Preparatório para Certificação OCJP 6

Objetivo

Preparar profissionais da linguagem Java para obter sucesso no exame **1Z0-851 – Java Standard Edition 6 Programmer Certified Professional**, através de aulas práticas, simulados e revisões dos principais pontos da prova. Curso totalmente prático e com foco no objetivo da prova através da realização de simulados com explicações exaustivas em cada uma das perguntas reunidas para cobrir todo o conteúdo e obter seu sucesso na Certificação Oracle. Obter o reconhecimento através da Certificação Oracle que estão entre as mais procuradas por sua credibilidade no mercado de Tecnologia da Informação.

Duração
32 horas/aula

Diferenciais X25

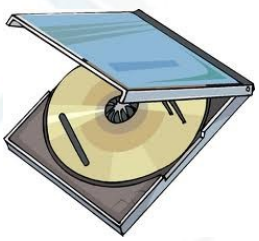
- *Instrutores altamente qualificados*
- *Livros como Material Didático*
- *Coffee-break*
- *Computadores de última geração*
- *Salas com projetores multimídia*
- *Somente 01 aluno por computador*
- *Certificado diferenciado pelo aproveitamento do aluno*
- *Parcerias internacionais*
- *Treinamentos in-company*
- *Treinamentos revisados periodicamente*

Pré-requisitos

Para obter um bom aproveitamento neste curso os alunos devem ter um conhecimento sólido da linguagem de programação JAVA. Leitura técnica no idioma Inglês.

Público Alvo

Profissionais que possuam o conhecimento sólido da linguagem Java e desejam confirmá-lo através da certificação oficial da ORACLE. Isso inclui, mas não limita a: Programadores, Desenvolvedores, Analistas de Sistemas e Estudantes.



Materiais

Será distribuído material eletrônico (em língua Inglesa) contendo artigos, exemplos, dicas e questões de simulados.

Conteúdo Programático

Declarações, Inicializações e Escopo

- Declarar classes (incluindo classes abstratas e todas as formas de classes aninhadas), interfaces e enumerações e incluir o uso apropriado do pacote e de importação de declarações (inclusive importações estáticas).
- Declarar, inicializar e utilizar primitivas, matrizes, enumerações e objetos, como por exemplo, estático e variáveis locais. Além disso, usar identificadores legais para nomes de variáveis.
- Implementar encapsulamento, acoplamento rígido e alta coesão em classes e descrever seus benefícios.
- Utilizar adequadamente parâmetros de tipo de declarações de classes ou interfaces, variáveis de instância, argumentos de métodos e tipos de retorno e escrever métodos genéricos ou métodos que façam uso de curingas e compreender as semelhanças e diferenças entre essas duas abordagens.
- Utilizar declarações e distinguir adequadamente seus usos.

Tipos Primitivos e Objetos

- Utilizar as classes *Wrapper* de primitivas (como *Boolean*, *Character*, *Integer*, *Double*, etc), usar *autoboxing* e *unboxing*. Discutir as diferenças entre as classes *String*, *StringBuilder* e *StringBuffer*.
- Distinguir substituições na correspondência entre métodos de *hashCode()* e *equals()*.
- Explicar a diferença entre o símbolo "==" e o método equals().
- Aplicar corretamente os operadores para produzir um resultado desejado, incluindo:
  - Operadores de atribuição (limitados a: =, + =, - =)
  - Operadores aritméticos (limitados a: +, -, *, /,%, + +, -)
  - Operadores relacionais (limitado a: <, <=,>,> =, ==, =!)
  - Operador *instanceof*
  - Operadores lógicos (limitado a: &, |, ^,, &&, | |)
  - Operador ternário (?:)
- Determinar a igualdade de dois objetos ou duas primitivas.

Modificadores, Construtores e Garbage Collector

- Explicar o efeito de modificadores na herança com relação a construtores, variáveis de instância ou estáticas e de instância ou métodos estáticos.
- Utilizar modificadores de acesso adequadamente, declarações de pacotes e as declarações de importação para interagir com (através de acesso ou herança).
- Reconhecer o ponto em que um objeto se torna elegível para o *Garbage Collector*
  - Determinar o que é e não é garantido pelo sistema.
  - Reconhecer os comportamentos da *Object*, através do método finalize() com base em um cenário de projeto.
  - Determinar quais as classes ou interfaces procede o recolhimento que deve ser usado, incluindo o uso da interface *Comparable*.

Conceitos de Orientação a Objetos

- Declarar ou invocar métodos substituídos ou sobrecarregados e código que declarar ou invocar construtores de superclasse ou sobrecarregados.
- Demonstrar o uso do polimorfismo. Além disso, determinar quando a intercalação será necessária e reconhecer compilador versus erros de execução relacionadas à referência de objeto.
- Fazer o uso apropriado de bloqueio de objeto para proteger variáveis estáticas ou exemplo de problemas de acesso simultâneo.
- Dado o nome totalmente qualificado de uma classe que é implantada dentro ou fora de um arquivo JAR, construir a estrutura de diretório apropriado para essa classe. Dado um código de exemplo e um caminho de classe, determinar se o *classpath* permite que o código seja compilado com sucesso.
- Reconhecer os estados em que um segmento pode existir, e identificar maneiras pelas quais um tópico pode fazer a transição de um estado para outro.

Formatação de Dados

- Utilizar API padrão da Java SE no pacote *java.text* corretamente formatar ou analisar datas, números e valores monetários para uma localidade específica, e, dado um cenário, determinar os métodos apropriados para utilizar uma localidade padrão ou específica. Descrever o propósito e uso da classe *java.util.Locale*.
- Utilizar as API padrão Java SE para os pacotes *java.util* e *java.util.regex* para formatar ou analisar classe *String* ou fluxos de dados.
    - No caso da classe String: utilizar as classes *Pattern* e *Matcher* e o método *String.split()*. Reconhecer e usar padrões de expressões regulares para correspondência (limitados a: **.** (Ponto), ***** (asterisco), **+** (plus), \d, \s, \w, [], ()?). O uso de *****, **+**, E? é limitado aos quantificadores e ao operador parênteses será usado como um mecanismo de agrupamento, e não para o conteúdo captura correspondente.
    - Para Fluxos de Dados, escrever código usando as classes *Formatter* e *Scanner* e os métodos PrintWriter.format() e System.out.printf(). Reconhecer e usar parâmetros de formatação (limitados a: b,% c%,% d,% f,% s) na formatação de strings.

Coleções e Generics

- Utilizar as versões genéricas da API *Collections*, em particular interfaces e classes de implementação da: Set, List e Map.
- Reconhecer as limitações da não-genérico API de coleções e como refatorar o código para usar as versões genéricas. Escrever código que usa o *NavigableSet* e interfaces *NavigableMap*.