

# Programação Orientada a Objetos

*profº Mauricio Conceição Mario*

# Conteúdo Programático

- Princípios de Programação Orientada a Objetos:
  - Abstração
- Classes, métodos e variáveis de instância: Objetos
- Encapsulamento
- Métodos construtores
- Coleta de lixo
- Métodos set e get
- Herança
- Polimorfismo
- Classes Abstratas e Interfaces
- Subtipagem
- Tratamento de exceções

# Critério de Avaliação

*1º bimestre:*

$$\text{Média 1} = 0,4 * \text{trabalho} + 0,6 * \text{prova 1}$$

*2º bimestre:*

$$\text{Média 2} = 0,4 * \text{trabalho} + 0,6 * \text{prova 2}$$

$$\text{Média final} = (\text{Média 1} + \text{Média 2}) / 2$$

*prova 3: substitui a prova1 ou prova2*

# Referências Bibliográficas

- Java 7 - Ensino Didático  
Sérgio Furgeri - Editora Érica
- Java - como Programar - 8ª edição  
H. M. Deitel & P. J. Deitel / Ed. Bookman
- Programação Orientada a Objeto - Referências: Estruturas de Dados & Algoritmos em Java.  
Goodrich, M. T. e Tamassia R.. Editora Bookman, Porto Alegre - 2013.
- Modelagem de Sistemas Orientados a Objetos – Ensino Didático  
Sérgio Furgeri - Editora Érica
- Programação Orientada a Objetos em Java  
Isaias Camilo Boratti – Editora Visual Books ISBN 85-7502-199-0.  
Florianópolis, 2007

## *Programação Orientada a Objetos*

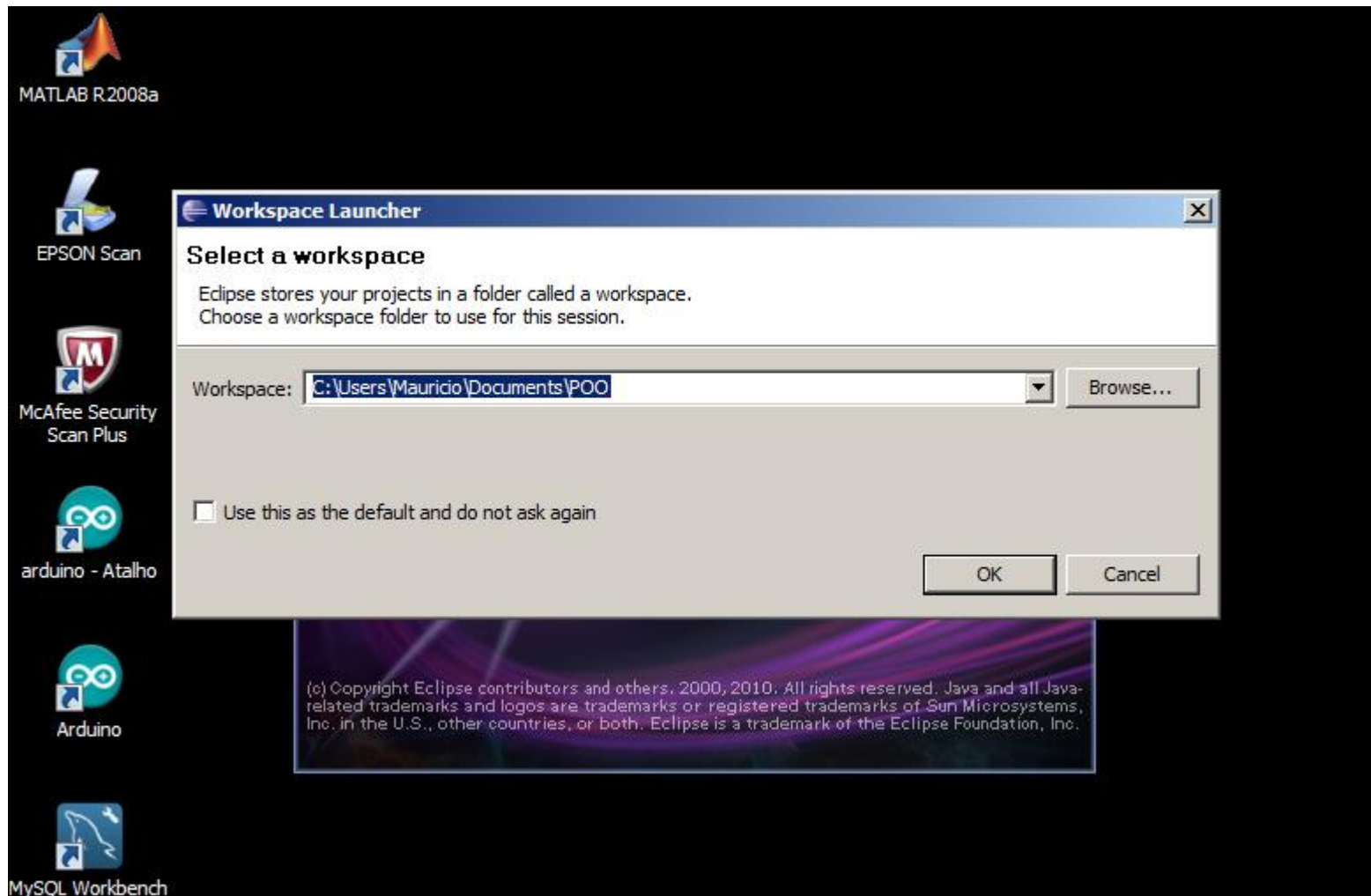
**Abstração:** genericamente abstração é o processo de análise de determinada situação do mundo real, através do qual se determinam os aspectos e fenômenos considerados essenciais àquela situação, excluindo os aspectos irrelevantes (Boratti, 2007).

Em programação orientada a objetos abstração é a decomposição de um sistema complexo em suas partes fundamentais, descrevendo-o em linguagem simples e precisa. A descrição das partes de um sistema consiste em atribuir-lhes nomes e descrever suas funcionalidades (Goodrich & Tamassia, 2013).

