



CURSO DE FORMAÇÃO PROGRAMADOR JAVA

Introdução a Plataforma JAVA - SE01

Objetivo Geral

Este módulo tem por objetivo capacitar os alunos com a sintaxe fundamental da linguagem Java e conhecimento das APIs básicas da plataforma JAVA SE.

Objetivos Específicos

Ao final desta disciplina, os participantes deverão ser capazes de desempenhar os objetivos específicos descritos abaixo.

Carga Horária: 20 horas

Objetivo	Conteúdo
1. Descrever a plataforma Java	✓ Visão geral da plataforma JAVA
2. Descrever e usar a sintaxe da linguagem Java para a escrita de programas.	✓ Estrutura de Programas ✓ Blocos de código ✓ Identificadores ✓ Palavras Reservadas ✓ Tipos primitivos ✓ Variáveis ✓ Operadores ✓ Tipos escalares e multidimensionais ✓ Comandos Condicionais ✓ Comandos de fluxo ✓ Comandos de repetição
3. Descrever as diferenças, vantagens e desvantagens do paradigma da orientação a objetos	✓ Benefícios da orientação a objetos ✓ Comparativo entre programação orientada a objetos e a programação procedural
4. Descrever e usar as primitivas Java para criação e manipulação de classes e objetos	✓ Classes ✓ Pacotes ✓ Atributos ✓ Métodos ✓ Objetos ✓ Modificadores de acesso ✓ Método construtores ✓ Modificador <i>static</i> ✓ Modificador <i>final</i> ✓ Modificador <i>native</i> ✓ Modificador <i>transient</i> ✓ Conversão de tipos ✓ Autoboxing e auto-unboxing ✓ Operador <i>instanceof</i>
5. Descrever e usar as primitivas Java para tratamento de exceções.	✓ Programação por Contrato ✓ Tratamento de exceções ✓ Exceções comuns ✓ Categorias de exceção ✓ Exceções Verificadas (Checked) ✓ Criação e Propagação de Exceções
6. Descreve e usar tipos enumerados	✓ Enumerações ✓ Apresentação de anti-padrões que viabilizam o uso de enumerações
7. Descrever e usar classes padrão do JDK para	✓ Classe <code>java.lang.Object</code>



a solução de pequenos problemas.

- ✓ Classe java.lang.String
- ✓ Classe java.lang.StringBuffer
- ✓ Classe java.util.StringTokenizer
- ✓ Classe java.lang.Math
- ✓ Classes Wrapper
- ✓ Classe java.util.Formatter
- ✓ Classe java.util.Scanner

Bibliografia

1. Java, Como Programar – Harvey Deitel et al., Bookman, 2007.
2. Use a Cabeça: Java – Kathy Sierra et al., Alta Books, 2007.

Programação Java Avançada - SE02

Objetivo Geral

Este módulo tem por objetivo capacitar os alunos no uso e aplicação de conceitos avançados da linguagem Java para a programação de aplicativos que usem arquivos, banco de dados e redes.

Objetivos Específicos

Ao final desta disciplina, os participantes deverão ser capazes de desempenhar os objetivos específicos descritos abaixo.

