

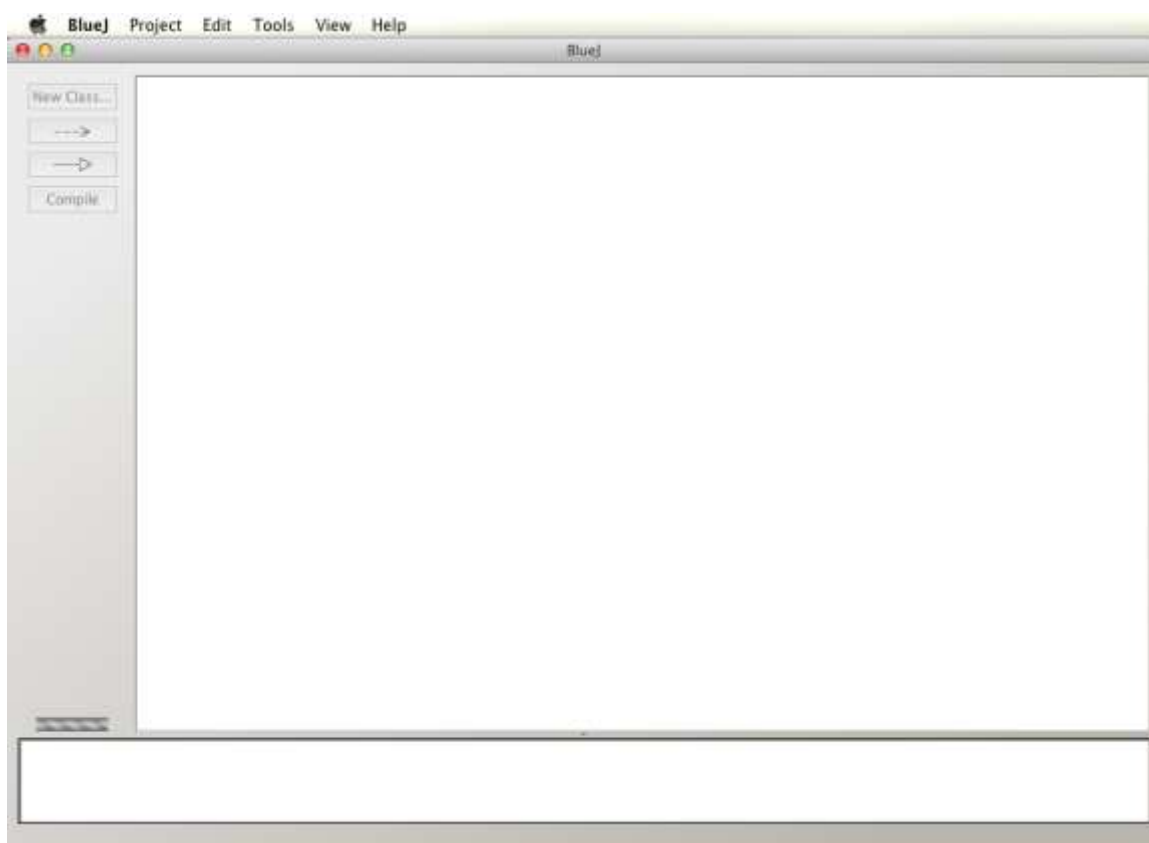
MÓDULO 01 – PRÁTICA DE LABORATÓRIO I

1 INTRODUÇÃO AO BLUEJ

O BlueJ é um IDE (Integrated **Development Environment**), ou seja, um ambiente para desenvolvimento de programas na linguagem Java. Você pode fazer o download do BlueJ, gratuitamente, no site: <http://bluej.org>.

Este ambiente foi desenvolvido como um projeto das Universidades: University of Kent (Canterbury, Inglaterra) e Deakin University (Melbourne, Austrália). O projeto é suportado pela Sun Microsystems. Para usar o BlueJ é preciso instalar também o Java SE Development Kit (JDK) que pode ser baixado no site: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

A tela inicial do BlueJ é a seguinte:



Para iniciar o desenvolvimento de uma aplicação, você deve criar um novo Projeto. Faça isto pelo menu **Project >> New Project**.

Todas as classes java criadas em um determinado projeto serão armazenadas num diretório de mesmo nome que o projeto criado com a extensão .java. Portanto, quando realizar a criação de um novo projeto, lembre-se de selecionar um diretório apropriado.

A finalidade deste módulo é a ambientalização com a ferramenta de desenvolvimento BlueJ (que será utilizado durante todo o curso) e um primeiro contato com a linguagem de programação Java.

Nos tópicos a seguir **estaremos praticando os conceitos visto no módulo 1 de Programação I**. Revise estes conceitos nos dois primeiros capítulos do livro de apoio de Programação I:

- Capítulo 01 - INTRODUÇÃO À PROGRAMAÇÃO;

- Capítulo 02 - ELEMENTOS BÁSICOS.

Na sequência, siga as instruções deste documento e tire suas dúvidas com o tutor no fórum de dúvidas.

