Curso de Pós-Graduação Lato Sensu a Distância em Engenharia de Software

Programação Orientada a Objetos com Java: 1ª parte

INTRODUÇÃO A DISCIPLINA

Data de Início:
08 de março de 2010 às 08h00min

Data de Encerramento:
09 de Abril de 2010 às 23h55min.

Data da Prova:
23 de Abril de 2010 de 17h30min às 20h30min.

Professor: Delano Brandes Marques

Introdução

Antes de iniciarmos a apresentação dos conceitos básicos do desenvolvimento orientado a objetos com Java, vamos primeiramente conceituar esta tecnologia.

Tecnologia? Sim, pode-se dizer que Java é uma tecnologia. Java pode ser vista como uma tecnologia utilizada no desenvolvimento de qualquer aplicação. Basicamente esta tecnologia constitui-se de uma linguagem de programação, um ambiente de desenvolvimento, um de aplicação e um de distribuição. Ao se programar em Java faz-se uso de uma linguagem de programação e um ambiente de desenvolvimento Java para se desenvolver um software. O software gerado será executado em um ambiente de distribuição Java. Esta é a tecnologia Java.

Esta tecnologia é a mais utilizada por programadores e desenvolvedores atualmente. Esta preferência se justifica devido a algumas características da linguagem. Abaixo apresentamos algumas delas:

Dentre as razões para escolhermos a linguagem Java dentre as demais, podemos destacar as apresentadas a seguir.

Java é orientada a objetos: diferentemente de outras linguagens de programação que permitem a existência de funções e variáveis em diferentes partes de um programa, em Java, todas as variáveis e métodos estão localizados dentro de classes, forçando o uso do paradigma da programação de orientação a objetos, e conseqüentemente todos seus benefícios.

Java é portável: o código-fonte de um programa em Java pode ser compilado em qualquer computador, independente do sistema operacional utilizado, desde que haja nele uma máquina virtual Java.

Java é gratuita: a máquina virtual Java, está disponível no site da Sun da mesma forma que o ambiente de desenvolvimento gratuito do Java – JDK (*Java Development Kit*).

Java é robusta: aspectos como alocação e liberação de memória, e o uso de ponteiros, fontes de erros mais freqüentes em programas desenvolvidos em C ou C++, são

gerenciados internamente, resultando em maior segurança para os programas sem diminuir sua utilidade. Além disso, Java também possui um poderoso mecanismo de exceções permitindo melhor tratamento de erros em tempo de execução.

Java possui várias bibliotecas prontas: a implementação de mecanismos de entrada e saída, manipulação de strings, recursos de acesso à Internet, estruturas de dados, implementação de interfaces gráficas, entre outros, já se encontram disponíveis em bibliotecas de classes Java. Estas bibliotecas são padrão de Java, ou seja, qualquer máquina virtual Java permite o uso destas bibliotecas.

O desenvolvimento da linguagem Java teve inicio em 1991 com o Projeto Green iniciado por James Gosling, Mike Sheridan e Patrick Naughton. A princípio o projeto não tinha como objetivo o desenvolvimento de uma linguagem, e sim ampliar os recursos de conectividade e acessibilidade em dispositivos eletrodomésticos, como aspiradores, liquidificadores, TVs, videocassetes e outros. No projeto surgiu a linguagem Oak, que posteriormente seria rebatizada de Java que teve seu lançamento oficial em 1995. O histórico da linguagem Java pode ser visto nos links:

- http://java.sun.com/features/1998/05/birthday.html
- http://www.java.com/en/javahistory/timeline.jsp

Curiosidades:

Duke: este pequeno personagem de desenho animado foi criado por Joe Palrang no Green Project da Sun. Duke se tornou o mascote Java assim que a tecnologia foi anunciada. Seu objetivo no projeto Green era auxiliar o usuário na interface do projeto. A partir de agora ele irá nos ajudar ao longo da disciplina aparecendo em pontos importantes do texto.

Java Cup: Java é o nome de uma ilha na Indonésia onde um dos principais produtos da agricultura comercial é o café. James Gosling primeiramente nomeou a linguagem obtida no Projeto Green de "Oak" (carvalho em inglês) devido a uma árvore que avistava na janela de seu escritório. Bem, alguns afirmam que o nome da linguagem foi mudado para Java devido ao café da ilha de Java era muito consumido pela equipe do Projeto Green. Existem outras versões espalhadas pela internet, verifique.

