UNIRIO

Técnicas de Programação II

BSI — 2018.2

Jefferson Elbert Simões

CCET/DIA

14 de agosto de 2018

Sobre o curso

Professor Jefferson
Sala CCET 115
E-mail jefferson.simoes@uniriotec.br

Todo o material do curso estará disponível do Moodle

- moodleccet.uniriotec.br

Horário 3as e 5as, 15h–18h
Sala CCET Lab 2

TP2 é sobre o que?

Quantas linguagens de programação existem?

TP2 é sobre o que?

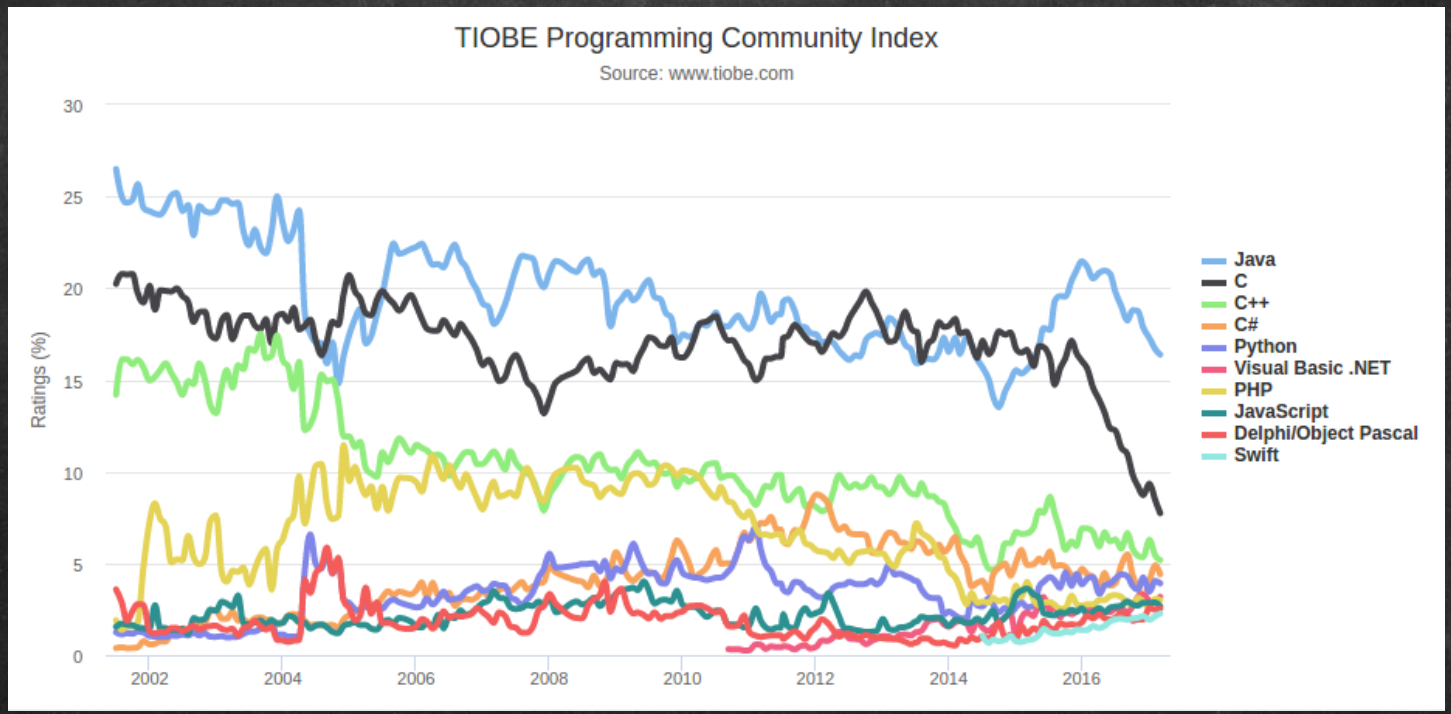
Quantas linguagens de programação existem?

Java	C	C++	C#	Python
VB .NET	PHP	JavaScript	Object Pascal	Swift
Perl	Ruby	Assembly	R	Visual Basic
Objective-C	Go	MATLAB	PL/SQL	Scratch
SAS	D	Dart	ABAP	COBOL
Ada	Fortran	Transact-SQL	Lua	Scala
Logo	F#	Lisp	LabVIEW	Prolog
Haskell	Scheme	Groovy	RPG (OS/400)	Apex
Erlang	MQL4	Rust	Bash	Ladder Logic
Q	Julia	Alice	VHDL	Awk

50 linguagens de programação mais utilizadas. Fonte: TIOBE Index

TP2 é sobre o que?