Carga horária: 40 horas

Objetivo	Conteúdo
8. Descrever, comparar e aplicar coleções Java para manipular estruturas de dados complexas	<ul style="list-style-type: none">✓ Frameworks Collections.✓ Interfaces e Classes de estruturas de dados do framework Collections
9. Descrever e usar tipos genéricos em Java.	<ul style="list-style-type: none">✓ Generics✓ Implementação de tipos genéricos.✓ Implementação de métodos genéricos.✓ Conceitos avançados de tipos genéricos.
10. Descrever e usar a API Java para manipulação de <i>streams</i> .	<ul style="list-style-type: none">✓ Manipulação de <i>streams</i>.✓ Hierarquia de classe de <i>streams</i>.✓ Filtros✓ Acesso aleatório a arquivos.
11. Descrever e usar a API Java JDBC para manipulação de bancos de dados relacionados.	<ul style="list-style-type: none">✓ Conceitos JDBC✓ Hierarquia de classes do JDBC✓ Tipos de Drivers✓ Conexões✓ Comandos SQL✓ Processamento de resultados.✓ Conceitos de mapeamento objeto relacional
12. Descrever e usar a API Java JDBC para manipulação de programas paralelos (threads)	<ul style="list-style-type: none">✓ A Classe Thread✓ Estados e prioridades de uma thread✓ Sincronização e monitores
13. Descrever e usar as primitivas Java para tratamento de exceções.	<ul style="list-style-type: none">✓ Programação por Contrato✓ Tratamento de exceções✓ Exceções comuns✓ Categorias de exceção✓ Exceções Verificadas (Checked)✓ Criação e Propagação de Exceções✓
14. Descrever e usar as primitivas Java para a escrita de programas em rede.	<ul style="list-style-type: none">✓ Potencialidades de rede em Java.✓ Conceitos básicos TCP, IP e URL.



- ✓ Classe URL
- ✓ Manipulação de Soquetes
- ✓ Aplicações cliente-servidor

Bibliografia

3. Java, Como Programar – Harvey Deitel et al., Bookman, 2007.
4. Use a Cabeça: Java – Kathy Sierra et al., Alta Books, 2007.

Workshop Programador Java - WS01

Objetivo Geral

Este módulo tem por objetivo capacitar os alunos nos objetivos do exame Sun Certified Programmer for Java 2 Platform 6.0 (SCJP)

Objetivos Específicos

Ao final desta disciplina, os participantes deverão ser capazes de desempenhar os objetivos específicos descritos abaixo, em conformidade com os objetivos definidos SUN definidos para a prova de certificação.

Carga horária: 16 horas

Usar e aplicar os elementos para declaração, inicialização e escopo de tipos e variáveis.	<ul style="list-style-type: none">✓ Desenvolver código para declarar classes (incluindo classes abstratas e todos os tipos de classes aninhadas), interfaces, e enums, e incluir o uso apropriado de pacotes (package) e expressões de import (incluindo static imports).✓ Desenvolver código para declarar interfaces.✓ Desenvolver código para implementar ou herdar (extends) uma ou mais interfaces✓ Desenvolver código para declarar uma classe abstrata (abstract).✓ Desenvolver código para herdar de uma classe abstrata.✓ Desenvolver código para declarar, inicializar, e usar primitivas, arrays, enums, e objetos como variáveis estáticas, de instancia, e locais. Também, usar identificadores válidos para nomes de variáveis.✓ Desenvolver código para declarar métodos estáticos e não-estáticos, e – se for apropriado - usar nomes de métodos com aderência ao padrão de nomeação de Javabeen. Também desenvolver código para declarar e usar uma lista de argumentos de tamanho variável.✓ A partir de um exemplo de código, determinar se um método está sobrescrevendo ou sobrecarregando corretamente outro método, e identificar valores de retorno válidos (incluindo retornos co-variantes), para um método.✓ A partir de um conjunto de classes e superclasses, desenvolver construtores para um ou mais classes.✓ A partir de uma declaração de classe, determinar se um construtor default será criado, e se for, determinar o comportamento deste construtor.



	<ul style="list-style-type: none">✓ A partir de uma lista de classes aninhadas ou não-aninhadas, escrever código para instanciar a classe.
Usar e aplicar fluxos de controle	<ul style="list-style-type: none">✓ Desenvolver código para implementar um comando if ou switch; e identificar tipos de argumentos válidos para estes comandos.✓ Desenvolver código para implementar todas as formas de loops e iteradores, incluindo o uso de for, o loop avançado (for-each), do, while, labels, break, e continue; e explicar os valores fornecidos pelas variáveis do contador do loop durante e depois da execução do loop.✓ Desenvolver código para fazer uso de afirmações (assertions), e distinguir apropriadamente de usos inapropriados de afirmações (assertions).✓ Desenvolver código para fazer uso de exceções (exceptions) e cláusulas de tratamento de exceções (try, catch, finally), e declarar métodos e sobrescrever métodos que disparam exceções (throw).✓ Reconhecer o efeito de se levantar uma exceção em um ponto específico em um fragmento de código. Note que a exceção pode ser uma runtime exception, uma checked exception, ou uma error exception.✓ Reconhecer situações que irão resultar em um disparo de uma das exceções abaixo: <i>ArrayIndexOutOfBoundsException, ClassCastException, IllegalArgumentException, IllegalStateException, NullPointerException, NumberFormatException, AssertionError, ExceptionInInitializerError, StackOverflowError</i> ou <i>NoClassDefFoundError</i>.✓ Identificar quais destas exceções são disparadas pela máquina virtual e reconhecer situações que de outra forma deverão ser disparadas programaticamente.