Java é desenvolvida e mantida pela Sun (http://www.sun.com).

Seu site principal é http://java.sun.com .

A seguir iremos apresentar os principais conceitos de Java necessários para o entendimento do funcionamento desta tecnologia. Em seguida iremos dar os primeiros passos na programação com a linguagem Java.

Fundamentos da Linguagem Java

Para entender o funcionamento da Máquina Virtual Java (*Java Virtual Machine* - JVM), um dos principais conceitos de Java que a diferencia de outras linguagens, vamos primeiramente relembrar como uma aplicação é gerada a partir de seu código fonte.

Em uma linguagem de programação como C e Pascal, por exemplo, ao compilar um código fonte um código binário (executável) é gerado para uma plataforma e sistema operacional específicos (Figura 01).

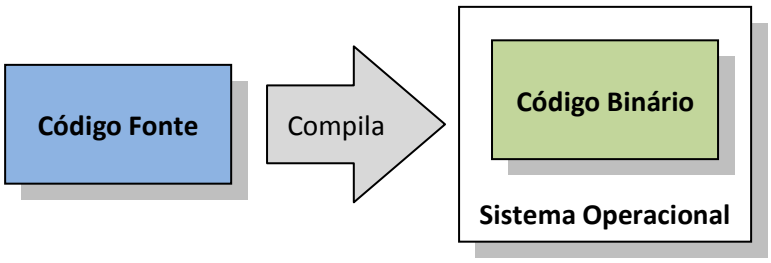


Figura 01 – Compilação de um código fonte

Diversas vezes, o próprio código fonte é desenvolvido objetivando uma única plataforma resultando em um código binário executado pelo sistema operacional. Conseqüentemente, esta aplicação deve saber "conversar" com o sistema operacional em questão. Desta forma teremos um código binário específico para cada sistema operacional. Isso faz com que seja necessário compilar o código fonte novamente a

cada vez que se desejar utilizar este mesmo programa em outro sistema operacional (figura 02).

Java faz uso do conceito de máquina virtual, uma máquina imaginária implementada através de um software emulador em uma máquina real, incluindo entre o sistema operacional e a aplicação uma camada. A JVM simula um computador permitindo que a aplicação rode sem envolvimento com o sistema operacional. Esta camada extra é responsável pela "tradução", entre outras finalidades, do que uma aplicação deseja realizar para as chamadas específicas do sistema operacional onde está sendo executada.

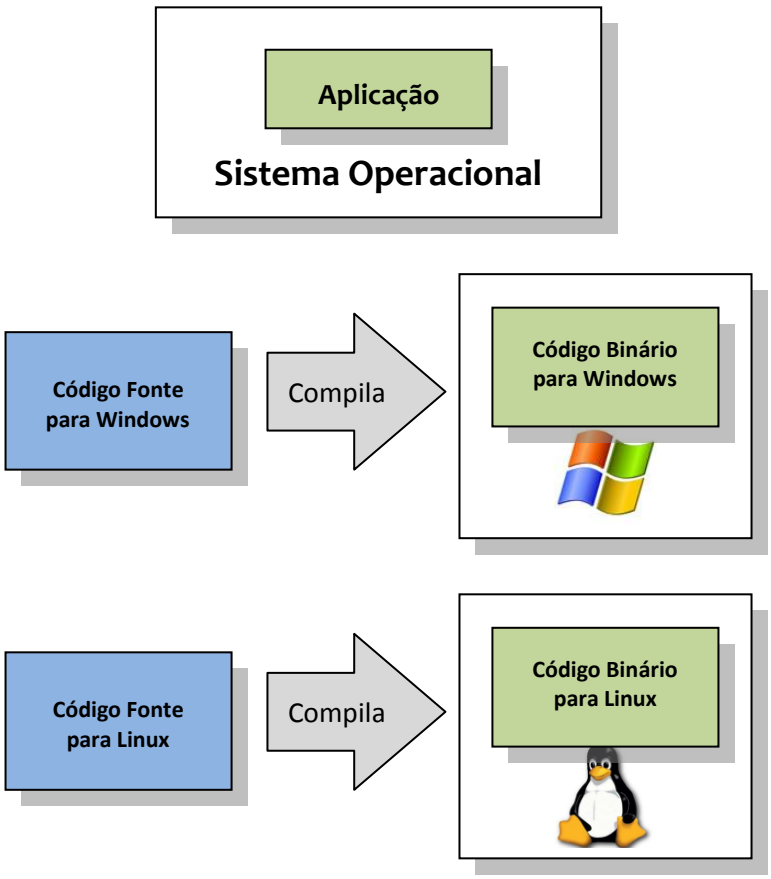


Figura 02 – Compilação de um código fonte para diferentes plataformas

Quando um código fonte em Java (extensão *.java*) é compilado um código intermediário entre o código fonte e o código de máquina é gerado. Este código intermediário é conhecido como *bytecode* (extensão *.class*). O *bytecode* será então