Quantas linguagens de programação existem?



Evolução de uso das top 10 LP's. Fonte: TIOBE Index

TP2 é sobre o que?

Quantas linguagens de programação existem?

Java	C	C++	C#	Python
VB .NET	PHP	JavaScript	Object Pascal	Swift
Perl	Ruby	Assembly	R	Visual Basic
Objective-C	Go	MATLAB	PL/SQL	Scratch
SAS	D	Dart	ABAP	COBOL
Ada	Fortran	Transact-SQL	Lua	Scala
Logo	F#	Lisp	LabVIEW	Prolog
Haskell	Scheme	Groovy	RPG (OS/400)	Apex
Erlang	MQL4	Rust	Bash	Ladder Logic
Q	Julia	Alice	VHDL	Awk

50 linguagens de programação mais utilizadas. Fonte: TIOBE Index

TP2 é sobre o que?

Quantas linguagens de programação existem?

Java	C	C++	C#	Python
VB .NET	PHP	JavaScript	Object Pascal	Swift
Perl	Ruby	Assembly	R	Visual Basic
Objective-C	Go	MATLAB	PL/SQL	Scratch
SAS	D	Dart	ABAP	COBOL
Ada	Fortran	Transact-SQL	Lua	Scala
Logo	F#	Lisp	LabVIEW	Prolog
Haskell	Scheme	Groovy	RPG (OS/400)	Apex
Erlang	MQL4	Rust	Bash	Ladder Logic
Q	Julia	Alice	VHDL	Awk

O que estas linguagens têm em comum?

TP2 é sobre o que?

Quantas linguagens de programação existem?

Java	C	C++	C#	Python
VB .NET	PHP	JavaScript	Object Pascal	Swift
Perl	Ruby	Assembly	R	Visual Basic
Objective-C	Go	MATLAB	PL/SQL	Scratch
SAS	D	Dart	ABAP	COBOL
Ada	Fortran	Transact-SQL	Lua	Scala
Logo	F#	Lisp	LabVIEW	Prolog
Haskell	Scheme	Groovy	RPG (OS/400)	Apex
Erlang	MQL4	Rust	Bash	Ladder Logic
Q	Julia	Alice	VHDL	Awk

O que estas linguagens têm em comum?

Linguagens orientadas a objeto!

Paradigmas

Paradigmas: ditam a "forma de pensar" de programas

- Determinam diversos aspectos do projeto da linguagem
- Fornecem construções e estratégias mais adequadas para determinados problemas ou tarefas

- Relevantes para pessoas, não máquinas

Paradigmas

Paradigmas: ditam a "forma de pensar" de programas
Exemplos:

- Imperativo — Fortran
- Procedural — C, PHP

- Orientado a objetos — Java, C++, Python

- Orientado a eventos — JavaScript
- Declarativo — SQL
- Funcional — Haskell

Paradigmas

Paradigmas: ditam a "forma de pensar" de programas
Exemplos:

- Imperativo — Fortran
- Procedural — C, PHP
  - Programas são sequências de passos organizados em procedimentos (rotinas, funções)
- Orientado a objetos — Java, C++, Python
  - Programas são compostos de objetos que interagem entre si
- Orientado a eventos — JavaScript
- Declarativo — SQL
- Funcional — Haskell

Ementa e conteúdo

Conteúdo do curso: Java

- Revisão de TP I
- Estruturas de dados
- Algoritmos recursivos
- Introdução à orientação a objetos
- Construtores e Destrutores
- Sobrecarga
- Derivação de Classes
- Polimorfismo

Ementa e conteúdo

Conteúdo do curso: Java

- Revisão de TP I
- Estruturas de dados
  - Listas
  - Filas
  - Pilhas
- Algoritmos recursivos
- Introdução à orientação a objetos
- Construtores e Destrutores
- Sobrecarga
- Derivação de Classes
- Polimorfismo

Ementa e conteúdo

Conteúdo do curso: Java

- Revisão de TP I
- Estruturas de dados
- Algoritmos recursivos
  - Módulos recursivos
  - Registros de ativação de procedimentos e funções
  - Algoritmos recursivos. Funções matemáticas recursivas
- Introdução à orientação a objetos
- Construtores e Destrutores
- Sobrecarga
- Derivação de Classes
- Polimorfismo

Ementa e conteúdo

Conteúdo do curso: Java

- Revisão de TP I
- Estruturas de dados
- Algoritmos recursivos
- Introdução à orientação a objetos
  - Conceitos de orientação a objetos
  - Classes e objetos
  - Atributos e métodos
- Construtores e Destrutores
- Sobrecarga
- Derivação de Classes
- Polimorfismo