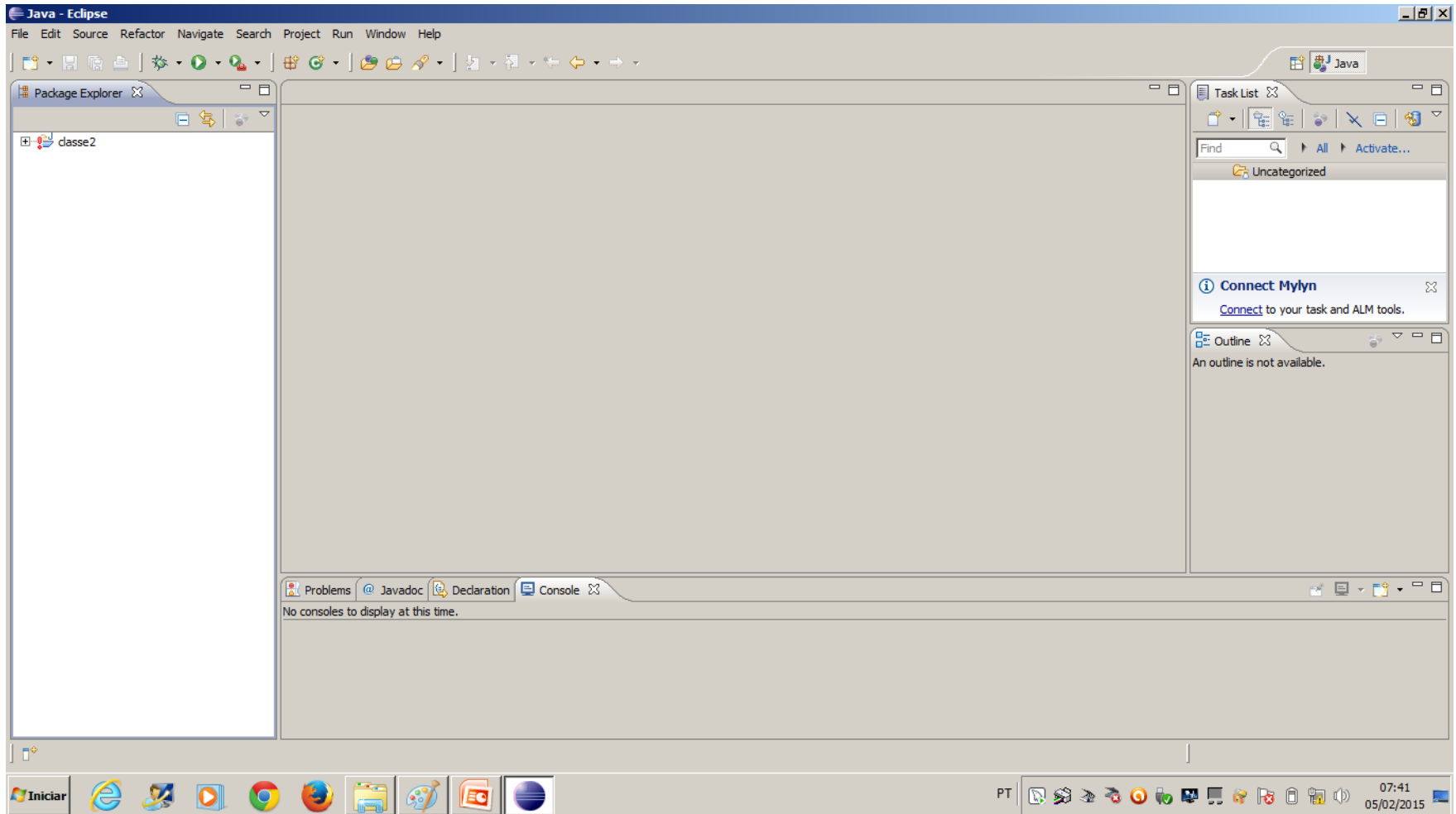
# Projeto Java no Eclipse

- Java: [www.sun.com](http://www.sun.com) (redireciona para página da *Oracle*)
- Fazer download de versão compatível com o sistema operacional →
- Windows ou Linux, 32 ou 64 bits; instalar
- Fazer download da IDE Eclipse: [www.eclipse.org](http://www.eclipse.org) → instalar
- Criar pasta onde será desenvolvido o projeto Java;
- Abrir o Eclipse, redirecionando através do browse, para a pasta criada:

# Projeto no Eclipse

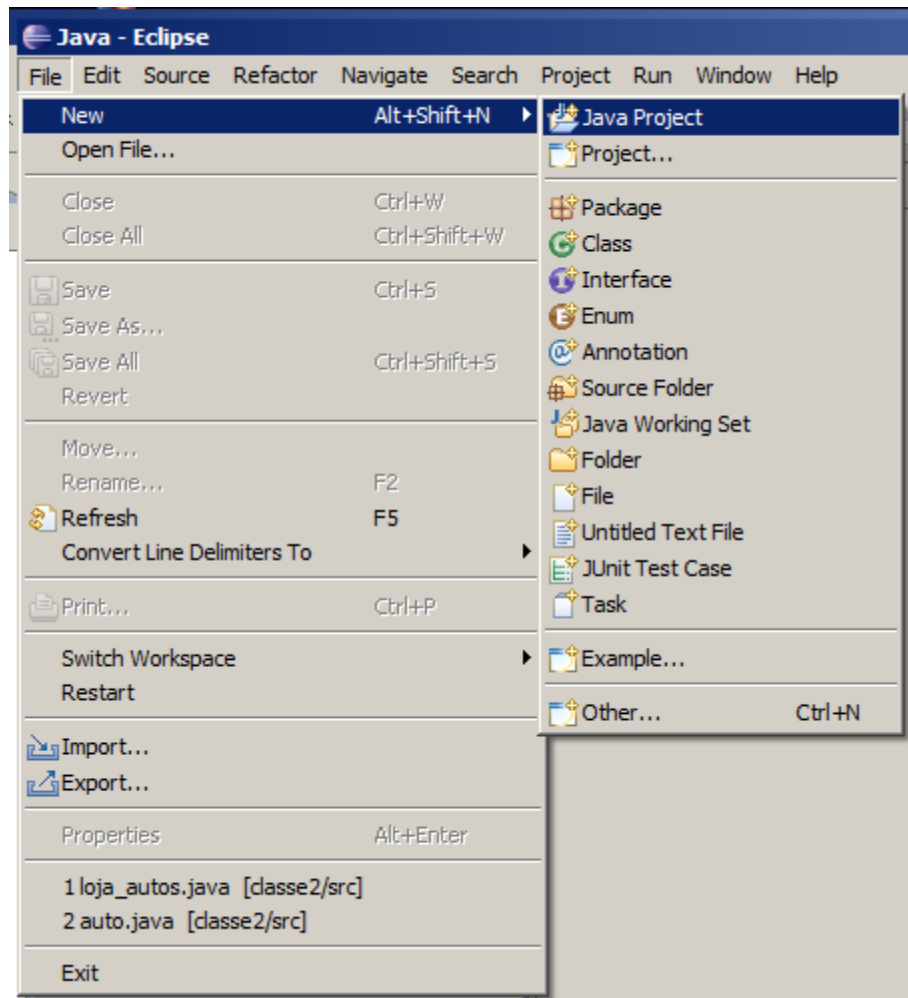


# Projeto no Eclipse





# Projeto no Eclipse



# Projeto no Eclipse

**New Java Project**

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location:  [Browse...](#)

