Programação Orientada a Objetos

Aula 15 - Estudos Avançados em Java

Apresentação

Até aqui, vimos rapidamente o que o paradigma da orientação a objetos e a tecnologia Java podem nos oferecer para ajudar-nos no desenvolvimento de sistemas de software. Partimos de exemplos simples, de forma a isolar técnicas e estratégias de resolução. E fomos aplicando gradativamente cada um dos conceitos e técnicas da POO usando a linguagem Java.

Nesta última aula, traremos dicas, informações e sugestões para que você possa continuar seus estudos avançados de Java e de Orientação a Objetos. Você vai perceber que ainda há muito a aprender e ser explorado para fazer programas interessantes e sofisticados. Destacaremos as tecnologias, aplicações e roteiros de aprendizagem. Alguns pontos já foram vistos e serão melhor detalhados, outros serão apresentados agora. Queremos que você sinta-se motivado a prosseguir no estudo da tecnologia Java de desenvolvimento de sistemas. Boa aula!!!

Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você será capaz de:

- Conhecer e compreender um pouco sobre o mundo da programação na tecnologia Java que nos é disponível.

Refazendo a Trilha

Nosso passeio pelo mundo da programação orientada a objetos em Java está chegando ao fim. Mas, a viagem só está começando!

Esse é um momento oportuno para revermos o caminho que tomamos no aprendizado da programação orientada a objetos e na linguagem Java. Vamos (re)ver características da linguagem e novos caminhos que poderemos tomar para um estudo especializado e mais aprofundado da programação OO em Java.

Para utilizarmos a linguagem Java, tivemos que instalar um conjunto de programas, chamado de JDK (*Java Development Kit*), ou Kit de Desenvolvimento Java, que serve como a ferramenta e "bancada" principal do desenvolvedor Java.

Como vimos anteriormente, é no JDK que vem o compilador e a máquina virtual Java (responsável por executar nossos programas), além também das classes padrões que usamos para construir nossos programas.

Pois bem, vamos ver um pouco mais sobre as principais versões de distribuição de Java e suas bibliotecas de classes principais disponíveis.

1. **JSE – Java Standard Edition** – fornece as funcionalidades essenciais da linguagem e APIs básicas. É nela que está contida as principais classes de uso costumeiro em Java, tais como, String, System e a própria biblioteca Collection, usada nas nossas aulas. A plataforma JSE oferece muito mais classes para implementar aplicações bem mais sofisticadas, tais como bibliotecas para interfaces gráficas, acesso a arquivos, comunicação em rede, acesso a banco de dados.
2. **JEE – Java Enterprise Edition** – essa plataforma é usada para o desenvolvimento de aplicações corporativas e cliente-servidor. Uma das principais funcionalidades de JEE atualmente é oferecer tecnologias para o desenvolvimento de aplicações web, tais como: (i) Servlets – permite definir classes capazes de receber dados de formulários web; (ii) Java Server Pages (JSP) – possibilita a construção de páginas HTML dinâmicas; e (iii) Enterprise Java Beans (EJB) – permite

definir classes que oferecem serviços distribuídos os quais contêm as regras de negócio principal da aplicação.

3. **JME – Java Micro Edition** – É a plataforma de classes Java voltada para os dispositivos com capacidade reduzida de processamento e armazenamento de dados, tais como aparelhos celulares, palms etc. Ela é dessa forma usada para desenvolvimento das aplicações denominadas de “embarcada”, que está presente em brinquedos, robôs, eletrodomésticos etc.

Esses 3 (três) grandes grupos de tecnologias Java envolvem uma gama de outras tecnologias, como foi o exemplo apresentado para *Java Enterprise Edition*. Porém, essa divisão serve para classificá-las ou organizá-las de forma estratégica de acordo com a categoria de problemas (e soluções) que iremos trabalhar.