Ementa e conteúdo

Conteúdo do curso: Java

- Revisão de TP I
- Estruturas de dados
- Algoritmos recursivos
- Introdução à orientação a objetos
- Construtores e Destrutores
  - Construtores
  - Armazenamento dinâmico
  - Argumentos default
  - Destrutores
- Sobrecarga
- Derivação de Classes
- Polimorfismo

Ementa e conteúdo

Conteúdo do curso: Java

- Revisão de TP I
- Estruturas de dados
- Algoritmos recursivos
- Introdução à orientação a objetos
- Construtores e Destrutores
- Sobrecarga
  - Sobrecarga de funções
  - Sobrecarga de operadores
  - Conversão entre tipos
- Derivação de Classes
- Polimorfismo

Ementa e conteúdo

Conteúdo do curso: Java

- Revisão de TP I
- Estruturas de dados
- Algoritmos recursivos
- Introdução à orientação a objetos
- Construtores e Destrutores
- Sobrecarga
- Derivação de Classes
  - Derivação de uma classe
  - Funções virtuais
  - Herança múltipla
  - Classes virtuais
- Polimorfismo

Bibliografia

- Kathy Sierra e Bert Bates. Use a Cabeça Java, 2ª edição, Alta Books, 2005.
- Cay Horstmann e Gary Cornell. Core Java, Vol. I — Fundamentos, 8ª Edição, Pearson Education, 2010.
- Herbert Schildt. Java — A Referência Completa, 8ª Edição traduzida, Alta Books, 2014

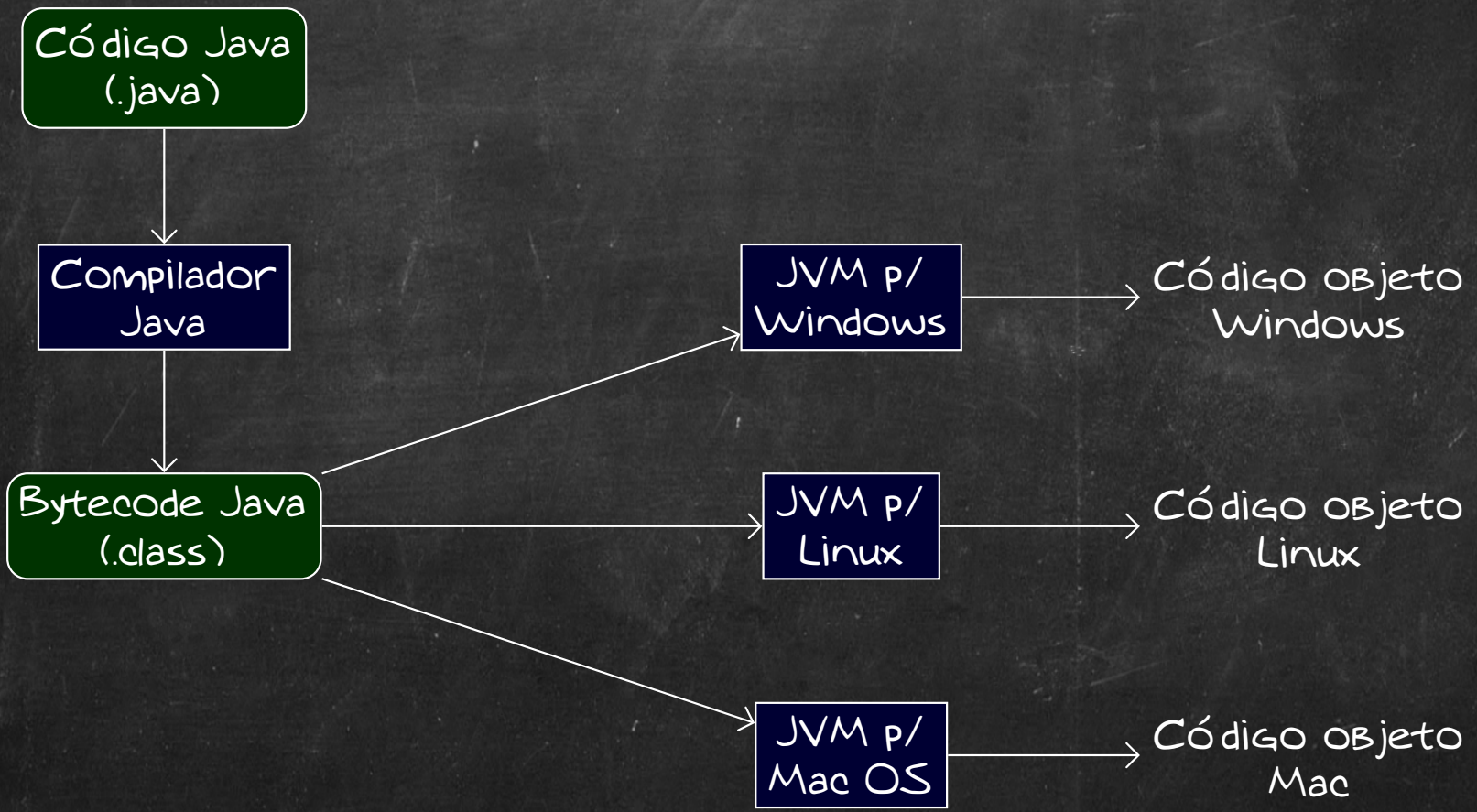
O que é Java?

