Programa do Minicurso

1 Identificação

Disciplina: Minicurso De Java Moderno

Carga Horária: 20 horas-aula Teóricas: 10 Práticas: 10

Período: Agosto a Novembro de 2018.

2 Cursos (público alvo)

Ciência da Computação;

Sistemas de Informação;

Interessados na linguagem Java.

3 Requisitos

Nenhum.

4 Ementa

Características principais da linguagem; *Bytecode* e Processo de compilação; Introdução ao paradigma imperativo; Tipos primitivos e *String*; Inferência de tipo; Estrutura condicional `if`. Laço de repetição *for*; *Raw-Arrays*; Tempo de vida de variáveis: Escopo; *For-each*. Funções e procedimentos; Sobrecarga (*overload*) de função; Busca linear; Complexidade de Algoritmos; Notação assintótica. Classe, Objeto e Atributo; Alocação Dinâmica; *Garbage-Collection*; Tempo de vida de objetos. *Message-Passing*; Cópia e Referência; Método Construtor; Boas práticas: API; Documentação de código. Estrutura de Dados: Pilha; Padrão de projeto: Composição; Programação Defensiva via Exceções e Asserções; Polimorfismo Estático via *Generics*; *Type-erasure*. Interfaces; *Lazy-Evaluation*; Métodos *default*; Classes abstratas; Herança e problema do diamante. Boas práticas: Manutenção; *Design-By-Contract*; *Early-Return*; Teoria de Tipos: Tipagem Estática; Teoria de Tipos: Tipagem Forte. *Callback*; API Funcional de Java; Classes Anônimas; Funções *lambda*.

Sistema de pacotes Java; *Classpath*; Makefiles; Arquivo JAR como biblioteca; Arquivo JAR executável. Otimizações de Arquitetura. Otimizações de Compiladores. Otimizações da JVM. Compilação *JIT* e *AOT*. *Tail-call recursion*. Complexidade amortizada.

5 Objetivos

Geral: Auxiliar na aprendizagem das disciplinas de Programação Orientada a Objetos 1 e 2, demonstrando os recursos da linguagem, boas práticas, reusabilidade, *tradeoffs* entre soluções (performance, uso de memória, manutenção...) e as novidades nas versões mais recentes..

Específico: São objetivos do minicurso:

- Apresentar os recursos básicos da linguagem Java, dentre eles:
  - Classes e método `main`;
  - Declaração/instanciação de variáveis;
  - Ciclo de vida de variáveis;
  - Referência e cópia;
  - Métodos e construtores.

  Além das características e funcionamento interno deles;
- Criar pequenos programas simples para mostrar o uso das estruturas básicas da programação imperativa (comparação, seleção, repetição, funções...);
- Melhorar o desempenho e entendimento dos alunos das disciplinas de Programação Orientada a Objetos 1 e 2;
- Instruir os participantes a respeito das boas práticas de programação, a fim de que façam códigos bem estruturados, reutilizáveis e legíveis, aplicando conceitos de código-limpo e modularidade;
- Introduzir noções de algoritmos e resolução de problemas computacionalmente;
- Expor formas de melhorar a manutenibilidade de software através de metodologias da programação moderna;
- Demonstrar o uso dos recursos recentes e progressão da linguagem Java;
- Demonstrar o funcionamento interno da JVM (*Java Virtual Machine*).

6 Conteúdo Programático

1. Introdução a Java (2 horas):
   (a) Características principais da linguagem;
   (b) *Bytecode* e Processo de compilação;
   (c) Introdução ao paradigma imperativo;
   (d) Tipos primitivos e *String*;
   (e) Inferência de tipo;
   (f) Estrutura condicional `if`.
2. Coleções (2 horas):
   (a) Laço de repetição *for*;
   (b) *Raw-Arrays*;
   (c) Tempo de vida de variáveis: Escopo;
   (d) *For-each*.
3. Introdução a Algoritmos (2 horas):
   (a) Funções e procedimentos;
   (b) Sobrecarga (*overload*) de função;
   (c) Busca linear;
   (d) Complexidade de Algoritmos;
   (e) Notação assintótica.
4. Tipos definidos por usuário (2 horas):
   (a) Classe, Objeto e Atributo;
   (b) Alocação Dinâmica;
   (c) *Garbage-Collection*;
   (d) Tempo de vida de objetos.
5. Métodos (2 horas):
   (a) *Message-Passing*;
   (b) Cópia e Referência;
   (c) Método Construtor;
   (d) Boas práticas: API;
   (e) Documentação de código.

6. Introdução a Estruturas de Dados (2 horas):
    (a) Estrutura de Dados: Pilha;
    (b) Padrão de projeto: Composição;
    (c) Programação Defensiva via Exceções e Asserções;
    (d) Polimorfismo Estático via *Generics*;
    (e) *Type-erasure*.
7. Polimorfismo Dinâmico (2 horas):
    (a) Interfaces;
    (b) *Lazy-Evaluation*;
    (c) Métodos *default*;
    (d) Classes abstratas;
    (e) Herança e problema do diamante.
8. Técnicas de programação (2 horas):
    (a) Boas práticas: Manutenção;
    (b) *Design-By-Contract*;
    (c) *Early-Return*;
    (d) Teoria de Tipos: Tipagem Estática;
    (e) Teoria de Tipos: Tipagem Forte.
9. Programação Funcional (2 horas):
    (a) *Callback*;
    (b) API Funcional de Java;
    (c) Classes Anônimas;
    (d) Funções *lambda*.
10. Pacotes e bibliotecas (2 horas):
    (a) Sistema de pacotes Java;
    (b) *Classpath*;
    (c) Makefiles;
    (d) Arquivo JAR como biblioteca;
    (e) Arquivo JAR executável.
11. (Extra) Otimização (2 horas):
    (a) Otimizações de Arquitetura;
    (b) Otimizações de Compiladores;
    (c) Otimizações da JVM;
    (d) Compilação *JIT* e *AOT*;
    (e) *Tail-call recursion*;
    (f) Complexidade amortizada.

7 Cronograma

O cronograma obedece a mesma ordem do Conteúdo Programático, sendo cada tópico uma aula.

Referências

[1] Oracle. Java Platform SE 10 Documentation. https://docs.oracle.com/javase/10/docs/api/index.html.

[2] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[3] Raul Sidnei Wazlawick. *Engenharia de Software para Sistemas de Informação: Conceitos e práticas que fazem sentido*. 2012.