TAD0065 - APRENDIZADO DE MAQUINA (optativa)
- Dependem de POO: TAD0019 - PROGRAMACAO WEB 3. nível; TAD0027 - PROGRAMACAO PARA DISPOSITIVOS MOVEIS 4. nível; TAD0029 - ARQUITETURA DE SOFTWARE 4. nível













CONCEITOS “NA PRÁTICA...” - LINGUAGENS



- Python, Javascript e C++ podem incluir conceitos OO
- **Java**, C# são intrinsecamente OO: C# é uma das linguagens da plataforma .NET, com o código fonte compilado em arquivo CIL (Common Intermediate Language) que é executado na máquina virtual CLR (Common Language Runtime) por meio de compilador Just In Time (JIT)

<https://www.tiobe.com/tiobe-index/>

Sep 2024	Sep 2023	Change	Programming Language		Ratings	Change
1	1			Python	20.17%	+6.01%
2	3	▲		C++	10.75%	+0.09%
3	4	▲		Java	9.45%	-0.04%
4	2	▼		C	8.89%	-2.38%
5	5			C#	6.08%	-1.22%
6	6			JavaScript	3.92%	+0.62%
7	7			Visual Basic	2.70%	+0.48%
8	12	▲▲		Go	2.35%	+1.16%
9	10	▲		SQL	1.94%	+0.50%
10	11	▲		Fortran	1.78%	+0.49%

Ranking 2023 – GitHub: Javascript, Python, Typescript, **Java**, Shellscript

COMPARAÇÃO...

ChatGPT: "Crie uma classe **Carro** com atributos cor, motor, ano modelo, fabricante, modelo, e mostre em Python, Java, C++, Javascript"


python

```
class Carro:
    def __init__(self, cor, motor, ano_modelo, fabricante, modelo):
        self.cor = cor
        self.motor = motor
        self.ano_modelo = ano_modelo
        self.fabricante = fabricante
        self.modelo = modelo

    def exibir_detalhes(self):
        print(f"Cor: {self.cor}")
        print(f"Motor: {self.motor}")
        print(f"Ano do Modelo: {self.ano_modelo}")
        print(f"Fabricante: {self.fabricante}")
        print(f"Modelo: {self.modelo}")

# Exemplo de uso
carro = Carro("Vermelho", "1.6", 2023, "Toyota", "Corolla")
carro.exibir_detalhes()
```

java

 Copiar código

```
public class Carro {
    private String cor;
    private String motor;
    private int anoModelo;
    private String fabricante;
    private String modelo;

    public Carro(String cor, String motor, int anoModelo, String fabricante, String modelo) {
        this.cor = cor;
        this.motor = motor;
        this.anoModelo = anoModelo;
        this.fabricante = fabricante;
        this.modelo = modelo;
    }

    public void exibirDetalhes() {
        System.out.println("Cor: " + this.cor);
        System.out.println("Motor: " + this.motor);
        System.out.println("Ano do Modelo: " + this.anoModelo);
        System.out.println("Fabricante: " + this.fabricante);
        System.out.println("Modelo: " + this.modelo);
    }

    public static void main(String[] args) {
        Carro carro = new Carro("Vermelho", "1.6", 2023, "Toyota", "Corolla");
        carro.exibirDetalhes();
    }
}
```


COMPARAÇÃO...

```
#include <iostream>
#include <string>

class Carro {
private:
    std::string cor;
    std::string motor;
    int anoModelo;
    std::string fabricante;
    std::string modelo;

public:
    Carro(std::string c, std::string m, int ano, std::string fab, std::string mod)
        : cor(c), motor(m), anoModelo(ano), fabricante(fab), modelo(mod) {}

    void exibirDetalhes() {
        std::cout << "Cor: " << cor << std::endl;
        std::cout << "Motor: " << motor << std::endl;
        std::cout << "Ano do Modelo: " << anoModelo << std::endl;
        std::cout << "Fabricante: " << fabricante << std::endl;
        std::cout << "Modelo: " << modelo << std::endl;
    }
};

int main() {
    Carro carro("Vermelho", "1.6", 2023, "Toyota", "Corolla");
    carro.exibirDetalhes();
}
```