Vídeo 02 - Refazendo a Trilha

Programação e Modelagem OO

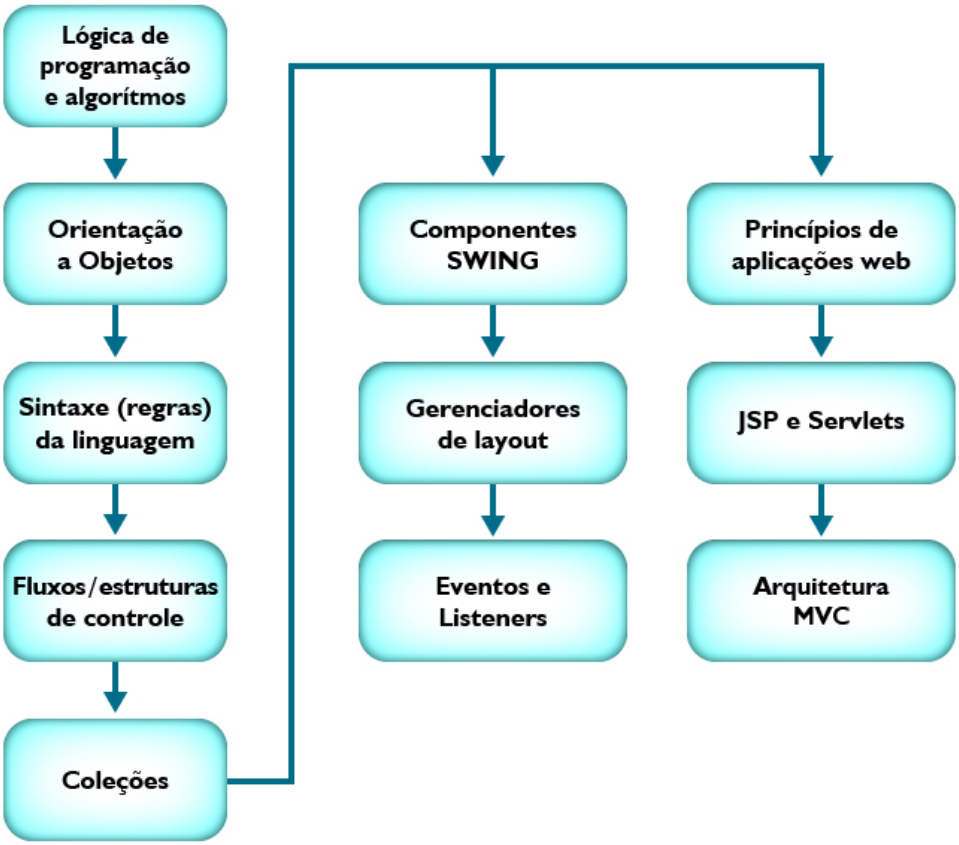
As classes são elementos fundamentais da linguagem Java, as quais definem tanto atributos (dados) quanto métodos (operações) que atuam sobre tais atributos e/ou seus parâmetros. Na programação em Java, é uma rotina criar e utilizar suas próprias classes. Em alguns casos, é comum também reusar classes que são definidas para um sistema, em um outro sistema que precise das mesmas funcionalidades.

Ao longo do curso, vimos também que está disponível para os projetistas e programadores Java a criação e a utilização de interfaces, que são declarações de métodos e atributos, os quais obrigam as classes que as implementam a definir a sua implementação para esses métodos. Fazer uso de interfaces não é trivial e requer um grau de maturidade considerável no projeto de sistemas.