interpretado e executado pela JVM. Diferentemente de outras linguagens de programação (sem máquina virtual) o *bytecode*, resultado da compilação do código fonte, irá servir para diferentes sistemas operacionais já será "traduzido" pela máquina virtual.

Sendo assim, o desenvolvedor não irá se preocupar com especificidades de um sistema operacional ao migrar a aplicação para outra plataforma diferente da utilizada em seu desenvolvimento. O código fonte responsável pela abertura de um formulário no Linux ou no Windows, por exemplo, será o mesmo. Com isso obtém-se independência de plataforma sendo necessária apenas uma máquina virtual Java para aquele sistema operacional. Neste aspecto, o sistema operacional em que sua aplicação irá rodar, ou o tipo de máquina, não serão mais preocupações para o programador. "Escreva uma vez, execute em qualquer lugar" (WORA – *Write Once, Run Anywhere)* lema da linguagem Java devido a sua portabilidade (figura 03).

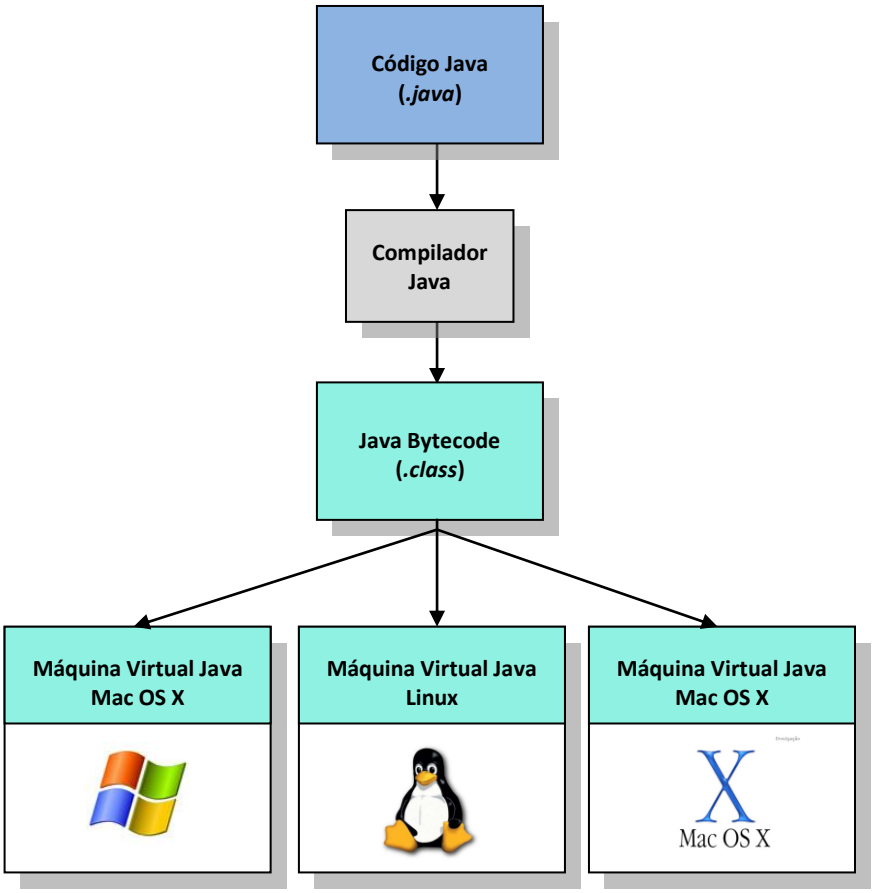


Figura 03 - Compilação de código fonte em Java

Antes de aprendermos como preparar nosso ambiente para o desenvolvimento de aplicações em Java vamos conhecer os principais elementos utilizados neste universo:

JRE (*Java Runtime Enviroment*): ambiente de execução Java formado pela Java API (bibliotecas de funcionalidades), a máquina virtual Java, além de outros elementos necessários para a execução de aplicações desenvolvidas em Java.