**JRE**

☐ Use an execution environment JRE:

☐ Use a project specific JRE:

☒ Use default JRE (currently 'jre1.8.0\_20') [Configure JREs...](#)

**Project layout**

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

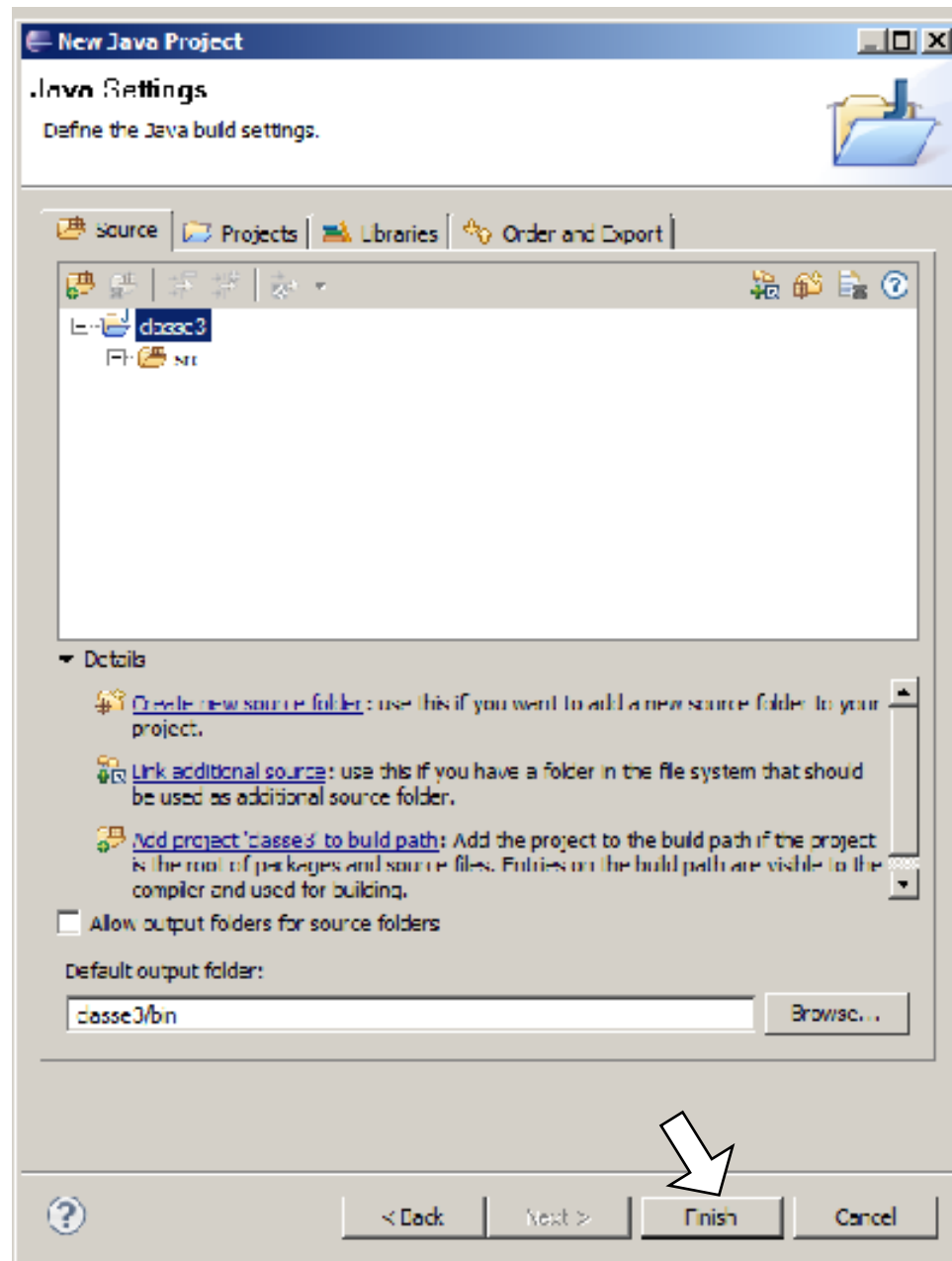
**Working sets**

☐ Add project to working sets

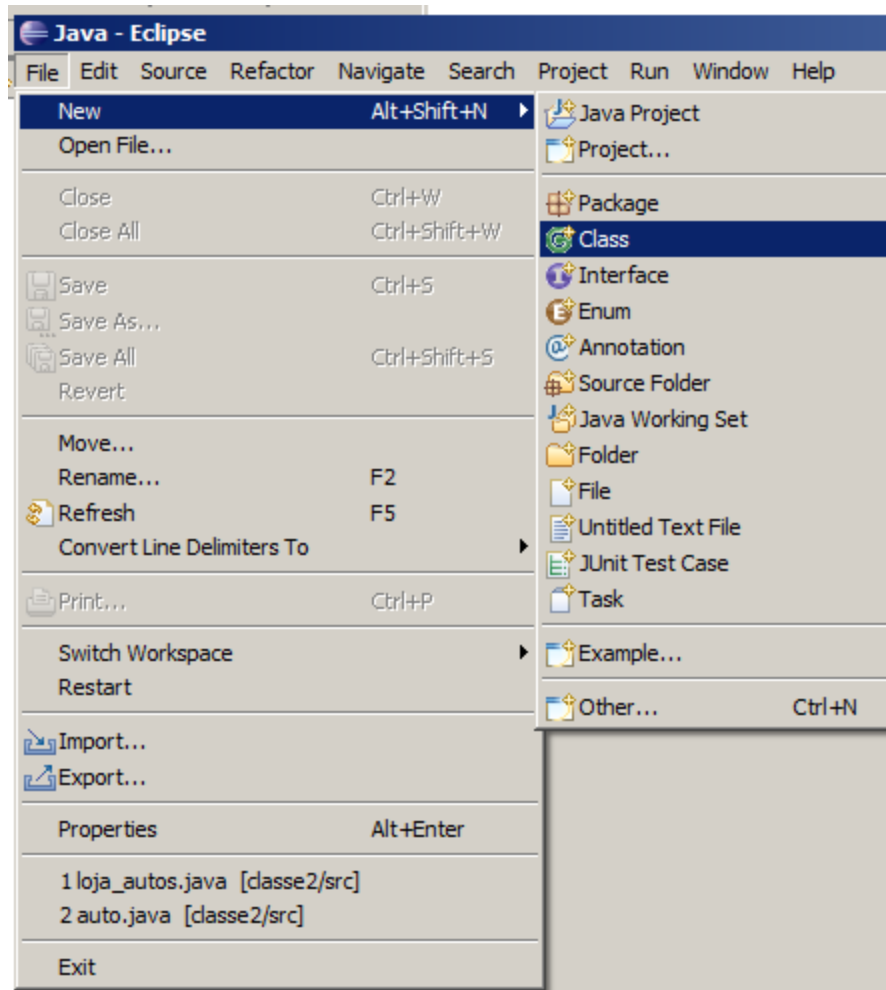
Working sets:  [Select...](#)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