Criando um Mapa para Guiar os Estudos

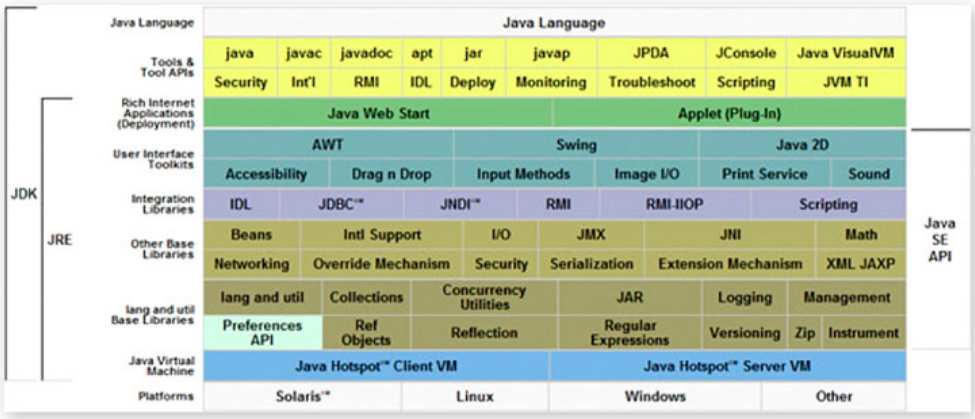
Partindo da visão de que a imensa maioria dos desenvolvedores Java tende a programar sistemas Web ou desktop, vamos construir um modelo que irá indicar os caminhos por onde o programador iniciante Java terá que passar para poder explorar com propriedade os recursos que essa linguagem oferece. O fluxo abaixo indica sugestões para o estudo da programação OO com a linguagem Java.

Figura 01 - Quadro geral de tecnologias Java



Mais à frente, serão indicadas fontes de pesquisa onde será possível encontrar materiais sobre todas as tecnologias mencionadas. O quadro da Figura 1 indica agrupamentos de tecnologias Java com os mesmos fins, porém, poderemos visualizar em detalhes o que a plataforma *Java Standard Edition* (JSE) pode nos oferecer, observando a Figura 2. Esse quadro foi retirado do site oficial da Sun Microsystems e compõe sua documentação.

Figura 02 - Quadro detalhado de tecnologias Java Standard Edition



Vídeo 03 - Prosseguindo com os Estudos

Depois de dominar os conhecimentos essenciais deste curso, o próximo passo é escolher um caminho e seguir em frente. Dentre eles, veremos dois caminhos comuns a serem seguidos: (i) desenvolvimento de aplicações para desktop; e (ii) desenvolvimento de aplicações web.

Programação de Aplicações Desktop

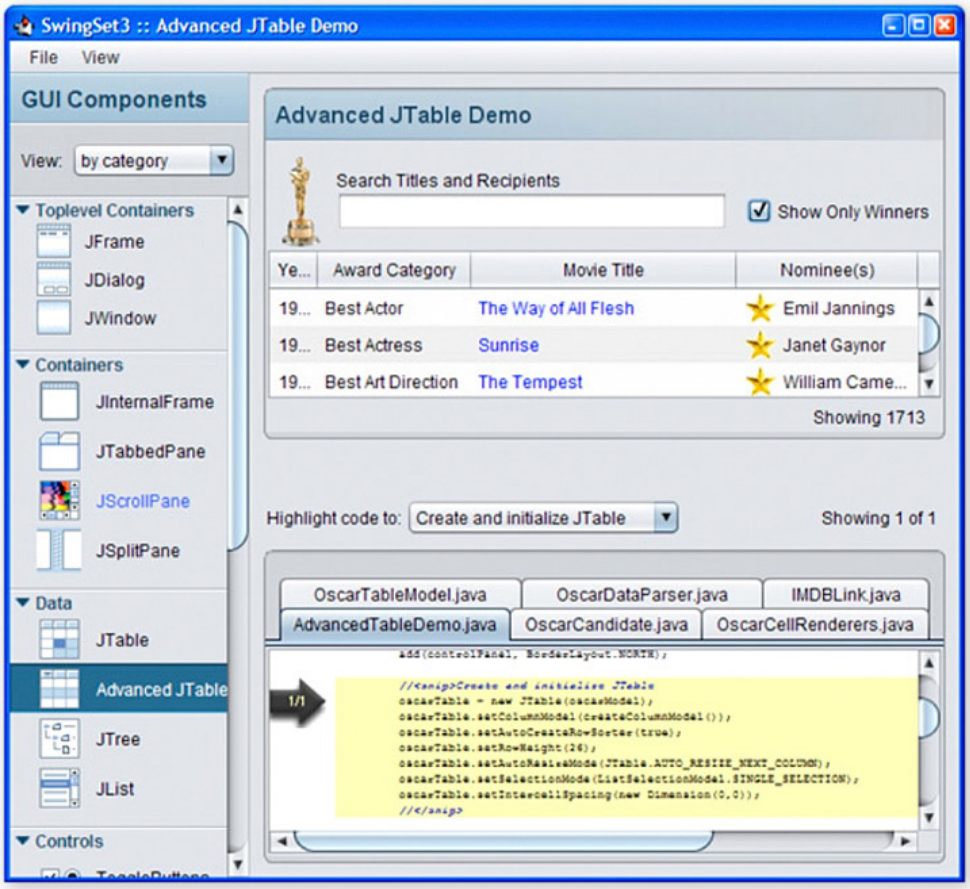
No*Java Standard Edition*, a biblioteca (API – *Application Programming Interface*) fundamental para desenvolvimento de interfaces gráficas (botões, janelas, janelas de diálogos), é conhecida como Swing.