- Linguagem de programação proprietária
  - Criada pela Sun, visando sistemas de entretenimento
  - Propriedade da Oracle
- Focada em programação orientada a objetos
  - Suporte a outros paradigmas (ex: estruturado, concorrente)

Características

- Linguagem compilada e interpretada
- Código Java passa por duas etapas:
  - **Compilador** converte código original para Byte code Java
    - Independente de plataforma
    - JDK: Java Development Kit
  - **JVM** (Java Virtual Machine) interpreta o Byte code
    - Dependente de plataforma
    - JRE: Java Runtime Environment

Características



Características

- Portabilidade
  - Bytecode pode ser executado em qualquer máquina que possua uma implementação da JVM
- Segurança
  - JVM varre Bytecode em Busca de Brechas de segurança antes de interpretá-lo

Características

- Gerência de memória
  - Programador não gerência memória diretamente: Java fornece um coletor de lixo (garbage collector)
- Reutilização de código
  - Diversas bibliotecas padrão: `java.{util,io,math,net,swing,...}`
- Multithreading
- Tratamento de exceções
- Aplicações distribuídas
  - Conexões a SGBDs, objetos remotos, ...

Variáveis

Elementos básicos de dados. Toda variável possui:

- Nome
- Tipo de dado
- Valor

Exemplo:

```
int a = 10;
```

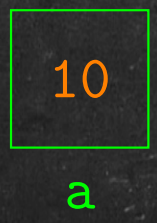
Variáveis

Elementos básicos de dados. Toda variável possui:

- Nome
- Tipo de dado
- Valor

Exemplo:

```
int a = 10;
```



Tipos primitivos

`boolean`	Booleano, valores `True` ou `False`
`byte`	inteiro de 8 bits, valores entre -128 e 128
`short`	inteiro de 16 bits, valores entre -2^{15} e 2^{15}
`int`	inteiro de 32 bits, valores entre -2^{31} e 2^{31}
`long`	inteiro de 64 bits, valores entre -2^{63} e 2^{63}
`float`	ponto flutuante de 32 bits (8+23): valores entre $\sim 1.40 \cdot 10^{-45}$ e $\sim 3.40 \cdot 10^{38}$ (+ negativos)
`double`	ponto flutuante de 64 bits (11+52): valores entre $\sim 4.9 \cdot 10^{-324}$ e $\sim 1.8 \cdot 10^{308}$ (+ negativos)

Operadores

Funções simplificadas operando sobre variáveis numéricas

`a = b`	atribuição: armazena em `a` o valor de `b`
`a + b`	retorna a soma dos valores de `a` e `b` outros: `a - b` , `a * b` , `a / b`
`a += b`	equivalente a `a = a + b` outros: `a -= b` , `a *= b` , `a /= b`
`a++`	equivalente a `a = a + 1` outros: `a--`
`a == b`	retorna `True` se os valores de `a` e `b` são iguais, c.c. `False` outros: `a != b` , `a < b` , `a <= b` , `a > b` , `a >= b`
`a && b`	retorna o E binário dos valores das variáveis `a` e `b` outros: `a || b` (OU binário), `!a` (NOT binário)

Condicionais

Selecionam blocos de código de acordo com valores de uma expressão lógica.

Exemplo:

```
float temperatura = 35.0;
boolean comFebre;
if (temperatura > 37.5)
{
    comFebre = True;
}
else
{
    comFebre = False;
}
```

Laços de repetição

Repetem um determinado bloco de código enquanto uma determinada condição for verdadeira.