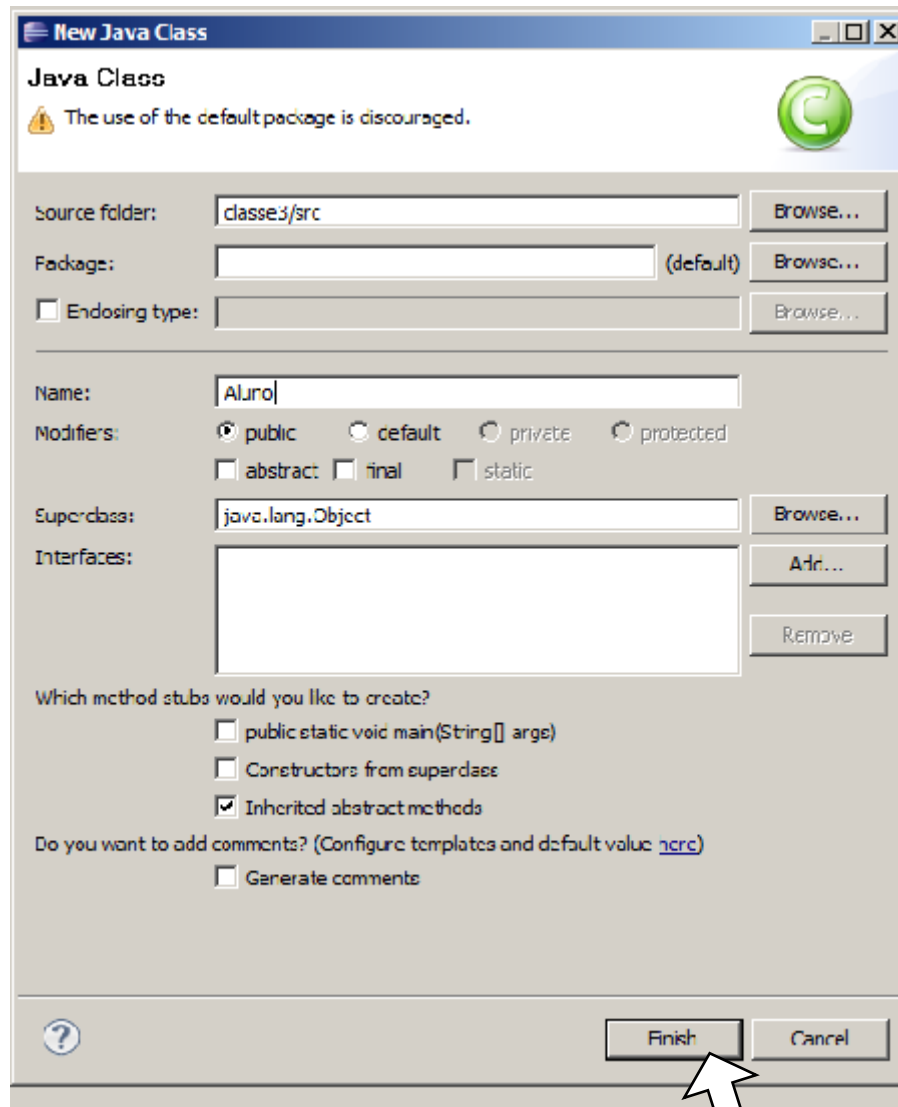
# Projeto no Eclipse



# Projeto no Eclipse



# Projeto no Eclipse



**New Java Class**

Java Class

⚠ The use of the default package is discouraged.

Source folder:

Package:

☐ Enclosing type:

---

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

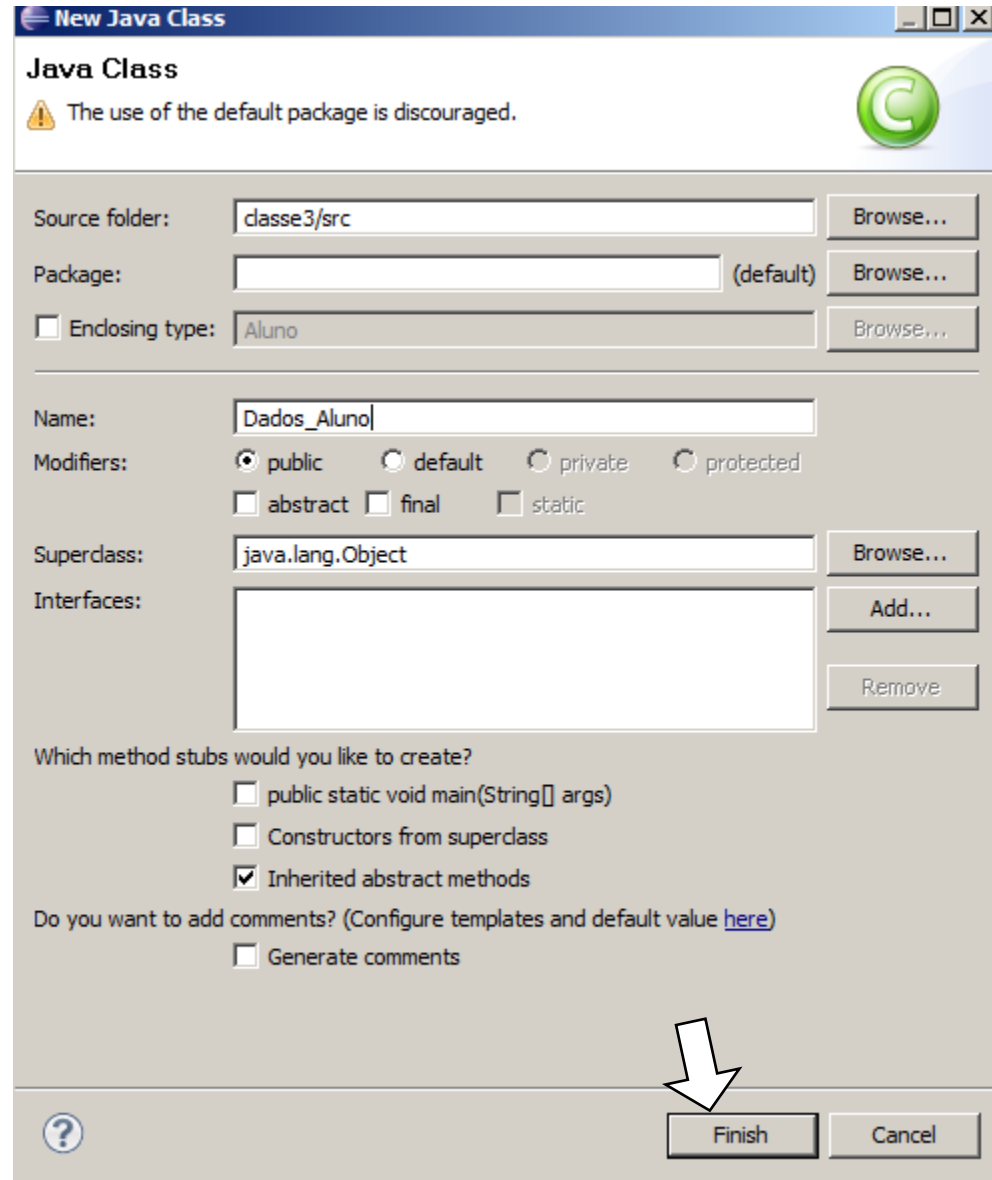
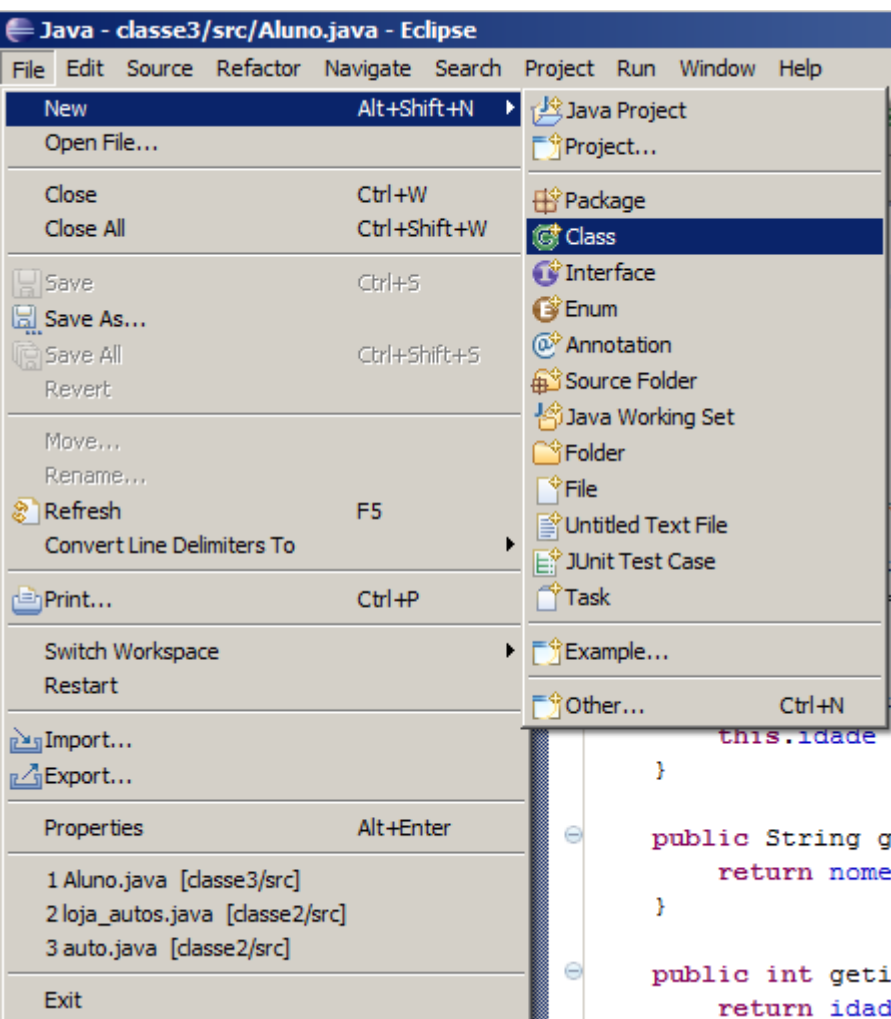
☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value: [ncrc](#))

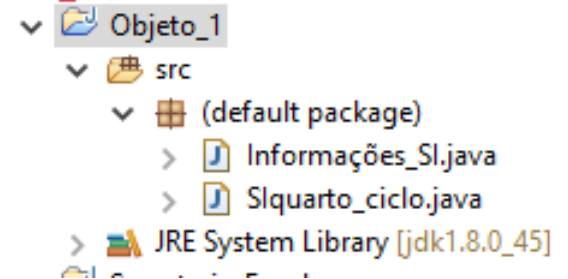
☐ Generate comments

# Projeto no Eclipse




# Projeto no Eclipse

```
1 public class SIquarto_ciclo {
2
3
4     //atributos
5     String nome_aluno;
6     String nome_disciplina;
7     String nome_professor;
8     String dia_semana;
9     double nota_prova;
10
11     //métodos relacionados aos atributos
12     public void nome (String nome){
13         System.out.println("o nome do aluno é: " + nome);
14     }
15
16     public void disciplina (String disciplina){
17         System.out.println("o nome da disciplina é: " + disciplina);
18     }
19
20     public void professor (String professor){
21         System.out.println("o nome do professor é: " + professor);
22     }
23
24     public void dia (String dia){
25         System.out.println("o dia da semana é: " + dia);
26     }
27
28     public void nota (double nota){
29         System.out.println("nota da prova: " + nota);
30     }
31
32 }
33
```