Lá, encontraremos tudo que precisamos para o desenho (design) de telas gráficas com bastante recursos, como arrastar-e-soltar, e a possibilidade de incorporar características multimídia para reprodução de vídeos e áudio. Vejamos um exemplo da própria Sun presente no JSE no seguinte diretório:

<diretório de instalação do jdk>/demo/jfc/SwingSet3/readme.html

Lendo esse documento/página HTML, você terá a opção de executar a aplicação com a interface apresentada na Figura 3. Nela, podemos observar todos os componentes e seus comportamentos.

Figura 03 - Exemplo de interface Swing da Sun



A API Swing é extensa e compreende os 18 (dezoito) pacotes, os quais são listados a seguir.

javax.accessibility	javax.swing.plaf	javax.swing.text
javax.swing	javax.swing.plaf.basic	javax.swing.text.html
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.html.parser
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.text.rtf
javax.swing.event	javax.swing.plaf.synth	javax.swing.tree

javax.swing.filechooser	javax.swing.table	javax.swing.undo

Tratando Eventos

É preciso um considerável aprendizado até estar apto a programar utilizando as classes dos pacotes javax.swing e java.awt, pois elas utilizam conceitos próprios de disparo e captura de eventos.

Os eventos são utilizados para representar ações realizadas pelo usuário, sistema operacional ou temporizadores, ou seja, entidades externas ou internas ao sistema capazes de indicar um fenômeno que modifique estados dentro da aplicação. Cada ação do usuário pressupõe um Evento, e do outro lado, é preciso existir uma entidade capaz de “ouvir” esses eventos e realizar as ações necessárias. Esses são os *listeners*, cadastrados para receberem determinados eventos.

Gerenciando Layout

A construção de uma interface gráfica que traga conforto ao usuário e que obedeça a critérios de usabilidade precisa ter seu design bem planejado e um *layout* de componentes gráficos que tenha um comportamento estável quando a janela for ocultada, movida e redimensionada.

A linguagem Java possui classes especiais, denominadas de classes de Layout, voltadas à manutenção das posições e dimensões dos componentes visíveis e de componentes que servem como organizadores ou containers de outros componentes.

Banco de Dados

O acesso a banco de dados em Java é feito pelas classes do pacote java.sql, que compõem a API JDBC (*Java Database Connectivity*). Nela, encontramos classes para estabelecimento de conexão com um banco de dados, execução de comandos ao banco de dados (escritos em uma linguagem padrão chamada de SQL), gerenciamento de transações, e outras funcionalidades. O estudo da API JDBC pressupõe conhecimentos básicos de banco de dados e linguagem SQL. Porém, dominar essa API leva poucas horas de estudo.