javascript

Copiar código

```
class Carro {
    constructor(cor, motor, anoModelo, fabricante, modelo) {
        this.cor = cor;
        this.motor = motor;
        this.anoModelo = anoModelo;
        this.fabricante = fabricante;
        this.modelo = modelo;
    }

    exibirDetalhes() {
        console.log(`Cor: ${this.cor}`);
        console.log(`Motor: ${this.motor}`);
        console.log(`Ano do Modelo: ${this.anoModelo}`);
        console.log(`Fabricante: ${this.fabricante}`);
        console.log(`Modelo: ${this.modelo}`);
    }
}

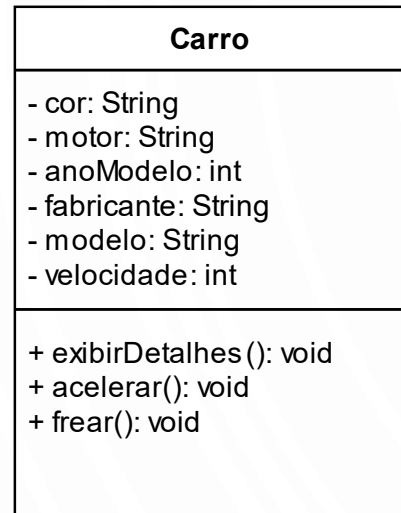
// Exemplo de uso
const carro = new Carro("Vermelho", "1.6", 2023, "Toyota", "Corolla");
carro.exibirDetalhes();
```

TEMOS ENTÃO NOSSO PRIMEIRO CONCEITO OO:

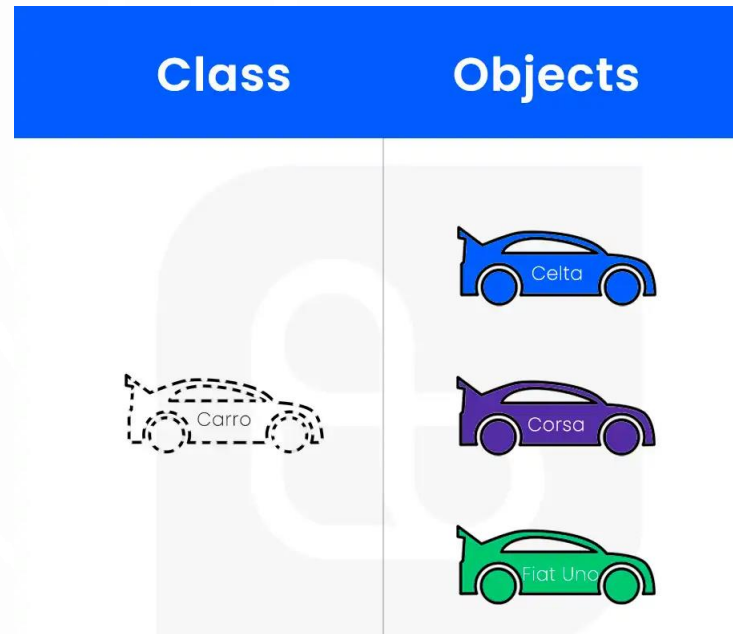
ABSTRAÇÃO

- Link entre real x virtual, concreto x abstrato – como posso representar algo, como posso descrever? Considere uma entidade, ou melhor, um **objeto**
 - Exemplos: bola, carro, camisa, gato, cachorro, relógio, pessoa, conta bancária, poema, smartphone, monitor, discente, docente, componente curricular, figura geométrica etc.
- Podemos pensar em termos de **atributos** (características) – **lembrem dos membros de uma estrutura em C++** - e ações que este pode realizar ou sofrer (**métodos**)
 - Temos então um “modelo” deste objeto, que pode ter descrições distintas, a depender do contexto específico de negócio
 - Exemplo: Carro numa locadora de veículos (fabricante, marca, ano, tipo câmbio, n. portas, placa); Carro no DETRAN (placa, chassi, potência, cor, PROPRIETÁRIO, DÉBITOS, etc.)
 - Exemplo: Avião num simulador de voo (velocidade, altitude, ângulos de rolagem, inclinação vertical, guinada; Avião num sistema de reservas de passagens (assentos)