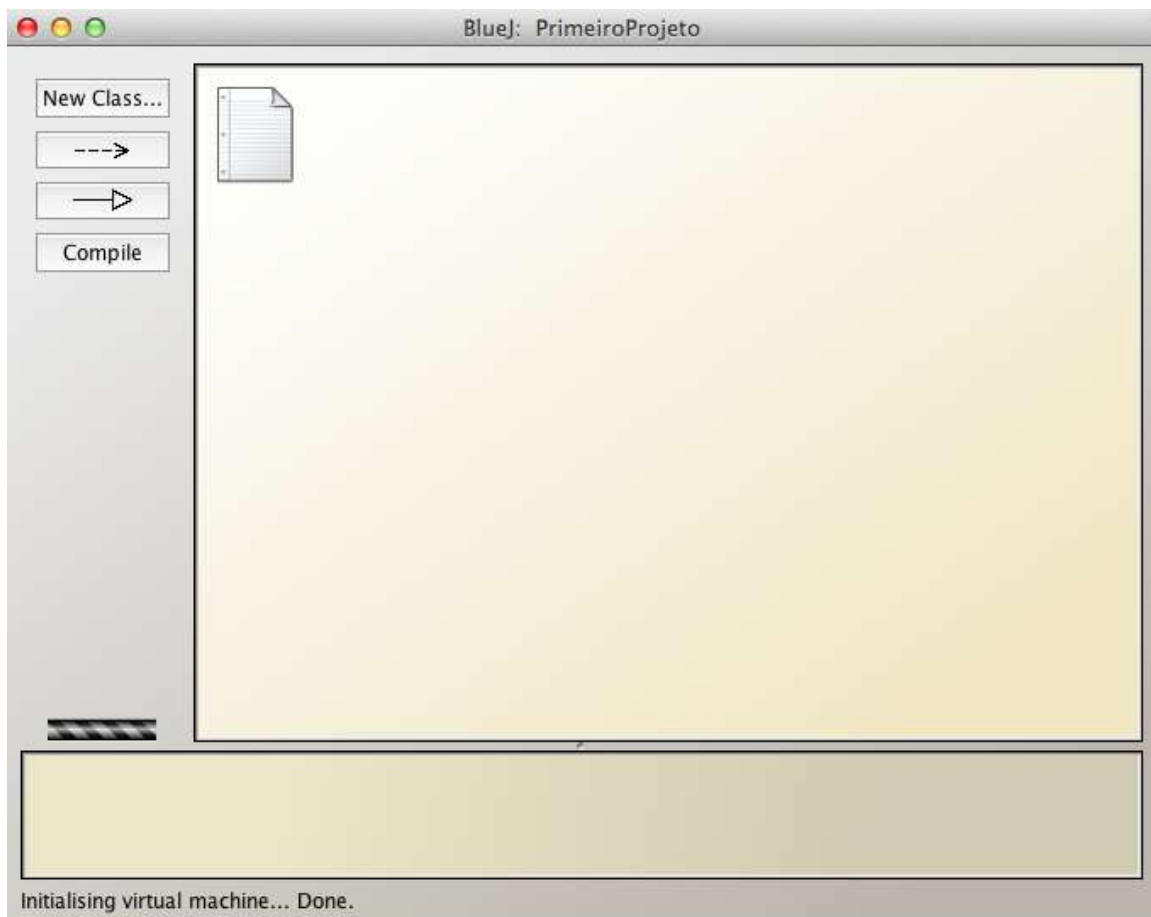
2 CONHECENDO O BLUEJ

Vamos iniciar a prática com o uso da ferramenta BlueJ. Para tanto, siga os passos indicados abaixo. Todos os itens que estaremos utilizando nesta prática serão aprofundados durante os módulos do nosso curso. Portanto, não se preocupe em compreender todos os conceitos neste momento.

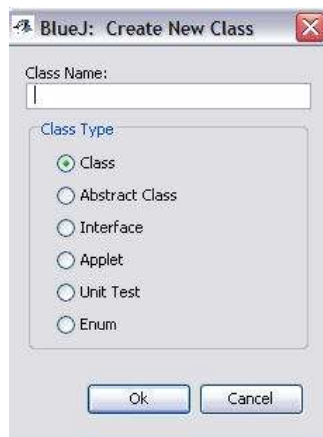
Lembre-se que o objetivo deste primeiro módulo é a introdução a ferramenta que estaremos utilizando.

2.1 Criando o Primeiro Projeto

Crie um projeto chamado PrimeiroProjeto. Faça isto pelo menu **Project >> New Project**. Ao criar um novo projeto, você verá a seguinte tela:



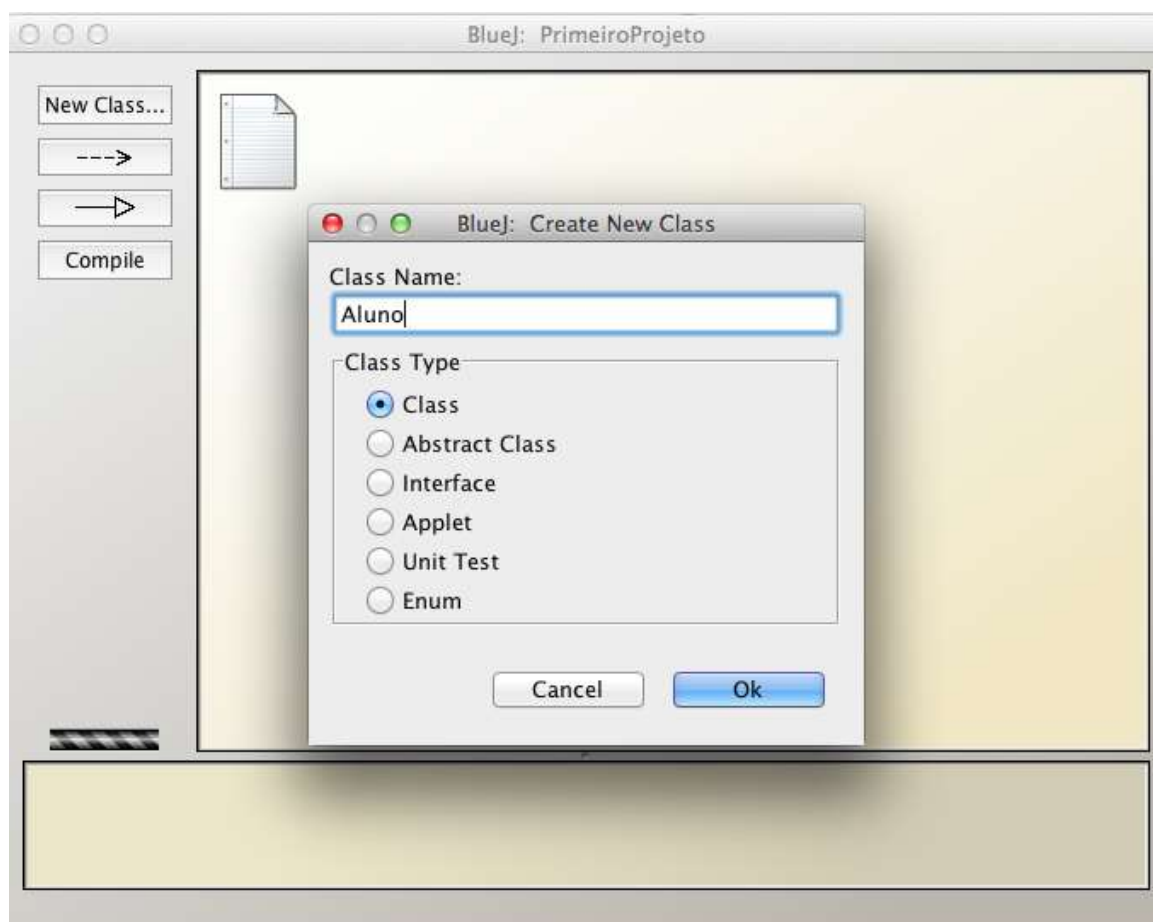
Para criar uma nova classe, use o botão **New Class**. Ao clicar neste botão, o BlueJ apresentará a seguinte janela:



Nesta janela, você deve definir o nome da nova classe java que está criando. Para a criação de cada classe, você deverá realizar o mesmo procedimento.