Programação Web

Atualmente, a tendência aponta para um mundo de sistemas corporativos integrados a outras redes e operando online em várias transações com sistemas de clientes, fornecedores e governos, onde seus usuários estão distribuídos geograficamente. Nesse contexto, os sistemas web se consolidaram ao longo das últimas duas décadas, e se estabeleceram como uma plataforma padrão.

A plataforma de sistemas web permite que seus clientes se mantenham atualizados automaticamente e, com custo mínimo, pois não necessitam realizar downloads e configurações constantemente a cada nova versão de softwares disponibilizados.

Mas, como tudo tem um lado negativo, a web carece de padronizações. Atualmente, órgãos como W3C padronizam, permitindo que provedores de ferramentas tenham um terreno menos movediço para desenvolver tecnologias que facilitem o desenvolvimento de sistemas web. Ações como essa não corrigem todas as discrepâncias geradas ao longo de anos por fabricantes de browsers e pequenos protocolos proprietários. Logo, desenvolver para a web é estar “costurando” sempre uma colcha de retalhos tecnológicos.

Vídeo 04 - Programação Web

Designer x Programador

No desenvolvimento web, é comum a separação clara entre dois tipos principais de profissionais:

- ***Web designers*** – responsáveis por produzir a arte gráfica de um site web, definindo desde sua organização geral, até imagens, cores e

fontes de texto usadas em diferentes páginas web;

- **Desenvolvedor ou programador** – aqueles responsáveis por programar as classes do sistema web, que manipulam e validam as informações enviadas por formulários e recupera e armazena tais informações em um banco de dados.

Uma questão sempre recorrente é: quanto um desenvolvedor web precisa ser um *webdesigner*, ou quanto um *web designer* precisa entender de programação para desenvolver uma aplicação web agradável e funcional. Essa questão não precisa ser respondida, pois a distinção entre essas funções é bastante grande, são perfis de profissionais completamente incompatíveis (conforme a descrição acima já demonstrada), porém, por muito tempo, as tecnologias eram praticamente indissociáveis. Atualmente, já podemos pensar em divisão de responsabilidades: ser desenvolvedor web não significa ser *web designer*!

Mas, vale salientar que é preciso conhecer as tecnologias envolvidas, pois é necessária uma boa comunicação com o setor de *design*, aquele que realmente traz arte e beleza para *sites* web. Mesmo o desenvolvedor web não está livre de ter que conhecer sobre tecnologias comuns hoje em dia, tais como: HTML, CSS e JavaScript.

Tecnologias

Vejamos algumas siglas e tecnologias Java para web.

- **JSP – Java Server Pages** – Tecnologia voltada para a geração dinâmica de páginas HTML. Todo o conteúdo da aplicação precisa ser renderizado em forma de conteúdo web, para isso, os valores contidos na página e as *tags* JSP são transformadas em HTML, como se já tivesse sido escrito daquela maneira.

- **Servlet** – São pequenos programas rodando no lado servidor e responsável por receber e processar as requisições HTTP. Geralmente, utilizam-se as classes servlets para receber as informações provenientes de formulários web, e utilizamos classes Java para processar, consultar e preencher dados que precisam ser retornados ao usuário.

- **MVC (Modelo-Visão-Controlador)** – Arquitetura muito utilizada por desenvolvedores web para separar as responsabilidades dos códigos em três grupos distintos: (i) Modelo – representando as classes de negócio da aplicação; (ii) Visão – conjunto de páginas JSP responsável pela visualização do conteúdo; e (iii) Controle – poderemos compará-lo ao "meio-de-campo" da arquitetura, serve de ponte entre a visão e o modelo, e é implementado pelos servlets.