# Projeto no Eclipse

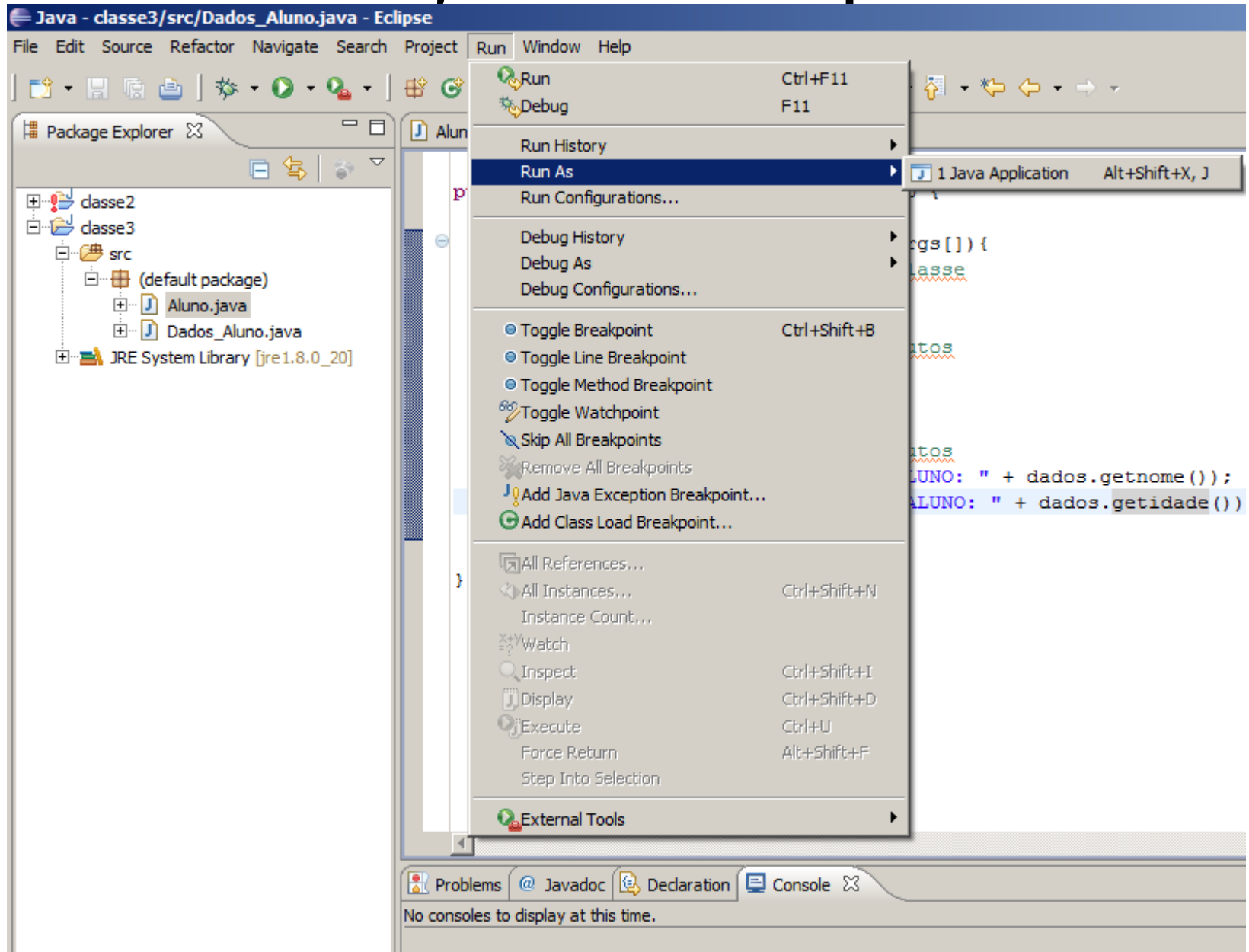
```
1
2 public class Informações_SI {
3     public static void main (String args[]){
4         //objeto do tipo SIquarto_ciclo
5
6         SIquarto_ciclo si = new SIquarto_ciclo();
7
8         si.nome_aluno = "Vitória";
9         si.nome_disciplina = "Programação Orientada a Objetos";
10        si.nome_professor = "Mauricio";
11        si.dia_semana = "segunda feira";
12        si.nota_prova = 10.0;
13
14        si.nome(si.nome_aluno);
15        si.disciplina(si.nome_disciplina);
16        si.professor(si.nome_professor);
17        si.dia(si.dia_semana);
18        si.nota(si.nota_prova);
19    }
20
21 }
```

Problems @ Javadoc Declaration Console  Debug

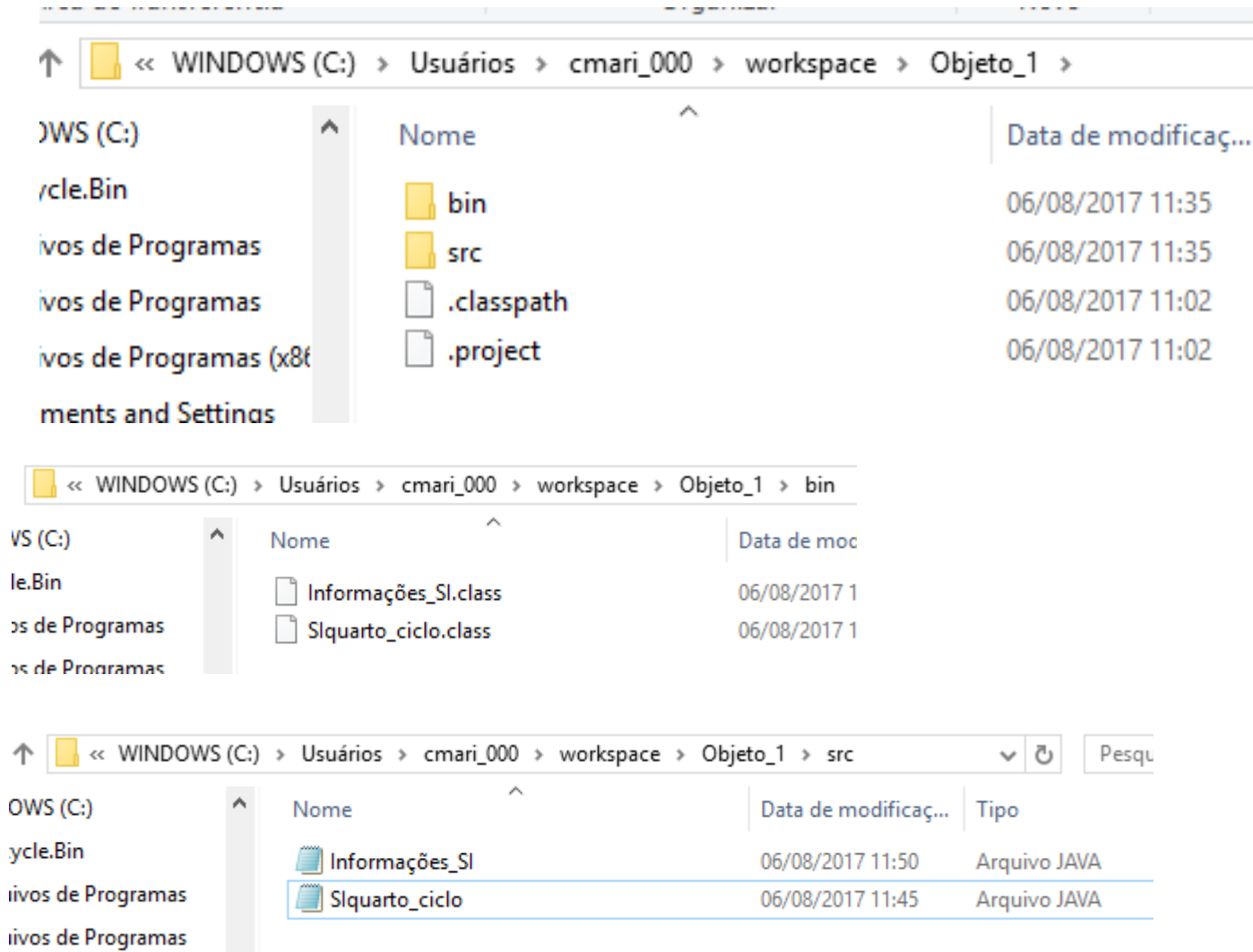
<terminated> Informações\_SI [Java Application] C:\Arquivos de Programas\Java\jdk1.8.0\_45\b  
o nome do aluno é: Vitória  
o nome da disciplina é: Programação Orientada a Objetos  
o nome do professor é: Mauricio  
o dia da semana é: segunda feira  
nota da prova: 10.0