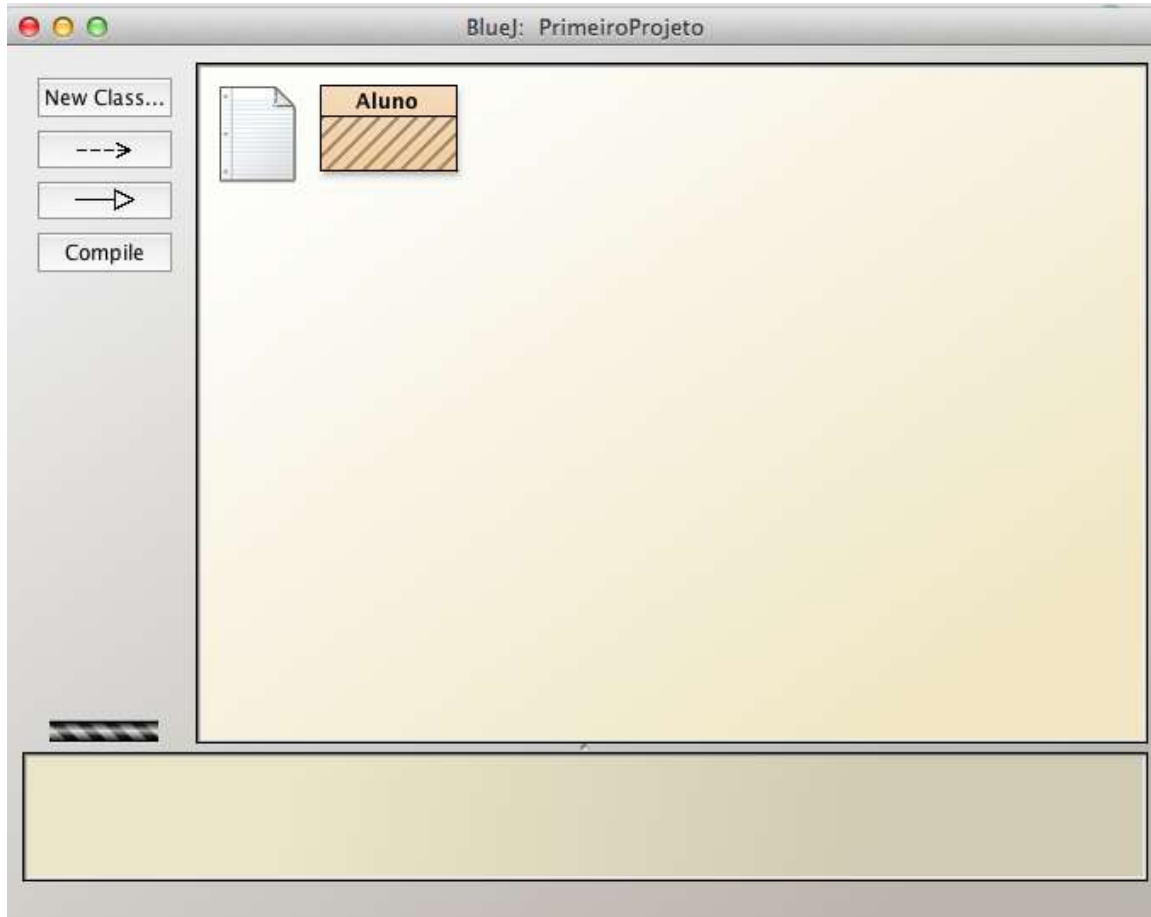
2.2 Criando a Primeira Classe

Crie uma classe chamada Aluno. Deixe a opção: **Class** marcada e clique em OK.

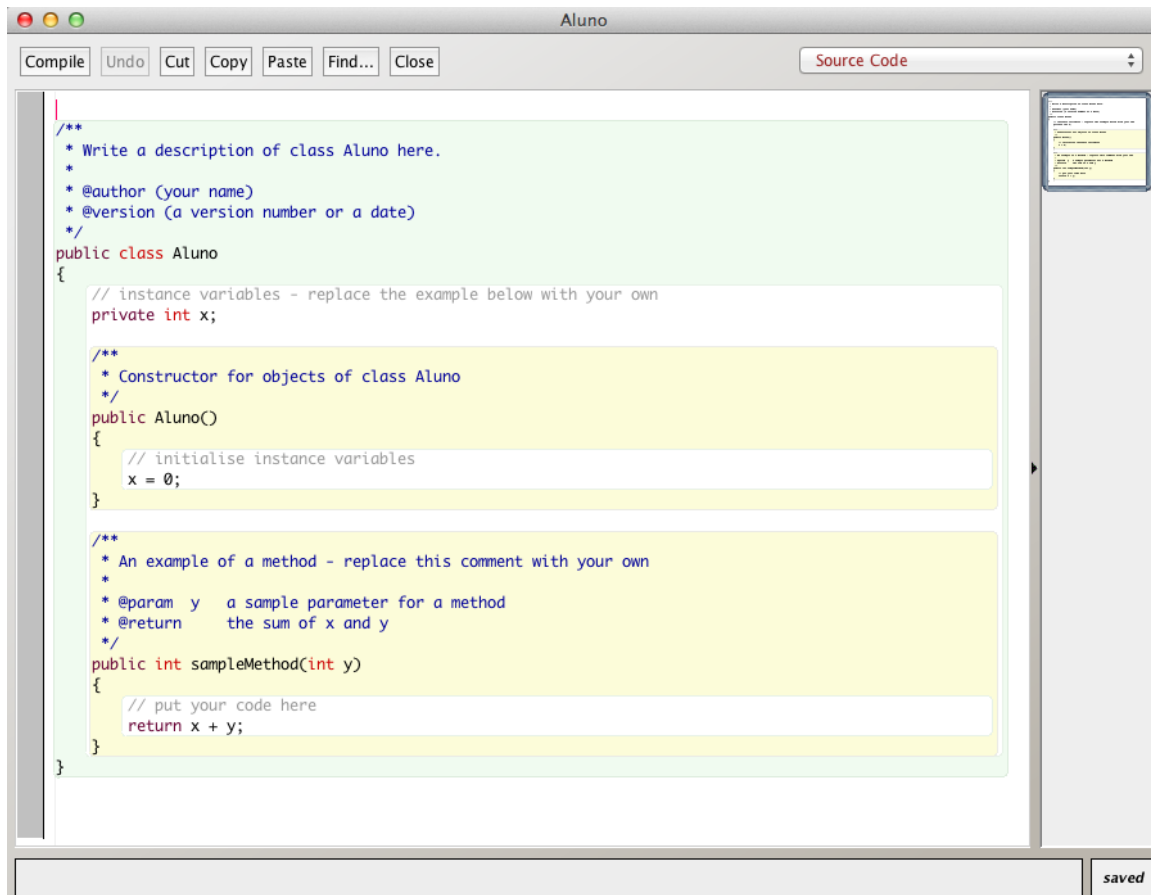


Um retângulo com a classe Aluno aparecerá na janela principal do BlueJ.

Clique com o botão direito do mouse duas vezes no ícone da classe:



Será aberta uma janela de edição, onde você verá o seguinte código:



Como você pode observar, o BlueJ gera automaticamente um template com um código padrão para qualquer classe criada.

Apague todo o código que está na janela do editor, e digite o seguinte código:

```
/**
 * Classe Aluno
 * @author Rosemary Francisco
 * @version 02/03/2012
 */
public class Aluno
{
    // atributos da classe
    private int matricula;
    private String nome;
    private int semestre;

    /**
     * Método Construtor
     */
    public Aluno(int mat, String nom)
    {
        matricula = mat;
        nome = nom;
        semestre = 1;
    }

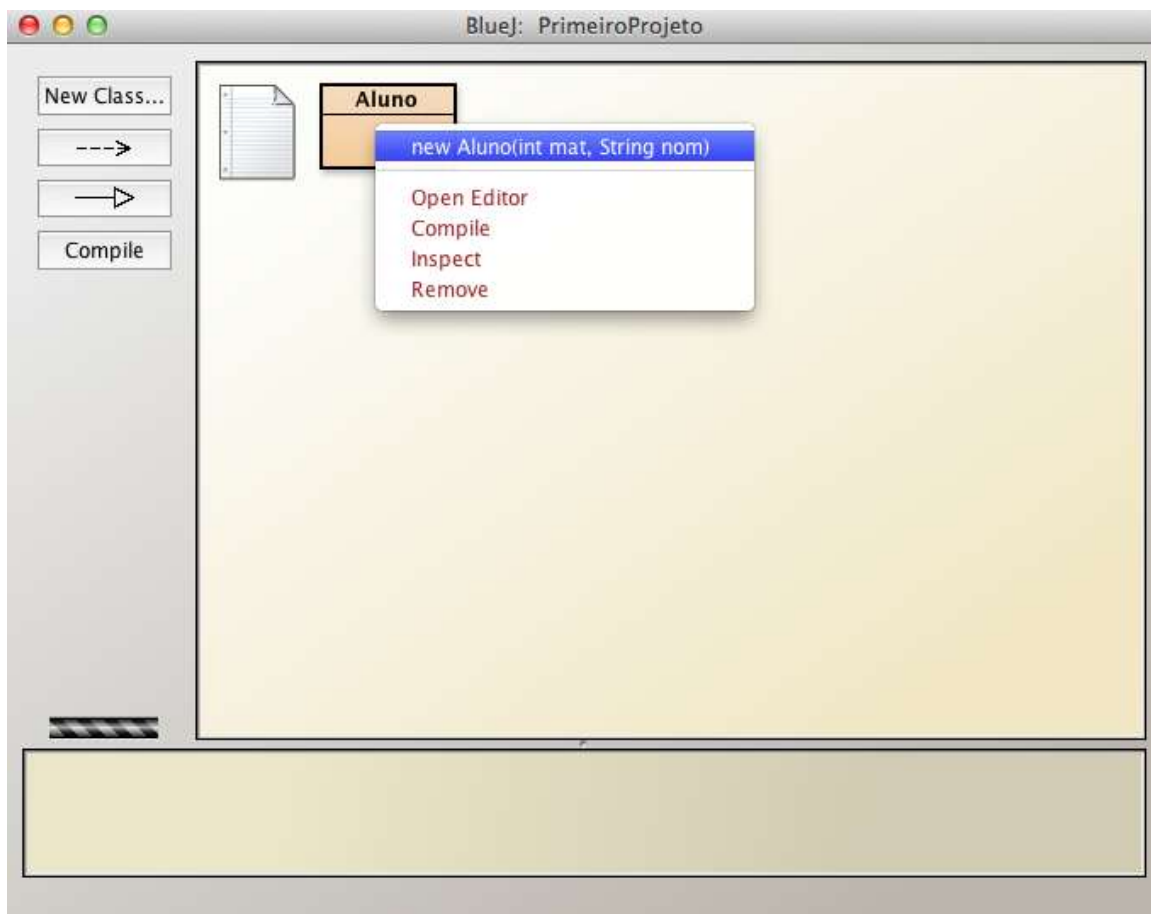
    /**
     * Método aprovado
     * Função: aprova o aluno e altera o semestre do aluno para o próximo semestre
     */
    public void aprovado()
    {
        semestre = semestre + 1;
    }

    /**
     * Método exibeDados
     * Função: exibe os dados do aluno
     */
    public void exibeDados()
    {
        System.out.println("Matrícula: " + matricula);
        System.out.println("Nome: " + nome);
        System.out.println("Semestre: " + semestre);
    }
}
```

Após concluir a digitação do código da classe, compile a mesma através do botão Compile localizado no topo da janela de edição. Pronto, sua primeira classe foi criada!

2.3 Criando Objetos com a Classe Aluno criada

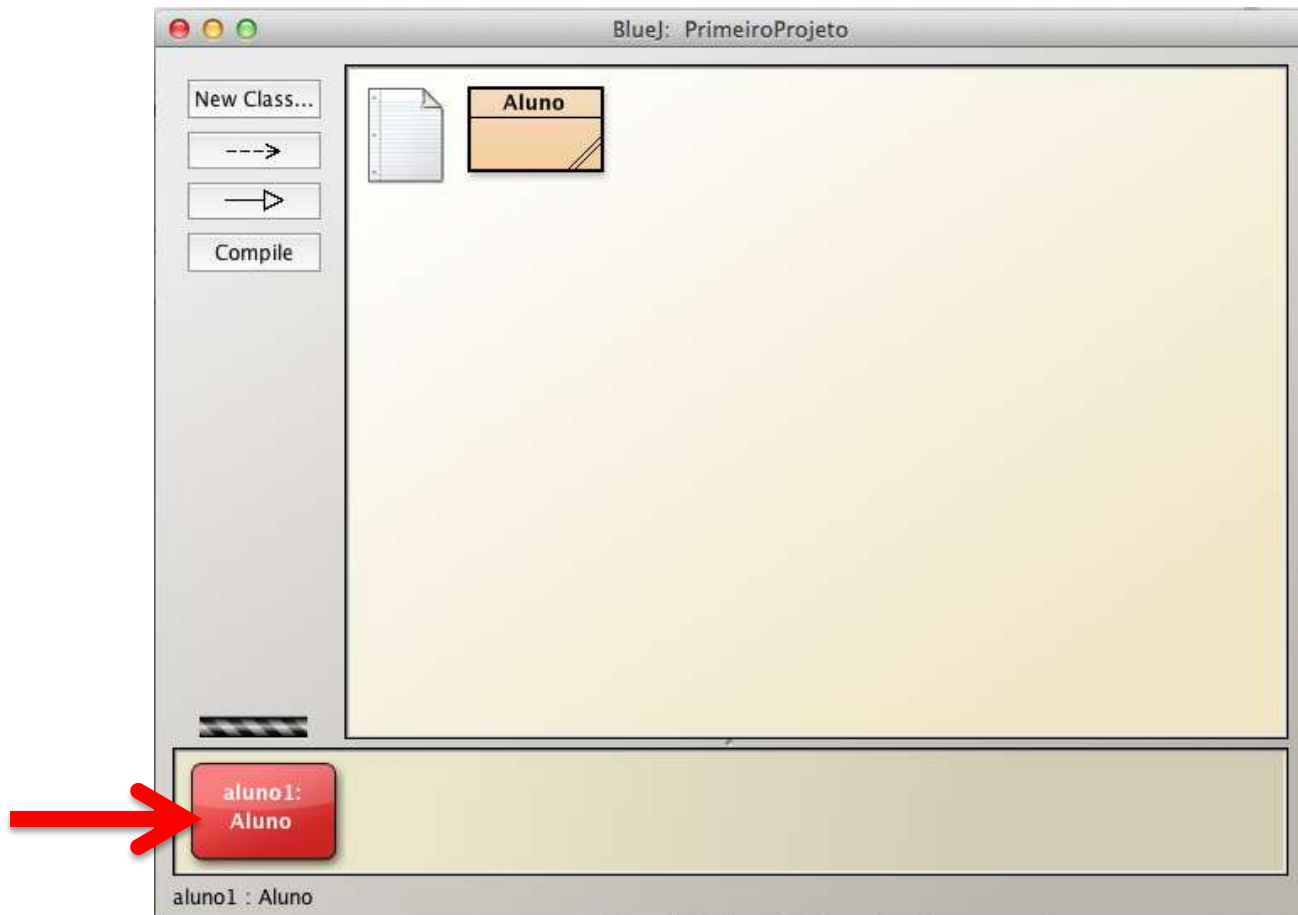
Clique com o botão direito do mouse no ícone da classe Aluno e escolha a opção new Aluno (int mat, String nom) para criar (instanciar) um aluno (ou seja, criar um objeto da classe Aluno).



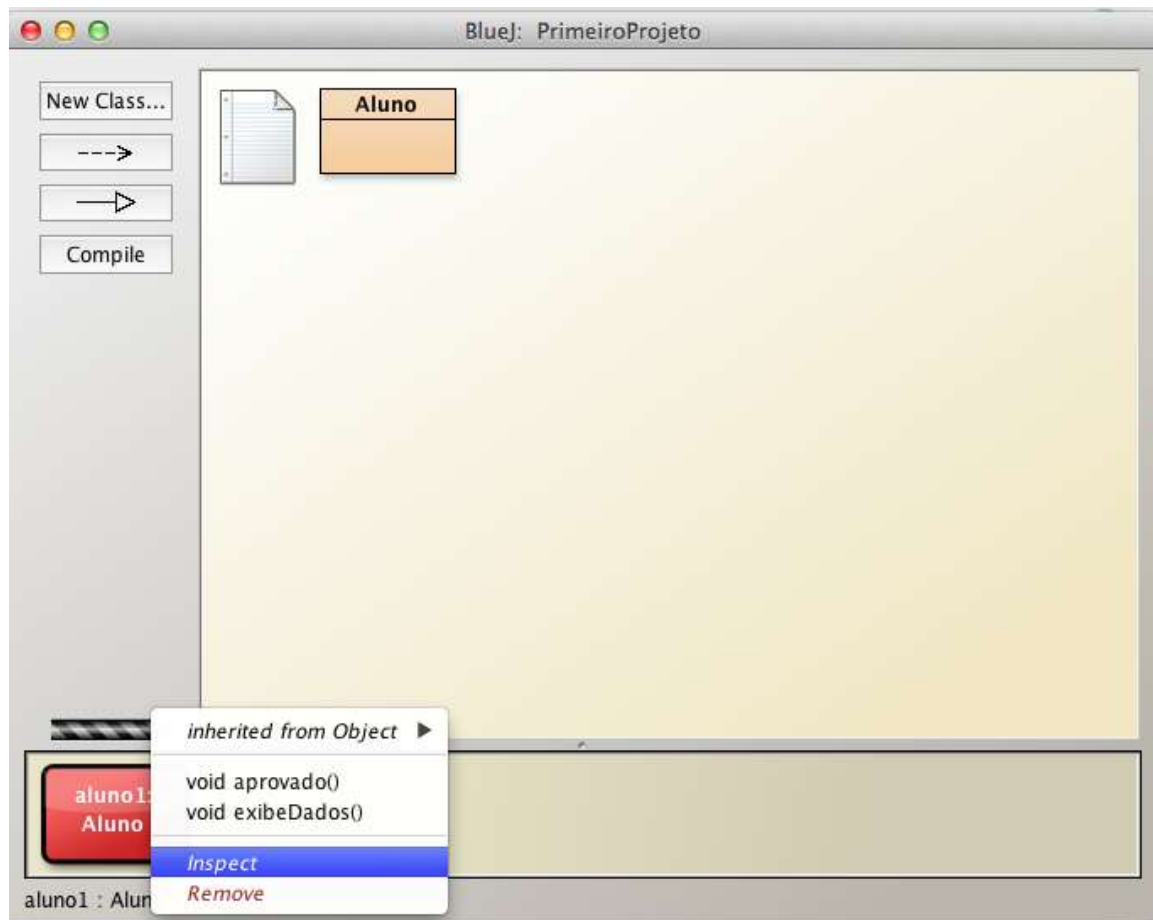
Digite a matrícula e o nome conforme a janela abaixo e clique em OK:

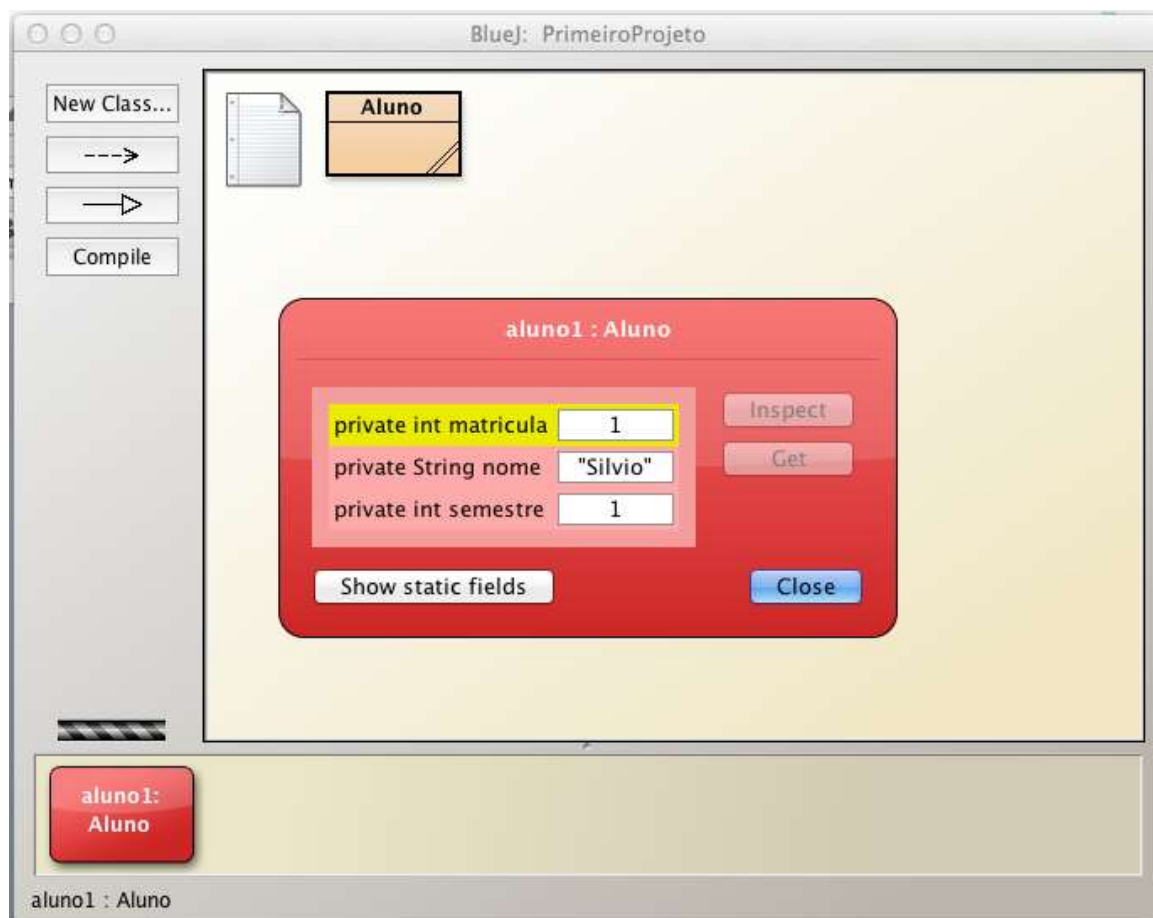


Após clicar no botão: OK, o objeto será criado, conforme o ícone em vermelho abaixo:



Clique com o botão direito do mouse no ícone do objeto e escolha a opção: **Inspect** e então você poderá observar o estado interno do objeto, conforme a imagem abaixo:

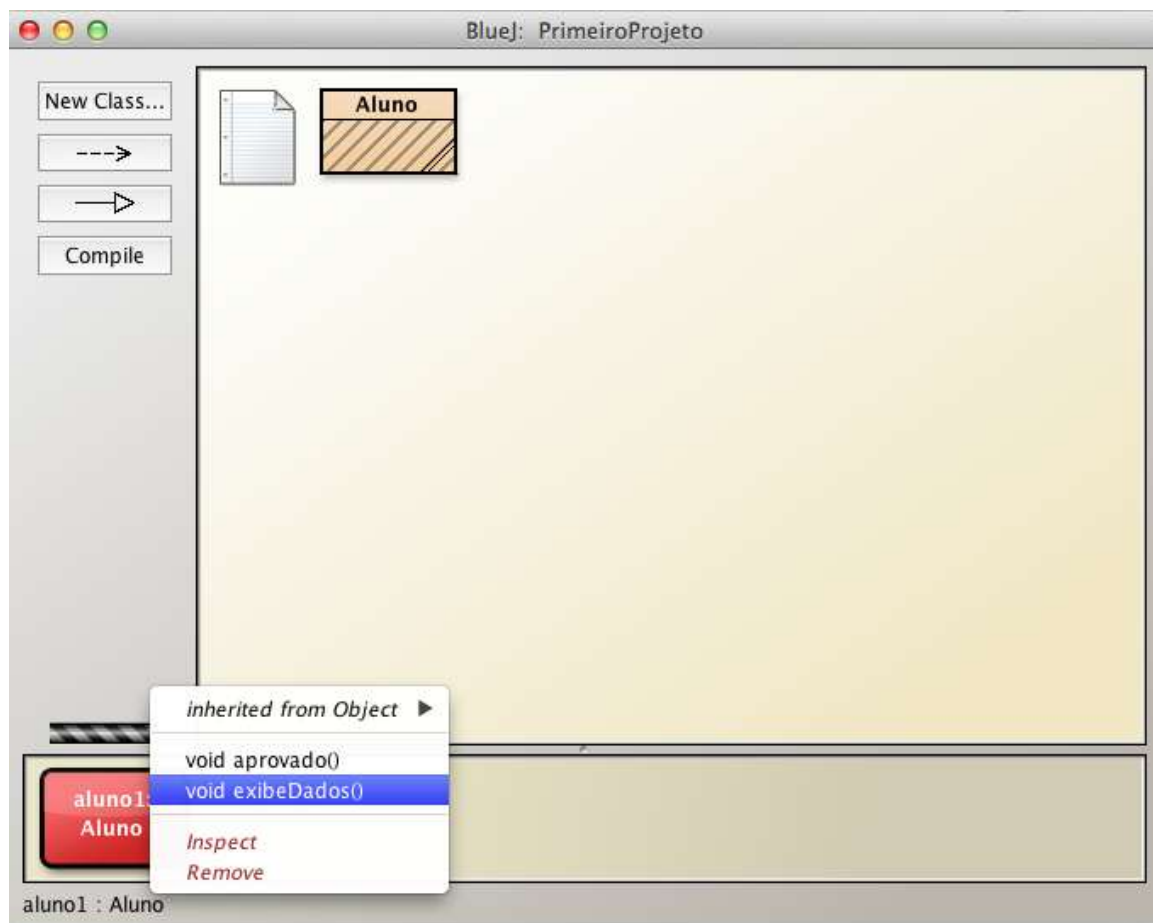




2.4 Executando os métodos do objeto Aluno

Clique com o botão direito do mouse no ícone do objeto e selecione as outras opções, conforme o exemplo abaixo. Observe o que acontece com os dados do objeto, por meio da opção: Inspect. Você vai perceber que haverá alterações no estado interno do objeto.

Veja abaixo o exemplo da execução do método: `exibeDados()`.





2.5 TRABALHANDO COM OS ELEMENTOS BÁSICOS

Neste tópico vamos praticar os elementos básicos: **Tipos de Dados**, **Constantes** e **Variáveis**. Para tanto, vamos primeiro revisar resumidamente o que significa cada elemento. **Atenção:** Para verificar mais detalhes sobre estes conceitos, revise o Capítulo 02 do livro de apoio de Programação I.

Vimos em Programação I os conceitos referente a Dados e Informação. Em um programa de computador utilizamos os Dados para realizar operações. Porém, para podermos realizar operações com os Dados, precisamos dizer ao compilador Java, por meio de comandos na nossas classes, quais **Tipos de Dados** estaremos utilizando.

Vamos então relembrar quais os Tipos de Dados que podemos utilizar:

- Números Inteiros = permitem armazenar e realizar operações com números inteiros. Em Java temos os **tipos de dados primitivos** abaixo para números inteiros. A diferença entre estes tipos é a capacidade dos números que serão utilizados.
 - byte;
 - short;
 - int;
 - long.
- Números com casas decimais = permitem armazenar e realizar operações com números reais, números com casas decimais. Em Java temos os **tipos de dados primitivos** abaixo

para números com casas decimais. A diferença entre estes tipos é a capacidade dos números que serão utilizados.

- float;
- double.
- Lógico ou booleano = permitem armazenar e realizar operações com valor lógico: TRUE/FALSE ou Verdadeiro/Falso em português. Em Java temos o **tipo de dado primitivo** abaixo para valores lógicos ou booleanos.
 - boolean.
- Caracter ou literal = permitem armazenar e realizar operações com caracteres. Em Java temos o **tipo de dado primitivo** abaixo para caracteres.
 - char.
- Sequencia de caracteres = permitem armazenar e realizar operações com sequencia de caracteres. Em Java temos somente um **tipo de dado objeto** para permitir ações com sequencia de caracteres. O tipo de dados é o:
 - String = por ser um tipo de dado objeto, inicia com a letra maiúscula.

Importante: Java é *case-sensitive*, isto significa que o Java faz diferença entre letras maiúsculas e minúsculas. Portanto, sempre temos que escrever os **tipos de dados primitivos com letras minúsculas** e o **tipo de dado objeto: String com o S maiúsculo**.

Como comentado acima, utilizamos estes tipos de dados para realizar operações nos dados. Porém, para isso, precisaremos fazer uso do Tipo de Dados e das Constantes ou Variáveis. Vamos relembrar o que são Constantes e Variáveis:

- **Constantes** = permitem armazenar determinado valor no programa. Estes valores são imutáveis, isto significa que os valores são definidos uma única vez e depois não poderão ser alterados.
 - As constantes são sempre de algum de tipo de dado primitivo ou de objeto;
 - Sempre possuem um nome;
 - De acordo com uma convenção (padrões comunidade Java) utiliza-se o nome todo em maiúsculo. Caso o nome da constante seja um nome composto, cada nome deve ser separado por um “_”, por exemplo:
 - 1 nome = IDADE;
 - 2 nomes (nome composto) = IDADE_MINIMA;
 - 3 nomes (nome composto) = IDADE_MINIMA_ACESSO.
 - Utilizam a palavra final antes do nome, para indicar que é uma Constante;
 - Exemplo de declaração de constantes:

```
final [TipoDeDado] [NOME_DA_CONSTANTE];
```

```
final int IDADE_MINIMA;
```

- **Variáveis** = permitem armazenar determinado valor no programa. Estes valores podem ser alterados a qualquer momento, por isso o nome: variável, pois o valor é variável.
 - As variáveis são sempre de algum de tipo de dado primitivo ou de objeto;
 - Sempre possuem um nome;
 - De acordo com uma convenção (padrões comunidade Java) inicia-se sempre o nome com letras minúsculas. Caso o nome da variável seja um nome composto, a primeira letra do segundo nome será sempre com letra maiúscula, por exemplo:

- 1 nome = idade;
- 2 nomes (nome composto) = idadeMinima;
- 3 nomes (nome composto) = idadeMinimaAcesso.
- Exemplo de declaração de variáveis:

[TipoDeDado] [nomeDaVariavel];

int idade;

Agora que já relembramos os Tipos de Dados as Constantes e Variáveis, vamos praticá-los em uma classe de teste. Vamos criar uma classe de teste que apresenta algumas informações na tela para o usuário.

Imagine a seguinte situação, você está em um parque de diversão e para entrar em algum brinquedo, você tem que passar um cartão para acessar a fila do brinquedo. No momento em que o cartão é passado na máquina de acesso ao brinquedo, é apresentada a seguinte informação para você:

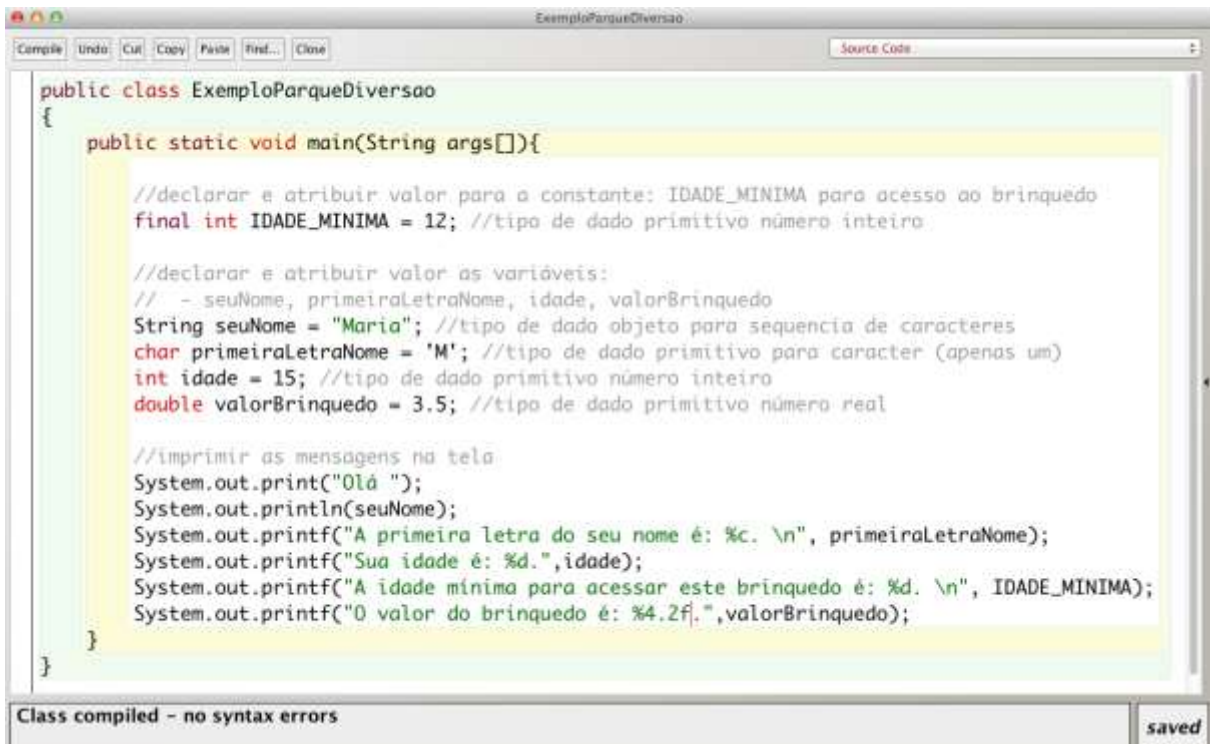
Olá [seuNome];

A primeira letra do seu nome é: [primeiraLetraNome];

Sua idade é: [idade]. A idade mínima para acessar este brinquedo é: [IDADE_MINIMA].

O valor do brinquedo é: [valorBrinquedo].

Vamos então criar uma classe de teste chamada: ExemploParqueDiversao conforme a imagem abaixo:



```

public class ExemploParqueDiversao
{
    public static void main(String args[]){

        //declarar e atribuir valor para a constante: IDADE_MINIMA para acesso ao brinquedo
        final int IDADE_MINIMA = 12; //tipo de dado primitivo número inteiro

        //declarar e atribuir valor as variáveis:
        // - seuNome, primeiraLetraNome, idade, valorBrinquedo
        String seuNome = "Maria"; //tipo de dado objeto para sequencia de caracteres
        char primeiraLetraNome = 'M'; //tipo de dado primitivo para caracter (apenas um)
        int idade = 15; //tipo de dado primitivo número inteiro
        double valorBrinquedo = 3.5; //tipo de dado primitivo número real

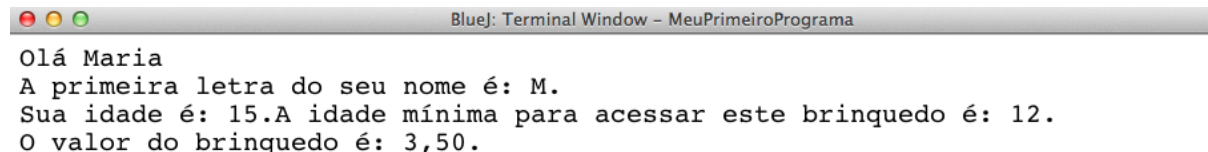
        //imprimir as mensagens na tela
        System.out.print("Olá ");
        System.out.println(seuNome);
        System.out.printf("A primeira letra do seu nome é: %c. \n", primeiraLetraNome);
        System.out.printf("Sua idade é: %d.", idade);
        System.out.printf("A idade mínima para acessar este brinquedo é: %d. \n", IDADE_MINIMA);
        System.out.printf("O valor do brinquedo é: %4.2f.", valorBrinquedo);

    }
}

```

Class compiled - no syntax errors

Ao testarmos a classe, teremos o seguinte resultado:



```
BlueJ: Terminal Window - MeuPrimeiroPrograma
Olá Maria
A primeira letra do seu nome é: M.
Sua idade é: 15.A idade mínima para acessar este brinquedo é: 12.
O valor do brinquedo é: 3,50.
```

Atenção: Para que você possa comparar a sua classe, com a classe desenvolvida neste exemplo, faça o download deste projeto que está disponível para download no módulo 01.

2.6 TRABALHANDO COM AS EXPRESSÕES

Neste tópico vamos praticar algumas das expressões que podem ser realizada nos dados. **Atenção:** Para verificar mais detalhes sobre as expressões, revise o Capítulo 02 do livro de apoio de Programação I.

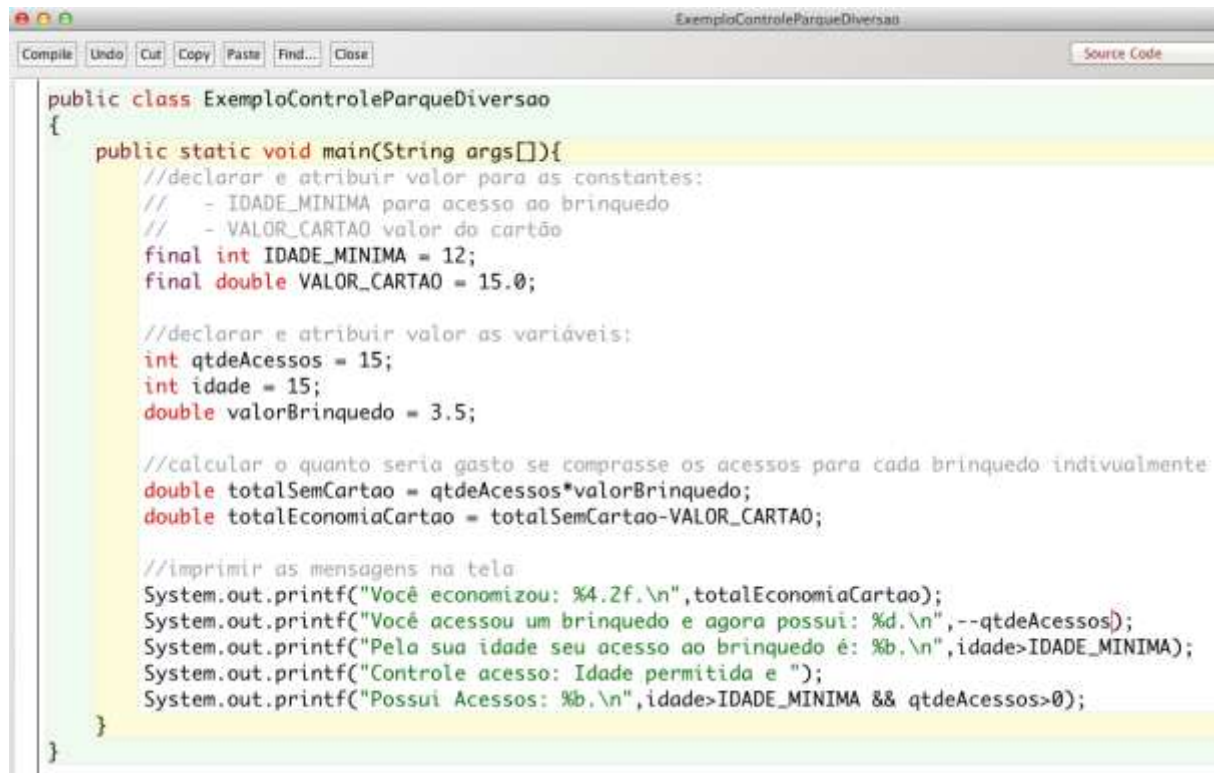
Nos exemplos anteriores nós já trabalhamos com o operador de atribuição “=”. Neste exemplo, estaremos praticando algumas expressões com os **Operadores Aritméticos**, **Operadores Relacionais**, **Operadores Lógicos** e **Operadores Unários**. Para isso estaremos utilizando novamente o cenário do Parque de Diversão.

Quando você foi registrado pelo atendente do parque, você ganhou 15 acessos para qualquer brinquedo. Esta quantidade foi definida no momento que você realizou a compra do cartão.

Como cada acesso a um brinquedo custa 3,50, e você pagou 15,00 pelo cartão, precisaremos realizar alguns cálculos para identificar quanto você economizou com a compra do cartão.

Outra situação que temos que controlar é a quantidade de acessos que você ainda possui depois de entrar em um brinquedo. Além disso, precisamos também verificar se você possui a idade mínima para acessar o brinquedo que deseja utilizar.

Para praticarmos estas situações, vamos criar a classe: ExemploControleParqueDiversao conforme imagem abaixo:



```

public class ExemploControleParqueDiversao
{
    public static void main(String args[]){
        //declarar e atribuir valor para as constantes:
        // - IDADE_MINIMA para acesso ao brinquedo
        // - VALOR_CARTAO valor do cartão
        final int IDADE_MINIMA = 12;
        final double VALOR_CARTAO = 15.0;

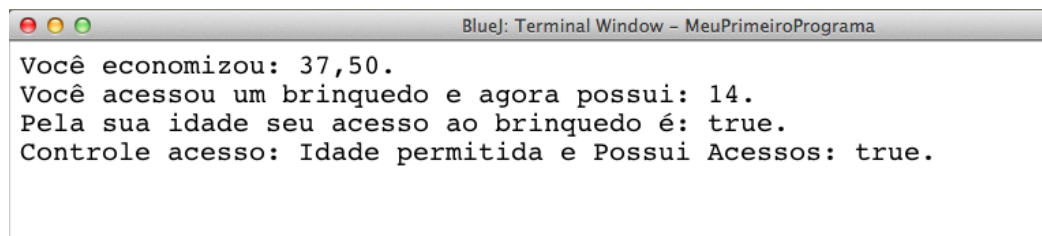
        //declarar e atribuir valor as variáveis:
        int qtdeAcessos = 15;
        int idade = 15;
        double valorBrinquedo = 3.5;

        //calcular o quanto seria gasto se comprasse os acessos para cada brinquedo individualmente
        double totalSemCartao = qtdeAcessos*valorBrinquedo;
        double totalEconomiaCartao = totalSemCartao-VALOR_CARTAO;

        //imprimir as mensagens na tela
        System.out.printf("Você economizou: %4.2f.\n",totalEconomiaCartao);
        System.out.printf("Você acessou um brinquedo e agora possui: %d.\n",--qtdeAcessos);
        System.out.printf("Pela sua idade seu acesso ao brinquedo é: %b.\n",idade>IDADE_MINIMA);
        System.out.printf("Controle acesso: Idade permitida e ");
        System.out.printf("Possui Acessos: %b.\n",idade>IDADE_MINIMA && qtdeAcessos>0);
    }
}

```

Ao executarmos a classe, teremos o seguinte resultado:



```

BlueJ: Terminal Window - MeuPrimeiroPrograma

Você economizou: 37,50.
Você acessou um brinquedo e agora possui: 14.
Pela sua idade seu acesso ao brinquedo é: true.
Controle acesso: Idade permitida e Possui Acessos: true.

```

Atenção: Para que você possa comparar a sua classe, com a classe desenvolvida neste exemplo, faça o download deste projeto que está disponível para download no módulo 01.