As páginas JSP são traduzidas em códigos Java, arquivos .java e compiladas em servlet, arquivos .class, que rodam em um sistema chamado de container web.

Por exemplo: Tomcat do projeto Apache, Glassfish da Sun Microsystems, entre outros.

Ferramentas

Não há desenvolvimento de software sem os softwares para desenvolvê-los! Vamos a eles. O próprio JDK é o software necessário e indispensável para se desenvolver um programa em Java, pois só precisamos, de forma simplificada, de um editor de textos, do compilador (javac.exe) e da máquina virtual para executá-lo (java.exe). Porém, é impraticável e pouco produtivo usar apenas tais ferramentas para desenvolvimento.

Atualmente, existe uma ampla gama de ferramentas de programação em Java com uma quantidade bem maior de recursos disponíveis. Tais ferramentas são conhecidas como IDEs (*IntegratedDevelopmentEnvironment*), Ambientes de Desenvolvimento Integrado. Elas trazem uma série de vantagens para os programadores, tais como:

- executam atividades repetitivas, como compilar todas as classes de um projeto, permitir a criação automática de construtores e métodos set() e get() de classes existentes, geração de código parcial de classes;

- possibilitam sempre uma visão privilegiada do que está sendo desenvolvido, quais classes possuem erros, os pacotes existentes com as respectivas classes;

- possuem editores que oferecem uma visualização agradável e que diferencie os elementos sintaticamente diferentes;

- possibilitam uma rápida localização de *bugs*, que possa gerar documentação do código.

Dentre essas e várias outras características, destacamos algumas ferramentas para o desenvolvimento em Java, as quais estão disponíveis gratuitamente para o seu uso.

- **Eclipse** – <www.eclipse.org>

  Projeto mantido por várias empresas e encabeçado pela IBM, que, entre 1999 e 2001, desenvolveu de forma fechada a ferramenta, porém, quando já se encontrava estável e com sua estrutura consolidada, tornou a IDE Eclipse disponível para a comunidade de software livre.

  Sua estrutura é baseada em plugins (componentes que se interligam ao módulo principal da ferramenta), os quais possibilitam um ajuste perfeito às necessidades do desenvolvedor, seja ele um projeto desktop, web ou mesmo para dispositivos móveis.

- **NetBeans** - <www.netbeans.org>

  A Sun conquistou espaços entre as principais IDEs com essa ferramenta após ter realizada algumas tentativas. O NetBeans veio com um conceito voltado desde o início para a comunidade de software livre e com características originais.

  No princípio, tínhamos o NetBeans como ferramenta de desenvolvimento web e o Eclipse como desenvolvimento desktop, porém, com o avanço das duas IDEs, hoje a escolha é baseada no suporte a tecnologias específicas e, principalmente, no estilo ou perfil do desenvolvedor.

- **JEdit** - <www.jedit.org/>

Ferramenta simples para edição e compilação de código Java. É uma alternativa a editores de texto comuns não reconhecedores da sintaxe Java.

Bem, chegamos ao final de nossa aula. É importante que você saiba que o surgimento e evolução de técnicas (tais como, orientação a objetos), tecnologias e ferramentas de programação ao longo dos últimos anos têm trazido grandes benefícios e produtividade para o desenvolvimento dos sistemas atuais. Entretanto, nunca fomos tão exigidos! De forma que não podemos relaxar quanto a nos mantermos atualizados e sempre dispostos e motivados a aprender novas técnicas e tecnologias, rever nossas práticas e aumentarmos nossa eficiência. Nunca os clientes pediram tanto e em tão pouco tempo para os engenheiros de software!

Bem, eis um novo mundo para você! Siga viagem, se aventure! E saiba que todo esforço sempre será recompensado, pois existe e continuarão a existir ao longo dos próximos anos inúmeras oportunidades para aqueles que desejam trabalhar com desenvolvimento de software. Boa Viagem!

Leitura Complementar

Há muitos *sites* na web cujo conteúdo é de grande valia para quem quer se tornar um bom desenvolvedor Java. A lista a seguir não esgota as possibilidades dos recursos Java, porém, já é um bom começo.

- Sites oficiais ou de conteúdos autorizados: IBM DW: <www-130.ibm.com/developerworks/java>
- O'Reilly On Java: <www.onjava.com>
- Java.Net: <www.oracle.com/splash/java.net/index.html>

Alguns sites não oficiais, mas mantidos por desenvolvedores sérios, servem como pontos de encontro e de discussão entre os desenvolvedores e um movimentado fórum de discussão para esclarecimento de dúvidas de iniciantes a *experts*. Os *Java UserGroups* (JUGs) são sempre um bom recurso, veja alguns:

- Portal Java: <www.portaljava.com.br>
- GUJ: <www.guj.com.br>
- JavaFree: <www.javafree.com.br>
- SouJava: <www.soujava.org.br>
- DFJUG: <www.dfjug.org>

Existem também revistas nacionais que sempre apresentam matérias para iniciantes e experientes. Sempre com atualizações importantes e apresentação de tendências no mundo Java:

- Java Magazine: <www.javamagazine.com.br>
- Mundo Java: <www.mundojava.com.br>
- The Java Tutorial: <www.java.sun.com/docs/books/tutorial>

Esse último é uma coleção de tutoriais que serve de ponto de partida para as mais variadas APIs Java. Desde os conceitos fundamentais a recursos avançados como programação com *threads*, conexão com banco de dados, acesso à rede e tantos outros. Ele também apresenta vários exemplos prontos para downloads, sendo mantido pela própria Sun, empresa que criou a linguagem. Esses tutoriais são atualizados anualmente pela Sun de acordo com a evolução da linguagem Java e suas tecnologias associadas. Os diversos exemplos existentes do tutorial estão também disponíveis para os usuários poderem fazer o download e executar na sua própria máquina.

Resumo

Nesta última aula, você foi apresentado às tecnologias Java, as quais não foram abordadas ao longo do nosso estudo por se tratarem de conteúdo específico ou extremamente técnico para iniciantes. Porém, a porta está aberta para os que quiserem se aventurar no mundo da Programação Orientada a Objetos com Java. É preciso dedicação, muitas leituras e muita prática, como fizemos em nossos exemplos. É importante exercitarmos todas as técnicas isoladamente para sermos capazes de, harmoniosamente, integrá-las em uma única solução ou um sistema complexo.

Autoavaliação

Para as atividades a seguir, use todos os conhecimentos adquiridos durante o curso, além de pesquisar na web utilizando os links apresentados nesta aula.

1. Qual é a diferença entre JSE, JEE e JME?
2. Pesquise sobre Java Server Pages (JSP) e crie um projeto, usando JSP, para mostrar a mensagem:”Seja bem-vindo <SeuNome>!”.
3. Pesquise sobre Servlet e crie um projeto, usando Servlet, para mostrar a mensagem:”Seja bem-vindo <SeuNome>!”.
4. Usando o Eclipse ou NetBeans, crie uma aplicação desktop utilizando a API Swing para cadastro de pessoas com os seguintes campos: nome, data de

nascimento, sexo, e-mail. O cadastro pode ser armazenado temporariamente (enquanto executa o programa) em um ArrayList. Além de armazenar a aplicação, deve permitir consultar pelo nome para:

- Exibir os dados da pessoa cadastrada;
- Alterar os dados da pessoa cadastrada;
- Remover a pessoa cadastrada.

5. Pesquise mais sobre Modelo-Visão-Controlador (MVC) e aplique no projeto da questão 4.

Referências

THE JAVA Tutorials. Disponível em: <http://java.sun.com/docs/books/tutorial/>. Acesso em: 17 maio 2010.