REPRESENTANDO O MODELO – DIAGRAMA DE CLASSES (UML)



by drawio (do autor)



Objeto = operações + dados

Fonte: <https://www.dio.me/articles/os-fundamentos-da-poo-em-java-entenda-classes-objetos-e-metodos>

- Imagine **Carro** como um tipo abstrato de dados criado pelo usuário (desenvolvedor)
- Podemos então criar vários “carros” que possuem estes dados/características comuns (**atributos**) e operações/funcionalidades (métodos), que definem seu comportamento
- Cada **instância** da classe Carro é denominada **Objeto** da classe Carro

ESTADO E COMPORTAMENTO

Carro
- cor: String - motor: String - anoModelo: int - fabricante: String - modelo: String - velocidade: int
+ exibirDetalhes(): void + acelerar(): void + frear(): void

➔ Identidade do objeto – como o referenciamos

carroA : {"azul", 1.6, 2017, "Fiat", "Toro", 80}

carroA : {"azul", 1.6, 2017, "Fiat", "Toro", 100}

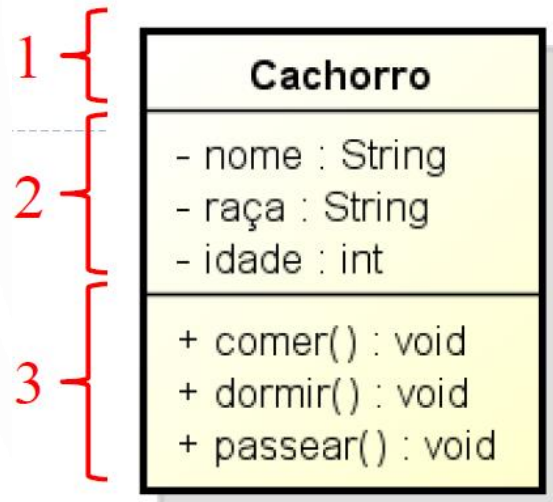
carroB : {"prata", 1.0, 2023, "Fiat", "Mobi", 90}

carroC : {"branca", 1.3, 2020, "Toyota", "Etios", 110}

Em Java, sua "criação" (instanciação) básica é Carro **carroA** = new Carro();

- O conjunto de atributos de um objeto em determinado momento define seu **ESTADO**
- O conjunto de métodos define o seu **COMPORTAMENTO**. Então o **ESTADO** representa o resultado cumulativo do seu comportamento em determinado momento.

ESTADO E COMPORTAMENTO: EXEMPLOS



Instância 1 (objeto)

Instância 2 (objeto)

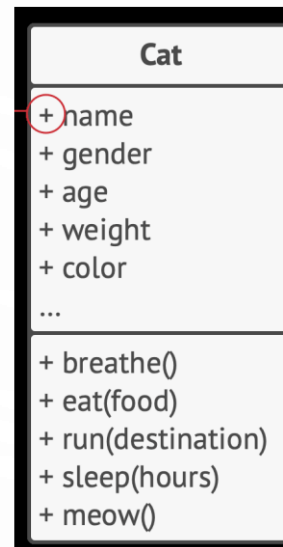


“chaveiro”
“dachshund”
5 meses



“lobo”
“husky”
26 meses

1. Nome da Classe (por padrão primeira letra maiúscula)
2. Atributos (alguns livros chamam campos da classe)
3. Métodos



Tom: Cat

name = "Tom"
sex = "macho"
age = 3
weight = 7
color = marrom
texture = listrada



Nina: Cat

name = "Nina"
sex = "fêmea"
age = 2
weight = 5
color = cinza
texture = lisa

Fonte: [4]

HANDS-ON (PRÁTICA 1)



Caneta
<ul style="list-style-type: none">- modelo: String- cor: String- ponta: float- carga: int- tampada: boolean
<ul style="list-style-type: none">+ status(): void+ escrever(): void+ tampar(): void+ destampar(): void+ isTampada(): boolean