Usar e aplicar a API Java dos pacotes Lang, útil e IO

- ✓ Desenvolver código para usar classes primitivas empacotadoras (the primitive wrapper classes such as Boolean, Character, Double, Integer, etc.), e/ou autoboxing & unboxing. Discutir as diferenças entre as classes String, StringBuilder, e StringBuffer.
- ✓ A partir de um cenário envolvendo navegação no sistema de arquivos, leitura de arquivos, ou gravação de arquivos, desenvolver a solução correta usando as seguintes classes (algumas vezes em combinação), do pacote java.io: BufferedReader,BufferedWriter, File, FileReader, FileWriter e PrintWriter.
- ✓ Desenvolver código para serialização e/ou deserialização de objetos usando as seguintes APIs do pacote java.io: DataInputStream, DataOutputStream, FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream e Serializable.
- ✓ Usar as APIs J2SE padrão do pacote java.text para formatar corretamente ou fazer parser em datas, números, e valores de moeda (currency) para um local específico (locale); e, a partir de um cenário, determinar o método apropriado para usar se você quer usar o local padrão (default locale) ou um local específico.
- ✓ Descrever o propósito de uso da classe java.util.Locale.
- ✓ Escrever código para usar as APIs J2SE padrão do pacote java.util e java.util.regex para formatar ou fazer parser em strings ou streams. Para strings, escrever código que use a classes Pattern e Matcher e o método.split.
- ✓ Reconhecer e usar expressões regulares para combinar padrões (limitado a : . (ponto), * (asterisco), + (mais), ?, \d, \s, \w, [], ()). O uso de *, +, e ? deve se limitar a qualificadores, e o operador parêntese deve ser usado apenas como um mecanismo de agrupamento, não para capturar conteúdo durante a combinação.
- ✓ Para streams, escrever código usando as classes Formatter e Scanner e os métodos PrintWriter.format/printf.
- ✓ Reconhecer e usar parâmetros de formatação (limitados a: %b, %c, %d, %f, %s) para formatar string.
- ✓

Usar e aplicar código para programas concorrentes.

- ✓ Escrever código para definir, instanciar, e disparar (start) novas threads usando ambos java.lang.Thread e java.lang.Runnable.
- ✓ Reconhecer os estados nos quais uma thread pode existir, e identificar os caminhos nos quais uma thread pode transitar de um estado para outro.
- ✓ A partir de um cenário, escrever código para fazer uso apropriado de lock de objeto para proteger variáveis estáticas ou de instancia de



Usar e aplicar conceitos OO

- problemas de acessos concorrentes.
- ✓ A partir de um cenário, escrever código para fazer uso apropriado de `wait`, `notify`, ou `notifyAll`.
- ✓ Desenvolver código para implementar: encapsulamento firme (tight encapsulation), acoplamento fraco (loose coupling), e alta coesão (high cohesion) em classes, e descrever os benefícios.
- ✓ A partir de um cenário, desenvolver código para demonstrar o uso do polimorfismo.
- ✓ Adicionalmente, determinar quando a conversão (casting) será necessário e reconhecer erros de compilação versus erros de execução relativos à referência do objeto convertido.
- ✓ Explicar o efeito dos modificadores na herança com relação aos construtores, variáveis de instância ou estáticas, ou métodos estáticos.
- ✓ A partir de um cenário, desenvolver código para declarar e/ou chamar métodos sobrescritos ou sobrecarregados e código para declarar e/ou chamar construtores de superclasses sobrescritos ou sobrecarregados.
- ✓ Desenvolver código para implementar relacionamentos "É-UM" (IS-A) e/ou "TEM-UM" (HAS-A).