Exemplo:

```
int regressiva = 10;
while( regressiva > 0 )
{
    System.out.println( regressiva );
    regressiva--;
}
```

Laços de repetição

Três tipos de laços:

```
int max = 5;
int soma = 0;
while( max > 0 )
{
    soma += max;
    max--;
}
```

```
int max = 5;
int soma = 0;
do
{
    soma += max;
    max--;
} while( max > 0 );
```

```
int soma = 0;
for( max = 5; max > 0; max++ )
{
    soma += max;
}
```

Arrays e listas

Estruturas para armazenar coleções de objetos

Arrays: estrutura mais simples — índices inteiros, tamanho fixo

```
String [] nomes = new String[16];
nomes[0] = "TP2";
```

ArrayList: mais sofisticado — tamanho variável, métodos de busca

```
ArrayList<String> nomes = new ArrayList<>();
nomes.add("TP2");
```

Funções

Rotinas autocontidas que realizam uma tarefa específica

Exemplo:

```
public static double celsius_fahrenheit(double cels) {
    double fahr;
    fahr = 1.8 * cels + 32;
    return fahr;
}
```

Classes

(Bloco fundamental do paradigma OO de programação.)
Neste momento, classes serão conjuntos de funções com contexto semelhante.

Um programa pode ter diversas classes. Uma, e apenas uma classe, deve conter uma função com o seguinte protótipo:

```
public static void main(String [] args)
```

- Ponto de entrada do programa

Entrada e Saída

Fornecidos pela classe `System`:

- `System.out`: saída padrão (terminal)
- `System.in`: entrada padrão (terminal)
- Funções úteis: `print`, `println`, `read`
- Classe `Scanner`: diretivas para leitura do terminal

Funções Matemáticas

Fornecidos pela classe `Math`. Funções úteis:

- `sqrt`: raiz quadrada
- `pow`: potenciação
- `log`: logaritmo
- `round`: arredondamento de decimais

Logística do curso

- A presença em sala é obrigatória!
  - Chamada duas vezes por aula
  - Frequência mínima: 75% das aulas (máximo de ≈ 5 faltas)
- Nosso curso terá duas etapas distintas:
  - 1ª etapa: conceitos de orientação a objetos
    - Aulas teóricas
    - Exercícios de implementação simples
    - Duração prevista: até início de outubro
  - 2ª etapa: trabalho de implementação
    - Grupos de ≈ 3 pessoas
    - Desenvolvimento de um software utilizando as técnicas aprendidas
    - Duração prevista: entre meio de outubro e início de dezembro

Avaliações (previstas)

- 1 avaliação escrita
  - entre 1ª e 2ª etapas do curso
- Exercícios de fixação e de implementação
  - Individuais, entrega rápida (em geral, até a aula seguinte)
- Trabalho final em grupo
- PF e segunda chamada
  - PF condicionada a presença em 75% das aulas (máximo de ≈ 5 faltas)
- Listas de exercício
  - Pontuação extra, entrega até o final do período

Dicas

- Não faltem!
- Não percam os prazos!
  - Toda atividade do curso vale ponto
- Troquem ideias uns com os outros
- Utilizem o apoio dos monitores
  - É mais fácil aprender se não estiver sozinho
- Pratiquem, pratiquem, pratiquem

Dúvidas? Perguntas?

Exercícios

Escreva funções para realizar as seguintes tarefas:

1. Determinar se um inteiro é primo;
2. Determinar se 3 inteiros formam um triângulo;
3. Contar inteiros pares em uma lista de inteiros;
4. Verificar se uma palavra é um palíndromo[1];
5. Calcular média e desvio padrão de uma lista de números
   - Para uma sequência $x_1, \ldots, x_n$, a média é dada por $\mu = \frac{\sum_{i=1}^{n} x_i}{n}$ e o desvio padrão por $\sigma = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n-1}$
6. (Desafio) Ordenar um array de números inteiros;
   - Dica: utilize o algoritmo Bubble Sort

[1] Não utilizar nenhuma função da classe `String` — exceto `charAt()`

Exercícios

Utilizando as funções que você criou no slide anterior, escreva um único programa que:

1. Determina se o número 180.421.597 é primo;
2. Determinar se os números 348, 561 e 937 formam um triângulo;
3. Verifica se a palavra "ATTACGGAGGTCCATACT TGGTTCATCCTGGAGGCATTA" (sem espaços) é um palíndromo;
4. Conta a quantidade de pares na lista {169, 4, 100, 64, 9, 16, 225, 121, 81, 196, 25, 36, 144, 493}.
5. Calcula a média e o desvio padrão da lista anterior;
6. (Desafio) Ordena a lista anterior.

Para próxima aula

1. Criem uma conta em uva.onlinejudge.org
   - Exercícios e desafios de maratonas de programação
   - Iremos fazer exercícios juntos na próxima aula

Créditos

- Baseado em slides de Gleison Santos