1. Criar Java Project (no build tools) de nome **caneta**
2. Criar classe Caneta.java
3. Escrever a lógica para cada método
4. Criar classe principal com método main() para instanciar caneta
 - setar atributos modelo: "compactor", cor : azul, ponta: 0.7, carga: 10ml
 - tampar a caneta
 - tentar escrever
 - ver status
 - destampar
 - tentar escrever
5. Criar mais duas canetas, sendo agora uma BIC preta e uma Faber Castell Vermelha com os demais atributos a sua escolha

HANDS-ON (PRÁTICA 2)

Crie a classe **Elevador** para armazenar as informações de um elevador dentro de um prédio. A classe deve armazenar o andar atual (térreo = 0), total de andares no prédio (desconsiderando o térreo), capacidade do elevador e quantas pessoas estão presentes nele. A classe deve também disponibilizar os seguintes métodos:

- ☐ Inicializa: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazios);
- ☐ Entra: acrescenta uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
- ☐ Sai: remove uma pessoa do elevador (só deve remover se houver alguém dentro dele);

1. Esboce o diagrama UML para esta classe
2. Crie um método na classe que apresente o status do elevador (andar atual e pessoas presentes/capacidade total)
3. Cria uma classe Principal que instancie um Elevador
4. Apresente um menu na tela para o usuário **escolher o método desejado**



DICA: TIPAGEM EM JAVA E LEITURA DO CONSOLE

TIPOS PRIMITIVOS E CLASSES WRAPPERS (EMPACOTADORAS)

- Classes que encapsulam tipos primitivos existentes e fornecem operações sobre estes
- Permite que tipos primitivos sejam tratados como objetos

Grupo numérico INTEIRO: byte (Byte), short (Short), int (Integer), long (Long)

Grupo numérico PONTO FLUTUANTE: float (Float), double (Double)

Grupo CHARACTER: char (Char)

Grupo BOOLEANO: boolean (Boolean)

Não necessita ser instanciada (removido nas versões atuais Java):

```
Integer x = 1;
```

```
Float y = 4f;
```


TIPOS PRIMITIVOS E CLASSES WRAPPERS (EMPACOTADORAS)

- Checagens numéricas (isNaN, isInfinite, compareTo, etc.)

```
Integer a = 10;  
Integer b = 20;  
// 10<20, saída -1  
System.out.println(a.compareTo(b));
```

- Conversões (parseInt, parseFloat, valueOf, intValue, toString, etc.)
- Tipo String não considerada classe Wrapper pois não existe tipo primitivo string em Java, no entanto possui vários construtores sobrecarregados, métodos de conversão de tipos e operações para lidar com caracteres e intervalos de dados (substrings, buscas etc.)

```
length() // comprimento da String  
charAt(int index) // caractere na posição específica da String  
concat(String str) // concatena duas strings (+)  
equals (Object anObject) // compara se duas strings são iguais  
valueOf(int i) // retorna String com base no valor de um inteiro  
valueOf(float f)  
valueOf(double d)  
valueOf(boolean b)  
lastIndexOf(String str) // retorna o último índice da String  
isEmpty() // retorna TRUE se possui tamanho 0  
isBlank() // retorna verdadeiro se só possui espaços em branco  
substring(int start[, int end]) // retorna String a partir do start até end  
trim() // remove espaços em branco ao redor da string  
split() // divide string com base em separador ou REGEX  
indexOf(char c) // índice de determinado caracter
```

CLASSE SCANNER

- Facilita o uso de métodos sobre o InputStream

- next() (até encontrar caractere em branco)
- nextLine() (até encontrar \n)
- nextInt()
- nextFloat()
- nextDouble()
- nextBoolean()
- hasNextInt(), hasNextFloat(), hasNextDouble()... – verifica se próximo é...

```
Scanner cin = new Scanner(System.in);
String nome = cin.nextLine();
if (cin.hasNextInt()) Integer v1 = cin.nextInt();
if (cin.hasNextFloat()) Float v2 = cin.nextFloat();
if (cin.hasNextDouble()) Double v3 = cin.nextDouble();
if (cin.hasNextBoolean()) Boolean v4 = cin.nextBoolean();
```