Usar e aplicar coleções e tipos genéricos

- ✓ A partir de um cenário de desenho, determine quais classes de coleções e/ou interfaces podem ser usadas para implementar corretamente o desenho, incluindo o uso da interface Comparable.
- ✓ Distinguir entre sobrescrita correta e incorreta dos métodos hashCode e equals, e explicar a diferença entre o uso do operador == e do método equals.
- ✓ Escrever código para usar a versão genérica da API Collections, em particular, das interfaces Set, List, e Map e implementação das classes. Reconhecer as limitações da API Collection não-generica e como refatorar o código usando as versões genéricas.
- ✓ Desenvolver código para usar adequadamente parametros do tipo das classes e interfaces declaradas, instanciar variáveis, argumentos de métodos, e tipos de retorno; e escrever métodos genericos ou métodos para fazer uso de coringas de tipos (wildcard types) e entender as similaridades e diferenças entre estas duas abordagens.
- ✓ Usar as capacidades do pacote java.util para escrever código para manipular uma lista para ordenar, fazer uma pesquisa binária, ou converter uma lista em um array.
- ✓ Usar as capacidades do pacote java.util para escrever código para manipular um array para ordenar, fazer uma pesquisa binária, ou converter um array em uma lista.
- ✓ Usar as interfaces java.util.Comparator e java.lang.Comparable para alterar a ordenação de listas e arrays. Além disso, reconhecer os efeitos da "ordenação natural" na ordenação das classes primitivas empacotadoras (wrapper classes) e da classe java.lang.String.



Usar e aplicar conceitos fundamentais da linguagem Java.

- ✓ A partir de um código de exemplo e um cenário, escrever código para usar o modificador de acesso apropriado, comandos de declaração de package, e import para interagir com o código de exemplo (através de acesso ou herança).
- ✓ A partir de um exemplo de classe e uma linha de comando, determinar o comportamento de execução específico.
- ✓ Determinar o efeito em cima de referências de objetos e valores de primitivas quando eles são passados em métodos que fazem atribuições ou outras operações de modificação nos parâmetros.
- ✓ A partir de um exemplo de código, reconhecer o ponto no qual um objeto torna-se elegível para coleção de lixo (garbage collection), e determinar qual é ou não garantido pelo sistema de coleção de lixo. Reconhecer o comportamento do método System.gc() e a finalização.
- ✓ A partir do nome completo (fully-qualified) de uma classe que está publicada (deployed) dentro e/ou fora de um arquivo JAR, construir a estrutura de diretórios apropriada para a classe. A partir de um código de exemplo e um caminho de classe (classpath), determinar se o caminho da classe (classpath) vai permitir que o código seja compilado com sucesso.
- ✓ Escrever código para aplicar corretamente o operador apropriado incluindo operadores de atribuição (limitados a: =, +=, -=), operadores aritméticos (limitados a: +, -, *, /, %, ++, --), operadores relacionais (limitados a: <, <=, >, >=, ==, !=), o operador instanceof, operadores lógicos (limitados a: &, |, ^, !, &&, ||), e operadores condicionais (?:), para produzir um resultado desejado. Escrever código para determinar a igualdade de dois objetos ou duas primitivas

Bibliografia

5. Java, Como Programar – Harvey Deitel et al., Bookman, 2007.
6. Use a Cabeça: Java – Kathy Sierra et al., Alta Books, 2007.
7. Certificação Java 5 Guia Preparatório Exame CX-310-055, Brasport, Roberto Rubinstein Serson
8. The Java Language Specification – Third Edition, James Gosling - <http://java.sun.com/docs/books/jls/> (Disponível em HTML e PDF)