# Projeto no Eclipse



# Projeto no Eclipse



# Projeto no Eclipse

```
public class SIquarto_ciclo {  
  
    //atributos  
    String nome_aluno;  
    String nome_disciplina;  
    String nome_professor;  
    String dia_semana;  
    double nota_prova;  
  
    public void nome (String nome){  
        System.out.println("o nome do aluno é: " + nome);  
    }  
  
    public void disciplina (String disciplina){  
        System.out.println("o nome da disciplina é: " + disciplina);  
    }  
  
    public void professor (String professor){  
        System.out.println("o nome do professor é: " + professor);  
    }  
  
    public void dia (String dia){  
        System.out.println("o dia da semana é: " + dia);  
    }  
  
    public void nota (double nota){  
        System.out.println("nota da prova: " + nota);  
    }  
}
```

# Projeto no Eclipse



Informações\_SI - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

---

```
public class Informações_SI {  
    public static void main (String args[]){  
        //objeto do tipo SIquarto_ciclo  
  
        SIquarto_ciclo si = new SIquarto_ciclo();  
  
        si.nome_aluno = "Vitória";  
        si.nome_disciplina = "Programação Orientada a Objetos";  
        si.nome_professor = "Mauricio";  
        si.dia_semana = "segunda feira";  
        si.nota_prova = 10.0;  
  
        si.nome(si.nome_aluno);  
        si.disciplina(si.nome_disciplina);  
        si.professor(si.nome_professor);  
        si.dia(si.dia_semana);  
        si.nota(si.nota_prova);  
    }  
}
```

# Exercícios

1. Construir a aplicação orientada a objetos com as classes SIquarto\_ciclo e Informações\_SI.
2. Dados os códigos das classes Fatec e Consulta\_Fatec, editar, compilar e executar os mesmos.



Fatec.java - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

```
public class Fatec {  
  
    String unidade;  
    String curso;  
  
    public void Informações_Fatec (String u, String c){  
        System.out.println("UNIDADE: \t" + u + "CURSO: \t" + c);  
    }  
}
```



Consulta\_Fatec.java - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

```
public class Consulta_Fatec {  
    public static void main (String args []){  
  
        Fatec o = new Fatec( );  
  
        o.unidade = "Baixada Santista";  
        o.curso = "Sistemas para Internet";  
  
        o.Informações_Fatec(o.unidade, o.curso);  
    }  
}
```

# Conceitos de Compilação

## Definições

**Gramática:** determina a forma ou sintaxe das instruções válidas de uma linguagem. É uma descrição formal da sintaxe.

**Sintaxe:** é o formato das instruções.

**Semântica:** está relacionada ao significado das instruções.

Exemplo: diferença entre sintaxe e semântica. Dadas as instruções a) e b) , onde as variáveis I, J e K são do tipo inteiro, e as variáveis X e Y são do tipo real:

a)  $I := J + K$

b)  $I := X + Y$

## Gramática, sintaxe e semântica:

As duas instruções têm a mesma sintaxe, ou seja, são compostas por um operador de atribuição “:=”, um operador aritmético “+”, e três variáveis, onde a soma de duas é atribuída a uma terceira. As instruções têm semânticas diferentes: em a) a semântica especifica que as variáveis da expressão serão somadas usando operador aritmético de cálculo inteiro, e o resultado será armazenado na variável I. Em b) há uma operação de soma com ponto flutuante, com a conversão do resultado para inteiro antes de ser armazenado na variável I.

Exemplo de notação utilizada para registrar a gramática de linguagens: Backus-Naur Form (BNF). Uma gramática BNF consiste em um conjunto de regras, cada uma das quais definindo a sintaxe de alguma construção de uma linguagem de programação.

Exemplo: gramática BNF para um subconjunto da linguagem PASCAL (definição da sintaxe de uma instrução READ do PASCAL, identificada na gramática como <read>).