JVM (*Java Virtual Machine*): software responsável pela interpretação do *bytecode*, possibilitando a execução de programas desenvolvidos em Java.

Java API (*Java Application Programming Interface*): conjunto de bibliotecas de classes Java que disponibilizam diversas funcionalidades.

JDK (*Java Development Kit*): também conhecido como SDK (*Software Development Kit*) é o ambiente destinado a desenvolvedores. Possui um conjunto de ferramentas além do JRE como compilador (javac), interpretador (java), gerador de documentação (javadoc), entre outras.

Abaixo uma ilustração dos componentes acima citados:

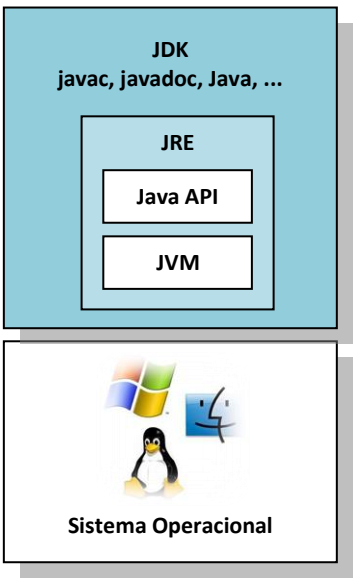


Figura 04 – Relação entre JDK, JRE, Java API, JVM e o Sistema Operacional

Preparação do Ambiente de Desenvolvimento Java

Após conhecer um pouco a respeito do Java, vamos iniciar a preparação do nosso ambiente de desenvolvimento. Tanto o ambiente de execução de aplicações Java (JRE) como o de desenvolvimento (JDK) podem ser obtidos no link: http://java.sun.com/javase/downloads/index.jsp. Como nosso objetivo é o desenvolvimento de aplicações Java, efetue o download da última versão do JDK. No momento em que este material foi desenvolvido a última versão disponível era o JDK 6 Update 18, disponível no link http://java.sun.com/javase/downloads/widget/jdk6.jsp (figura 05).

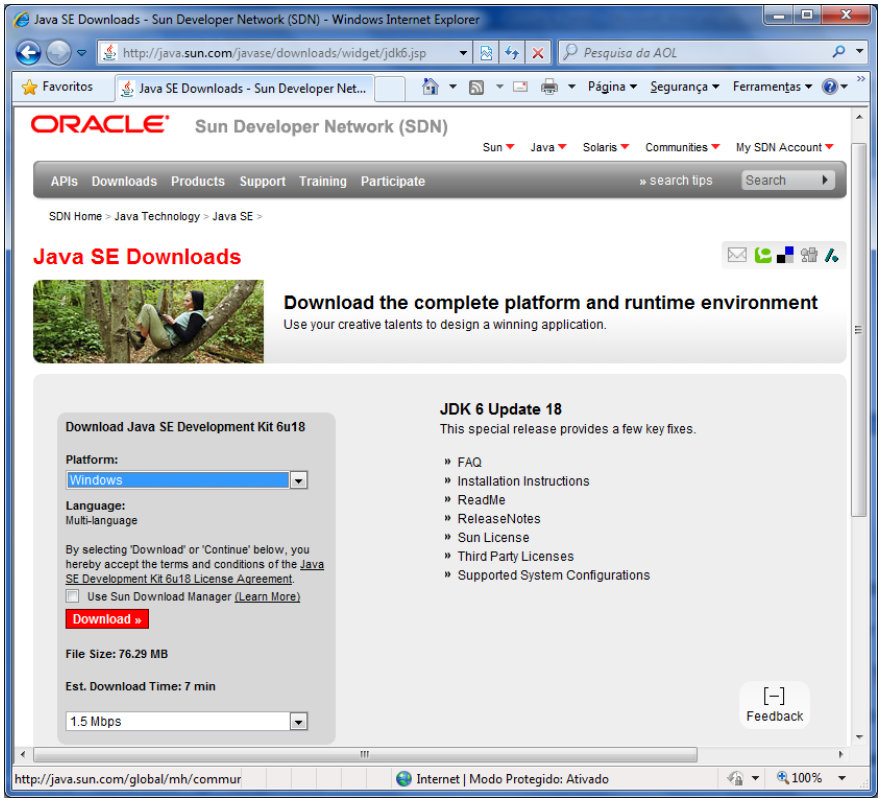


Figura 05 – Download do JDK 6 update 18

Selecione a plataforma utilizada (Windows, Linux, Solaris, etc) e dê um clique em **Download**. Em seguida uma tela de login opcional será apresentada. Caso não tenha uma conta, e não deseje se cadastrar dê um clique no link **Skip the Step** (Figura 06).

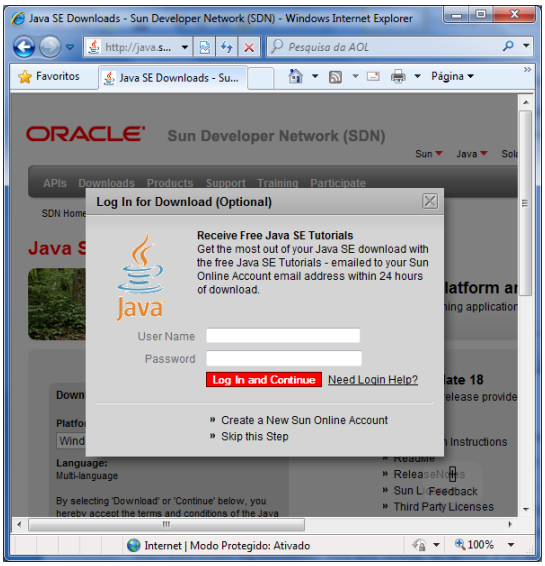


Figura 06 – Tela de login opcional no Download do JDK

Após ser efetuado o download do JDK, sua instalação é bem simples. Basta prosseguir a cada nova tela na instalação com as opções padrões.

Existem diversas ferramentas que auxiliam o desenvolvimento de aplicações conhecidas como IDEs. Uma IDE (*Integrated Development Enviroment*) elimina o uso de linhas de comando para compilação e execução de um programa. Outros benefícios são obtidos pelo uso de seus editores de código com diversos recursos úteis e que podem ser utilizados para facilitar a vida de um programador. Para dar início, iremos utilizar o JCreator, uma IDE bem simples e fácil de usar, ideal para quem está iniciando o desenvolvimento de aplicações Java. Posteriormente faremos uso do Eclipse (http://www.eclipse.org/) IDE da IBM muito popular no desenvolvimento Java. Da mesma forma que o JDK, o JCreator está disponível para download na internet. A ferramenta pode ser obtida no link http://www.jcreator.org/download.htm (figura 07). A versão JCreator LE version é livre.

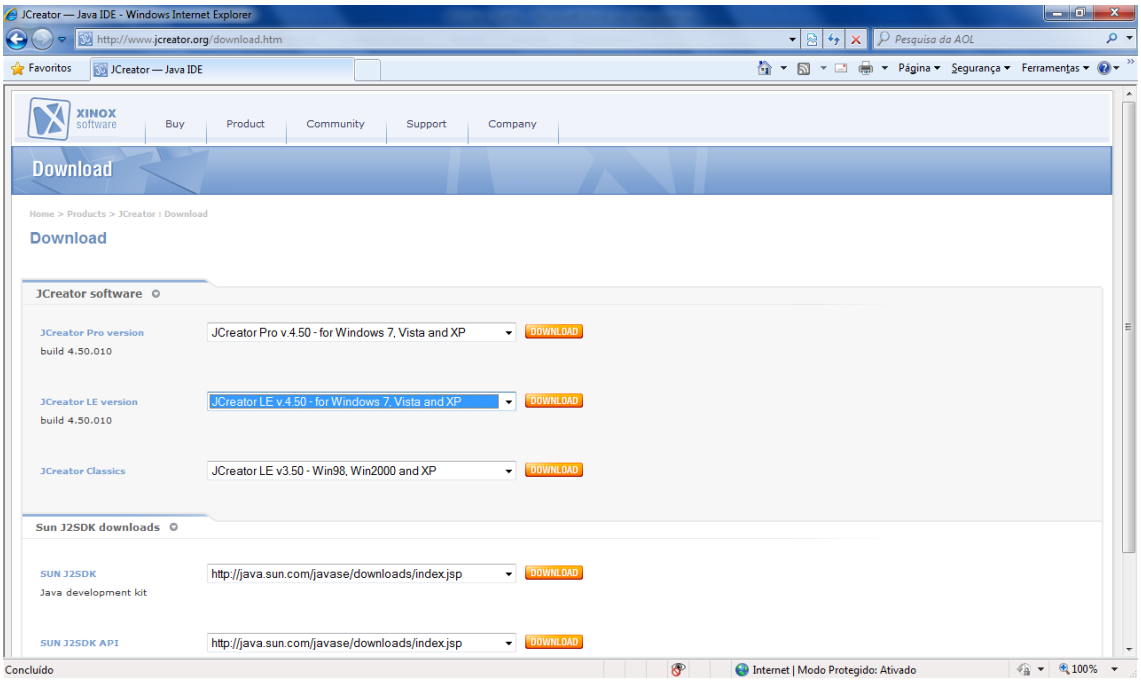


Figura 07 – Download do JCreator LE version

Após ser concluído o download do instalador efetue a instalação do JCreator. Verifique como instalar o JDK e do JCreator nos vídeos abaixo:

- Instalando o JDK
- Instalando o JCreator

Testando o ambiente – a primeira classe

Após a instalação do JCreator abra a IDE e crie um novo arquivo selecionando as seguintes opções do menu: File –> New –> File, como mostra a figura 08.

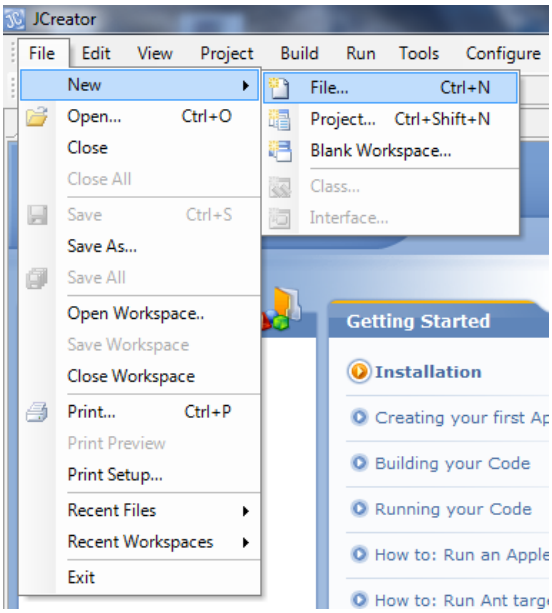


Figura 08 – Criando um novo arquivo no JCreator

Na tela seguinte você deverá selecionar o tipo do arquivo a ser criado. Dentre os modelos existentes, selecione **Java Class** para criarmos nossa primeira classe em Java e dê um clique no botão **Next** (figura 09).

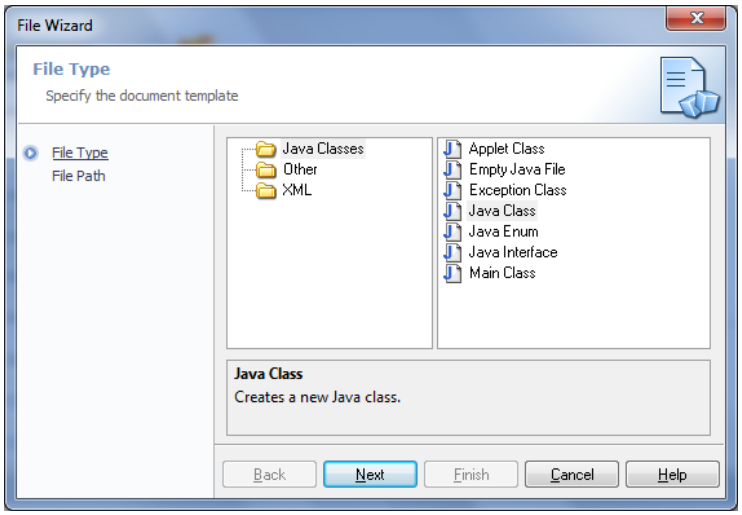


Figura 09 – Escolhendo o tipo do arquivo a ser criado

Em seguida você deverá informar o nome do arquivo e o local onde deseja salvá-lo. Aconselhamos que você crie uma pasta em seu computador para salvar os arquivos gerados ao longo da disciplina. Nomeie o arquivo como “PrimeiroPrograma” dê um clique no botão **Finish** como mostra a figura 10.

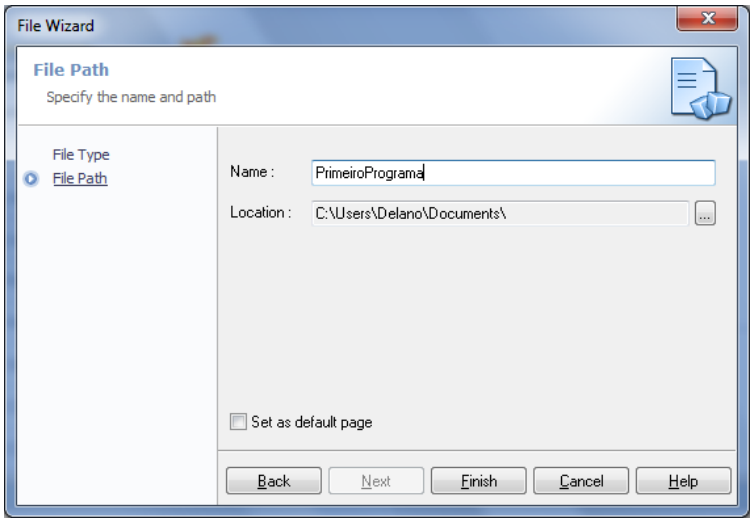


Figura 10 – Nomeando o arquivo a ser criado

Veja que no arquivo recém criado a ferramenta já disponibiliza um modelo com comentários a respeito do código e as estruturas básicas de uma classe (figura 11).

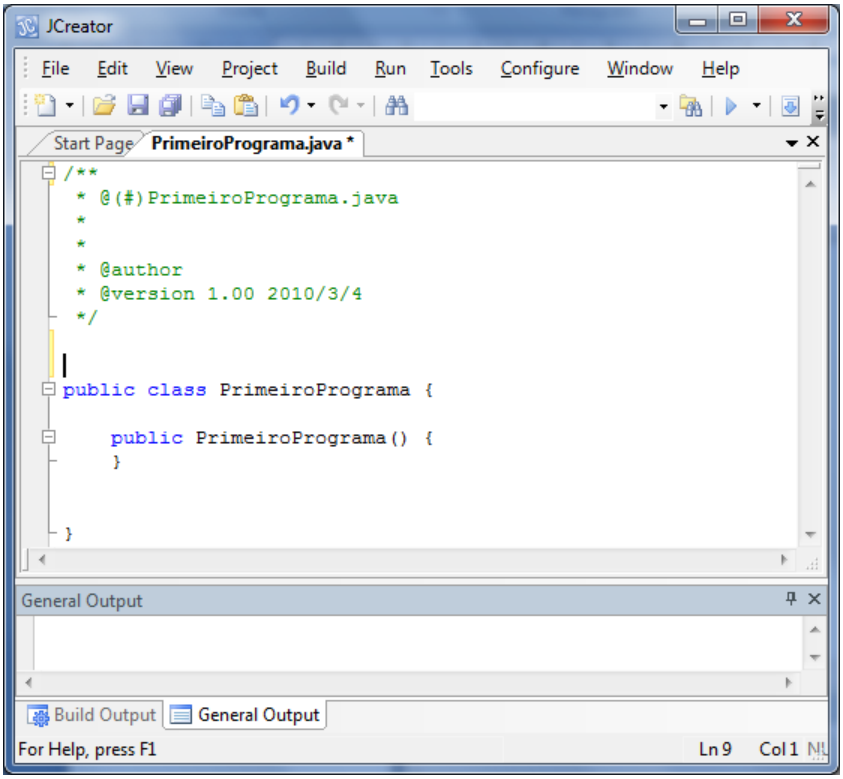
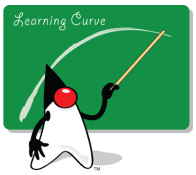


Figura 11 – Nova classe Java criada no JCreator



Comentários em Java são feitos com os caracteres // quando fazem uso de apenas uma linha do código. Quando for necessário fazer uso de mais de uma linha, o texto utilizado pelo comentário deve ser iniciado por /* e finalizado por */.

Apague o texto e digite o código abaixo:

```
public class PrimeiroPrograma{
        public static void main(String args[])  {
                System.ou.println("Meu primeiro programa em Java");
        }
}
```

Para compilar o código e em seguida executar ao arquivo clique nos atalhos indicados na figura 12. O resultado da compilação pode ser visualizado na janela **Build Output**.

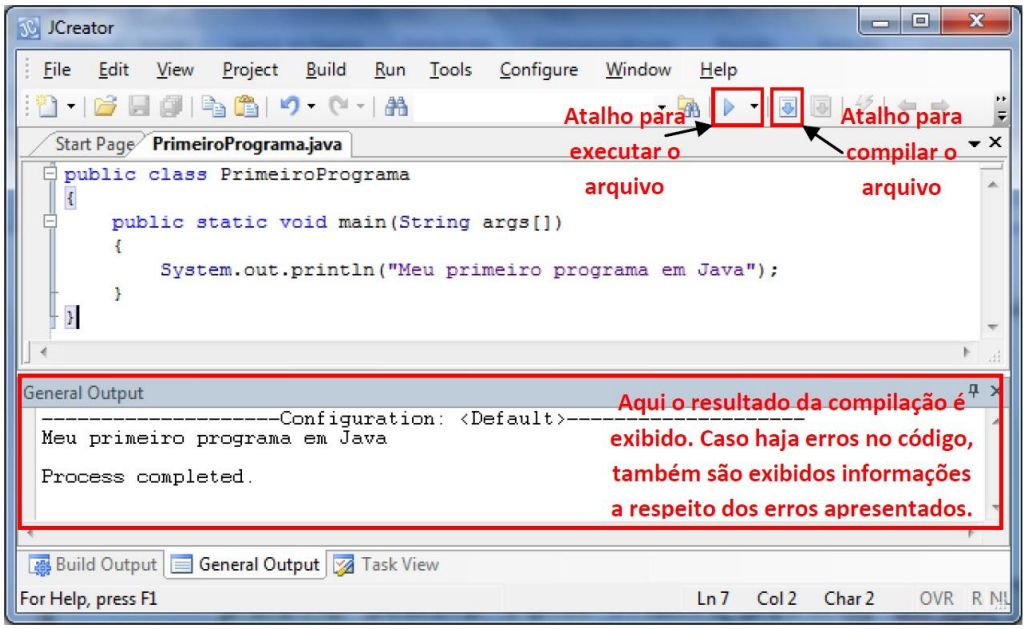
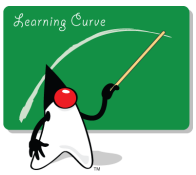


Figura 12 – Compilando e executando o arquivo

Perceba que o código está organizado de forma estruturada, dividido em blocos delimitados por chaves, sendo iniciado por { e finalizado por }.

A instrução *public class PrimeiroPrograma* determina a criação de uma classe pública chamada "PrimeiroPrograma". Todo arquivo deve ter o mesmo nome da classe seguido da extensão *.java*. Logo o nome do arquivo do nosso primeiro exemplo é "PrimeiroPrograma.java".

Atenção! A linguagem Java é case sensitive. Ou seja, há diferenciação de letras maiúsculas e minúsculas. Cuidado ao digitar seus códigos!



Objetivando o entendimento do código Java por outras pessoas adota-se a convenção de sempre iniciar o nome de uma classe com letra Maiúscula. Em casos de nomes compostos, a primeira letra de cada palavra também será maiúscula, como em **PrimeiroPrograma**. Esta é uma convenção, não sendo obrigatório, mas extremamente recomendável por facilitar a leitura e entendimento do código.

A instrução seguinte *public static void main(String args[])* declara o método **main**. Está é a declaração mais comum e utilizada de uma classe Java. Este é o método principal de uma classe e é o primeiro bloco a ser executado pela Máquina Virtual Java. É aqui que o processamento de uma aplicação Java tem início. Obrigatoriamente este método deve:

- Ser público (*public*);
- Estático (*static*);

- Não retornar valores (*void*);
- Possuir um parâmetro array do tipo *String* que por padrão é chamado de *args*.

A última instrução de nosso primeiro exemplo exibe uma mensagem. A instrução *System.ou.println()* responsável por exibir esta mensagem deve receber do tipo *String*, um valor textual, ou um texto entre aspas duplas, como em nosso exemplo.

Faça modificações no código do exemplo apresentado e verifique os resultados. Explore a interface da IDE.

Mais a frente, iremos apresentar com mais detalhes os conceitos de classes e métodos. O objetivo aqui é se familiarizar com o ambiente de desenvolvimento e a codificação em Java.