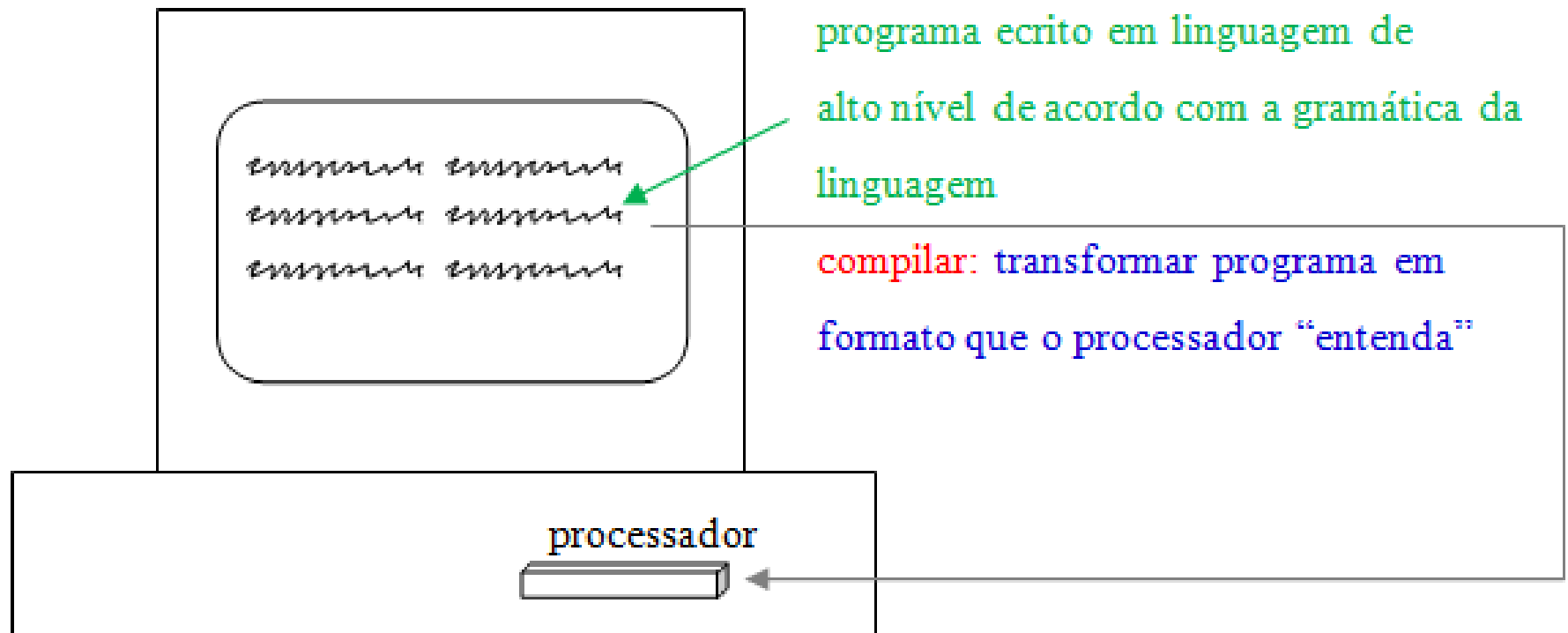


## Gramática, sintaxe e semântica:



# Conceitos de Compilação

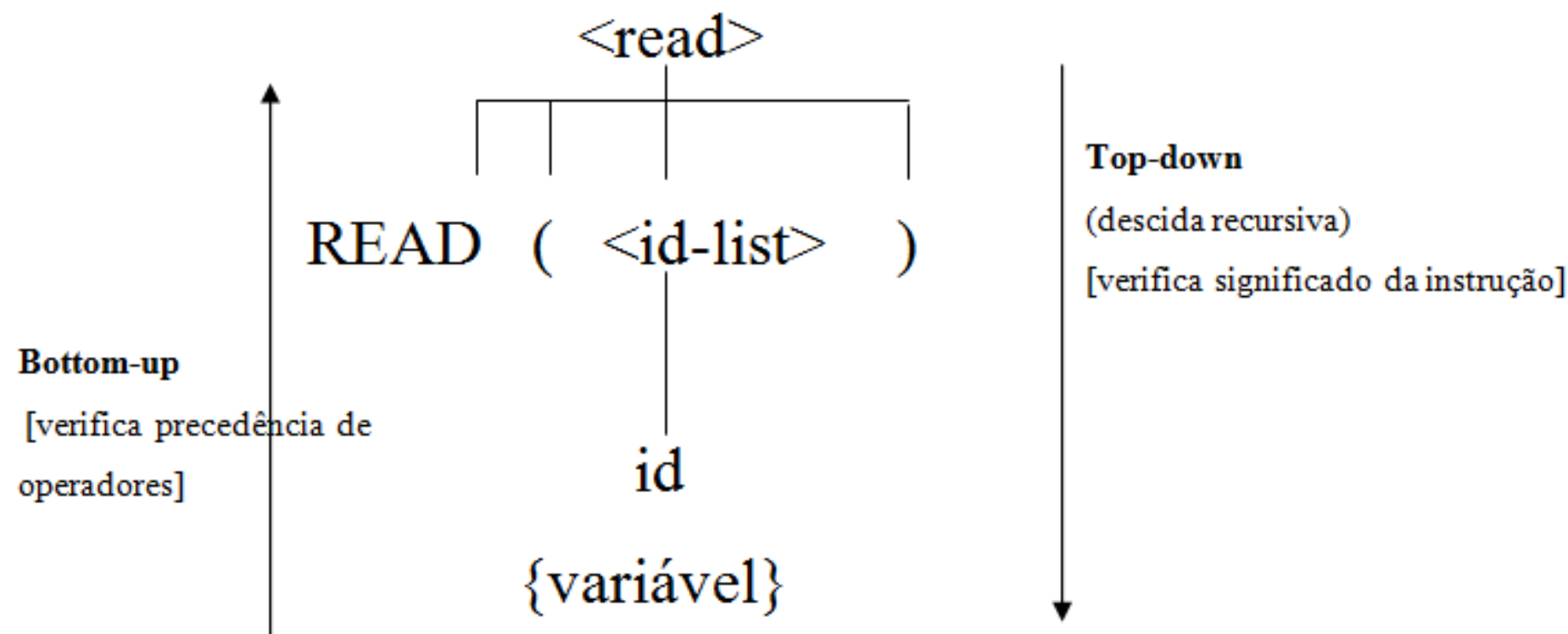
## Etapas da Compilação



# Etapas da Compilação:

**Análise Léxica:** leitura do programa → reconhecimento dos tokens (cada token tem um código numérico equivalente). Ferramenta: SCANNERS ou ANALISADORES LÉXICOS (software).

**Análise Sintática:** Após análise dos tokens, cada instrução deve ser reconhecida como uma das construções válidas da gramática. Ferramenta: PARSER. → construção da Árvore Sintática ou Árvore de Parsing.



## Etapas da Compilação:

***Geração do Código:*** após o reconhecimento da instrução como sendo pertencente à gramática da linguagem, é executada uma rotina semântica para a geração do código (código objeto), que depois é transformado em linguagem Assembly (linguagem de baixo nível, de máquina, de acordo com o processador). A linguagem Assembly, quando compilada, gera um arquivo binário de “0” e “1”, onde o zero equivale a um nível lógico digital = zero (tensão de 0V) e o um equivale a um nível lógico digital = um (tensão de  $\cong 5V$ ). O processador só reconhece uma sequência de sinais digitais.

## Etapas da Compilação:

Exemplo:

