



Aula 2

Programação Orientada a Objetos (POO)

Prof. Dr. Richarlyson Alves D'Emery
grupo: http://groups.google.com/group/mpoo_uast
email grupo: mpoo_uast@googlegroups.com
contato: richarlyson.demery@ufrpe.br

Desenvolvimento de software com POO



Objetivo

Desenvolver software por OO nada mais é que a aproximação do manejo das coisas de um mundo real com o manejo das estruturas de um sistema de software.



O que é preciso saber?

1)

A OO, tanto na Programação quanto na análise e na modelagem de sistemas, está centrada na resolução de problemas na forma **bottom-up**, ou seja, resolvendo as pequenas partes do problema, chega-se à solução completa.



O que é preciso saber?

- Uso de conceitos de:
 - **Abstração** de dados
 - Um tipo abstrato de dados (TAD) é uma especificação de um conjunto de dados e operações que podem ser executadas sobre esses dados.
 - **Modularização**
 - Transformar um TAD em um “componente de software”, que pode ser reutilizado e, se necessário, adaptado para diferentes projetos.

2)

TADs e modularização constituem os alicerces dos conceitos de **classes e objetos** da OO.



O que é preciso saber?

3)

O desenvolvimento de software:

- é um processo que envolve:
criação, implementação e manutenção.
- utiliza **metodologias**
 - Etapas bem definidas que ocorrem desde a concepção da ideia até a entrega do produto final.



Processo de desenvolvimento de software

Cada modelo de processo é recomendado para determinadas situações.

Modelo cascata ou sequencial linear ou ciclo de vida clássico.

Baseados em prototipagem

Modelos RAD (*Rapid Application Development*)

Modelo espiral

Modelo incremental

Modelo iterativo e incremental

Modelos ágeis

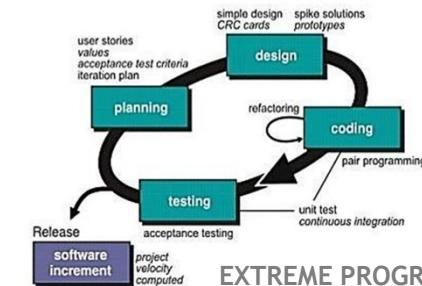
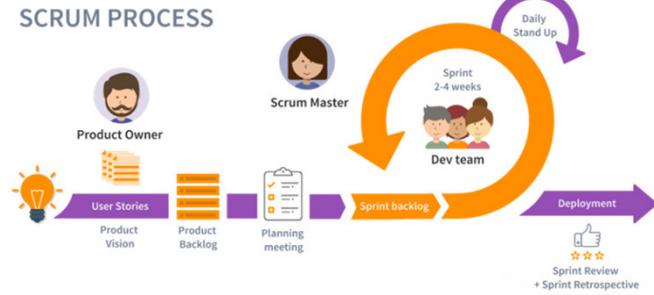
Processo unificado

(PRESSMAN, 2011; SOMMERVILLE, 2007)



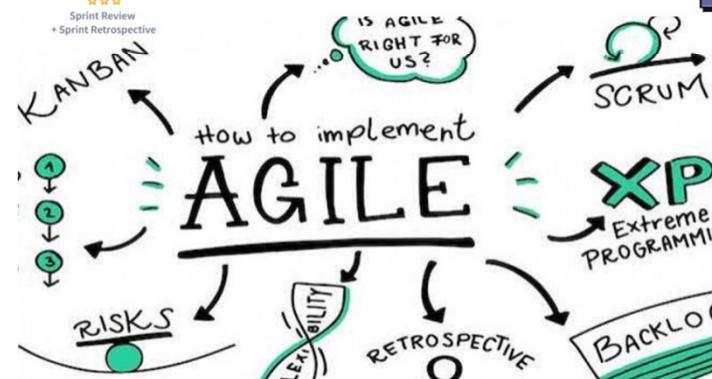
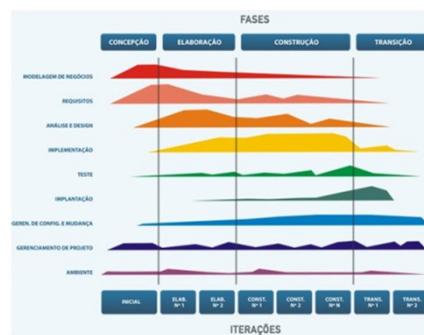
Métodos de desenvolvimento de software

SCRUM PROCESS

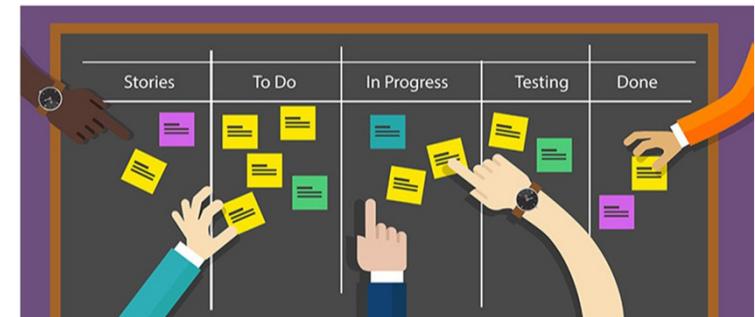


EXTREME PROGRAMMING (XP)

IBM Rational Unified Process (IRUP)



KANBAN





Métodos de desenvolvimento de software



Empresas obtém resultados em um período menor de tempo [Filho 2008]



Métodos de desenvolvimento de software



vantagens x metodologias ágeis x empresa



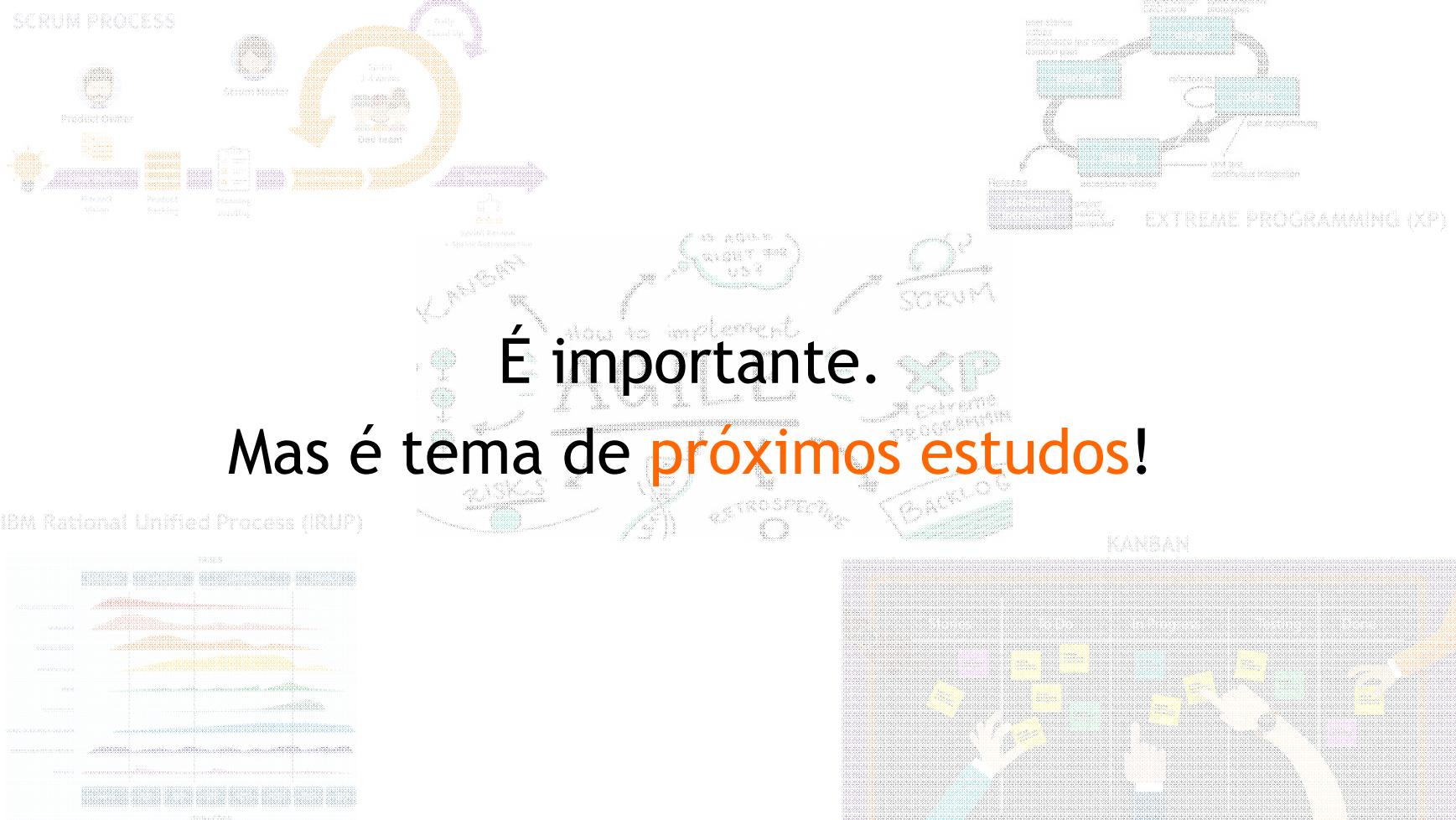
É preciso
aprender:
praticando!



Metodologias ágeis durante
estudos



Métodos de desenvolvimento de software





Vamos Praticar!

Problemática Login/Logoff

Entrar



Vamos Praticar!

Mas, por onde começar?



Aproveitando o que já sabemos!





Começando por algum lugar

O que devemos fazer?

Modelar o *front-end* ou *back-end*?



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

■ Características:

- Utiliza apenas 4 diagramas UML (apenas os necessários), ao invés dos 14 diagramas.
- Minimiza a paralisia da análise: ignora palavras confusas do padrão UML tais como << extends >>, << includes >>, etc que não adicionam valor algum a essa etapa e que retardam a passagem da análise para o projeto;



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

■ Características:

- Utiliza apenas 4 diagramas UML (apenas os necessários), ao invés dos 14 diagramas.
- Minimiza a paralisação da análise: ignora palavras confusas do padrão UML tais como <<extends>>, <<includes>>, etc. **Mas usaremos <<extends>> e <<includes>>** que não são ignorados, só que nessa etapa é que retardam a passagem da análise para o projeto;



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

■ Características:

- Utiliza apenas 4 diagramas UML (apenas os necessários), ao invés dos 14 diagramas.
- Minimiza a paralisia da análise: ignora palavras confusas do padrão UML tais como << extends >>, << includes >>, etc que não adicionam valor algum a essa etapa e que retardam a passagem da análise para o projeto;
- Possui alto grau de rastreamento: todos os requisitos são associados a casos de uso e classes, que por sua vez formam o eixo de sustentação de todo o processo;
- É iterativo e incremental
- É baseado nas questões fundamentais de OO: o que fazem os usuários? (**casos de uso**); quais os objetos do mundo real? (**modelo de domínio**); quais os objetos relacionados com os casos de usos? (**robustez**); como os objetos colaboram entre si? (**sequência**); como realmente será construído o software? (**classes**).



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- Obtenção dos casos de uso
- Modelagem do domínio
- Análise de robustez
- Modelagem de interação
- Especificação do sistema



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

Processo de desenvolvimento dirigido por casos de uso

Metodologia orientada ao domínio

Metodologia prática, simples, intermediária entre a **complexidade do IRUP** e a **simplicidade do XP**, mas sendo ao mesmo tempo poderosa para guiar a análise e projeto orientado a objetos.



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- **Obtenção dos casos de uso**
- **Modelagem do domínio**
- **Análise de robustez**
- **Modelagem de interação**
- **Especificação do sistema**

Os **casos de uso** capturam os **requisitos** da aplicação através da representação dos atores que usam o (ou são usados pelo) sistema e possíveis ações. Eles são **descritos** sob o **ponto de vista do usuário**: o que ele pode fazer, como ele interage com o sistema. Os casos de uso são apresentados na forma de diagramas **UML** de casos de uso para o sistema e descrição textual de cada caso de uso. A descrição textual contempla pelo menos o fluxo de eventos para a execução da ação associada ao caso de uso e os fluxos alternativos, caso alguma situação interrompa o fluxo principal de eventos.



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- **Obtenção dos casos de uso**
- **Modelagem do domínio**
- **Análise de robustez**
- **Modelagem de interação**
- **Especificação do sistema**

Muitas pessoas têm **dificuldade** em **visualizar** um **software** apenas por **meio textual**.

Incentiva-se o **uso** de uma sequência de **telas** e interfaces gráficas como ponto de partida **para** simular a interação entre usuários e sistema, **identificando** os **casos de uso** e facilitando o seu entendimento pelas partes interessadas.

Essas telas podem ser **wireframes**, sem muita elaboração gráfica, mas devem conter o máximo de detalhes de tal forma que explique os fluxos alternativos do caso de uso.



VIDA DE
PROGRAMADOR
.COM.BR

```
real historia;  
string sender = "Cintia";
```

#1756





Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- **Obtenção dos casos de uso**
- **Modelagem do domínio**
- **Análise de robustez**
- **Modelagem de interação**
- **Especificação do sistema**

A partir dos textos associados aos **casos de uso**, **extrair** (por uma análise dos substantivos presentes nos textos) os **conceitos** que serão potencialmente associados a **objetos** da aplicação. **Representar** esses conceitos através de um **diagrama de classes UML** (neste momento, **sem atributos ou métodos**).



Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- **Obtenção dos casos de uso**
- **Modelagem do domínio**
- **Análise de robustez**
- **Modelagem de interação**
- **Especificação do sistema**

Para cada caso de uso, identificar os objetos e os eventos que os relacionam através de um diagrama de robustez. Objetos podem ser de três tipos: de **fronteira**, de **controle** ou **de entidade**.

- Atores só interagem com objetos de fronteira, que não podem interagir diretamente com objetos de entidade.





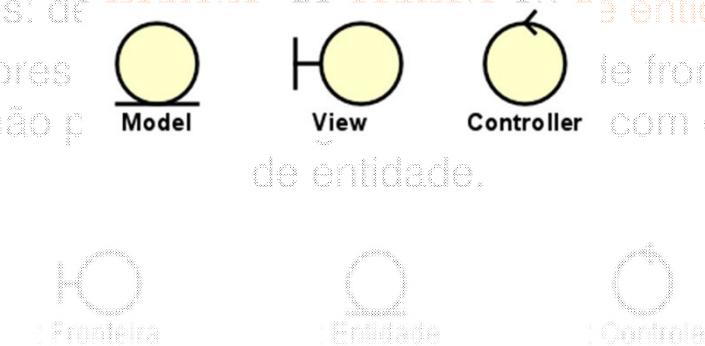
Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- Obtenção dos casos de uso
- Modelagem do domínio
- Análise de robustez
- Modelagem de interação
- Especificação do sistema

Para cada caso de uso MPOO, usaremos um diagrama de UML para descrever os três tipos de entidade: Atores, que não pertencem ao sistema; Model, que é a entidade de fronteira, View, que é a interface com o usuário; e Controller, que é a entidade de controle.





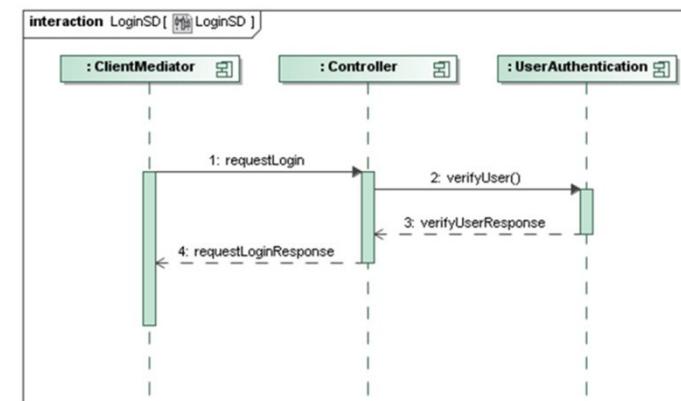
Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- Obtenção dos casos de uso
- Modelagem do domínio
- Análise de robustez
- Modelagem de interação
- Especificação do sistema

Para cada caso de uso, um **diagrama de sequência UML** é desenvolvido. Nesta etapa do processo, **comportamento** (um conjunto de métodos) é **atribuído** a cada **objeto** da aplicação.



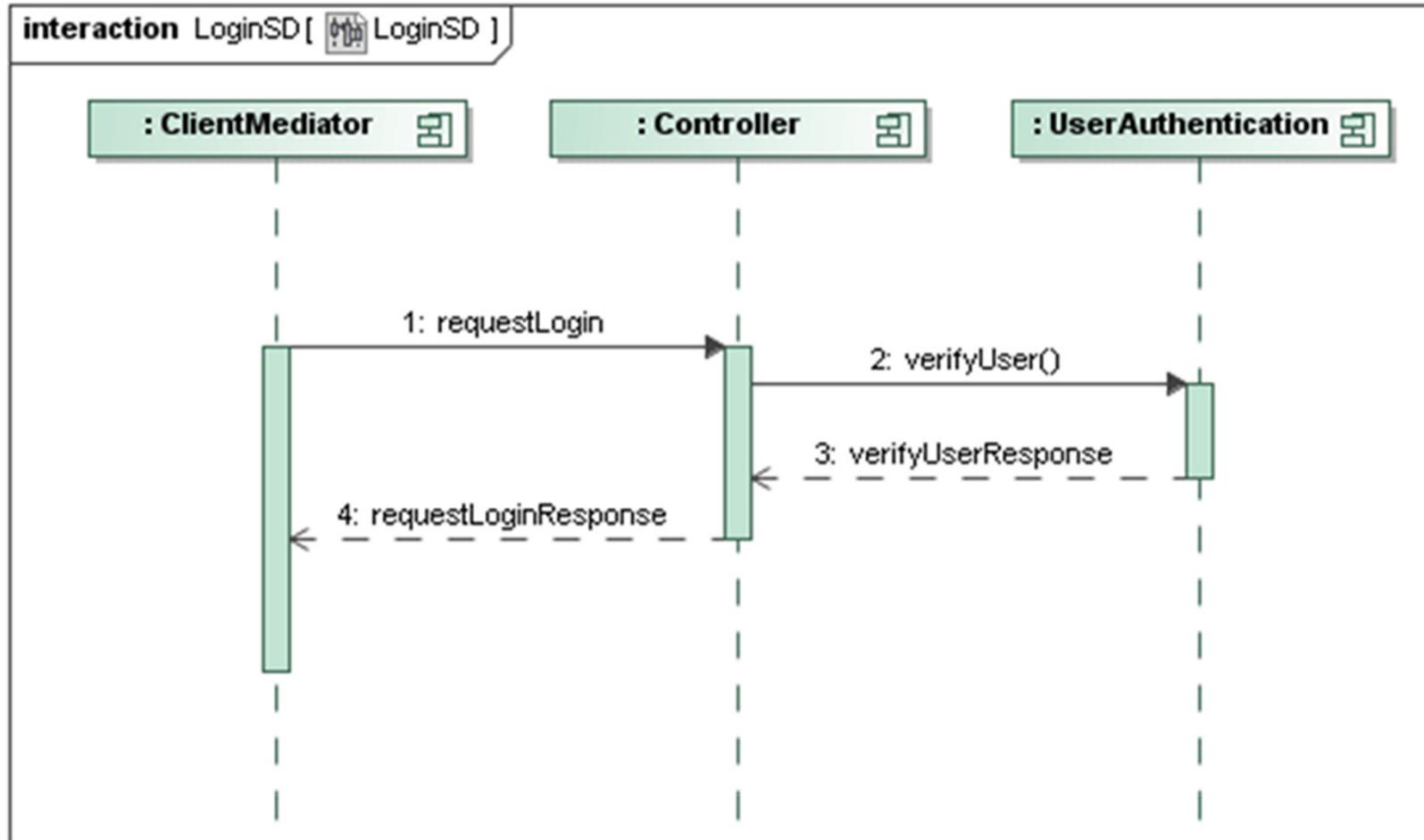
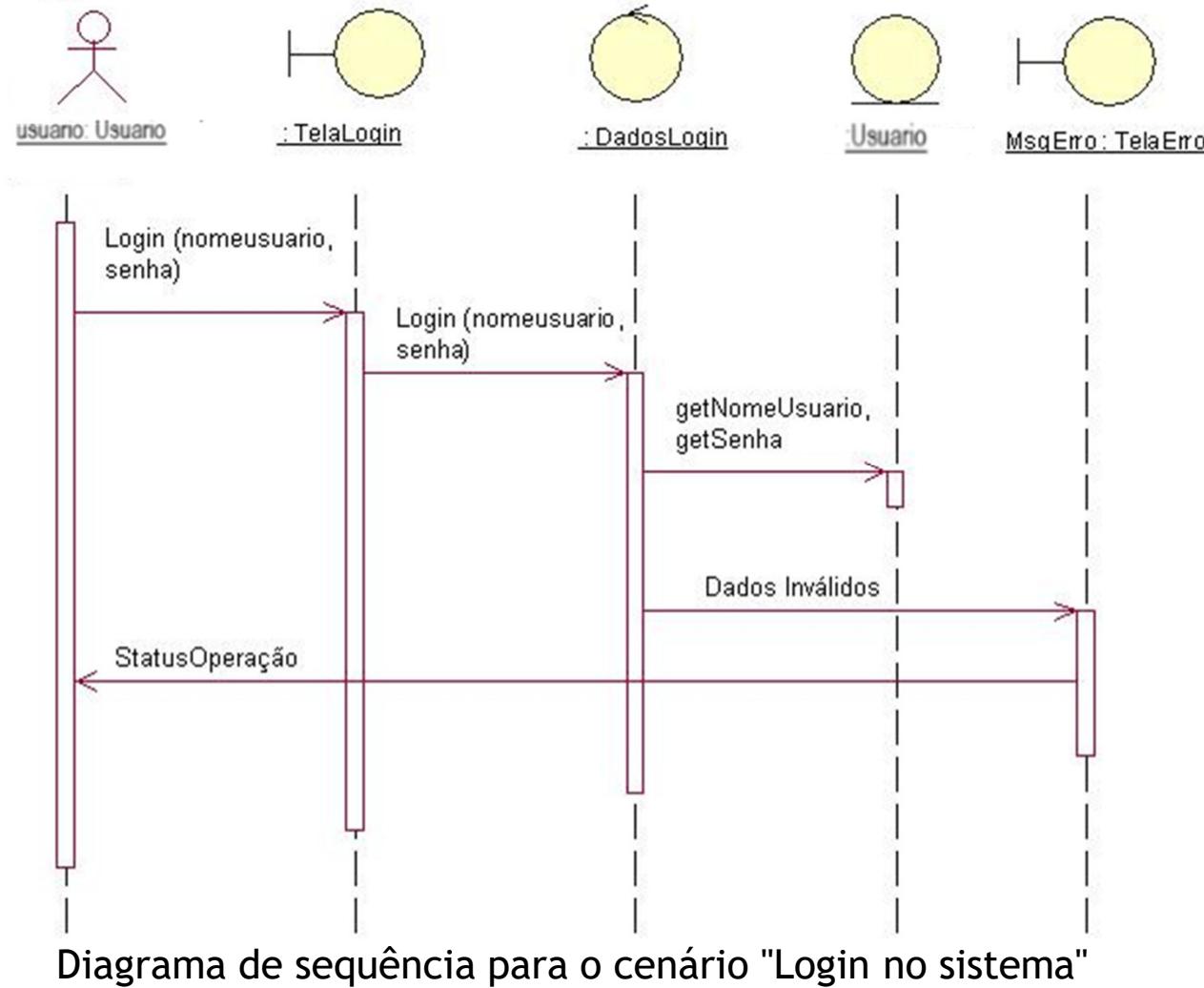


Diagrama de sequência para o cenário "Login no sistema"





Começando por algum lugar

ICONIX

processo simples de desenvolvimento dirigido por casos de uso

- Obtenção dos casos de uso
- Modelagem do domínio
- Análise de robustez
- Modelagem de interação
- **Especificação do sistema**

Com os métodos para cada tipo de objeto já definidos, o **diagrama de classe é revisto** para incorporar a definição dessa interface operacional a cada classe.



Onde Modelar e Programar?



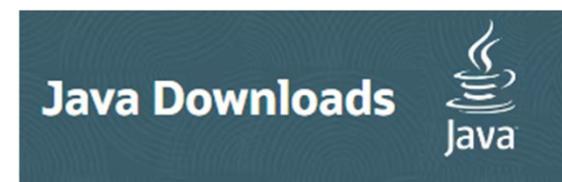
[Download](#)

<https://astah.net/support/astah-uml/system-requirements/>



[Download 2024-03](#)

<https://www.eclipse.org/downloads/packages/release>



[Java downloads](#)

<https://www.oracle.com/br/java/technologies/downloads/>

Relembrando

<https://www.java.com/releases/>



JDK Releases

The release information on this page covers the JDK releases that were widely distributed or significant to the development of Java. It does not cover patch releases or other one-off releases.

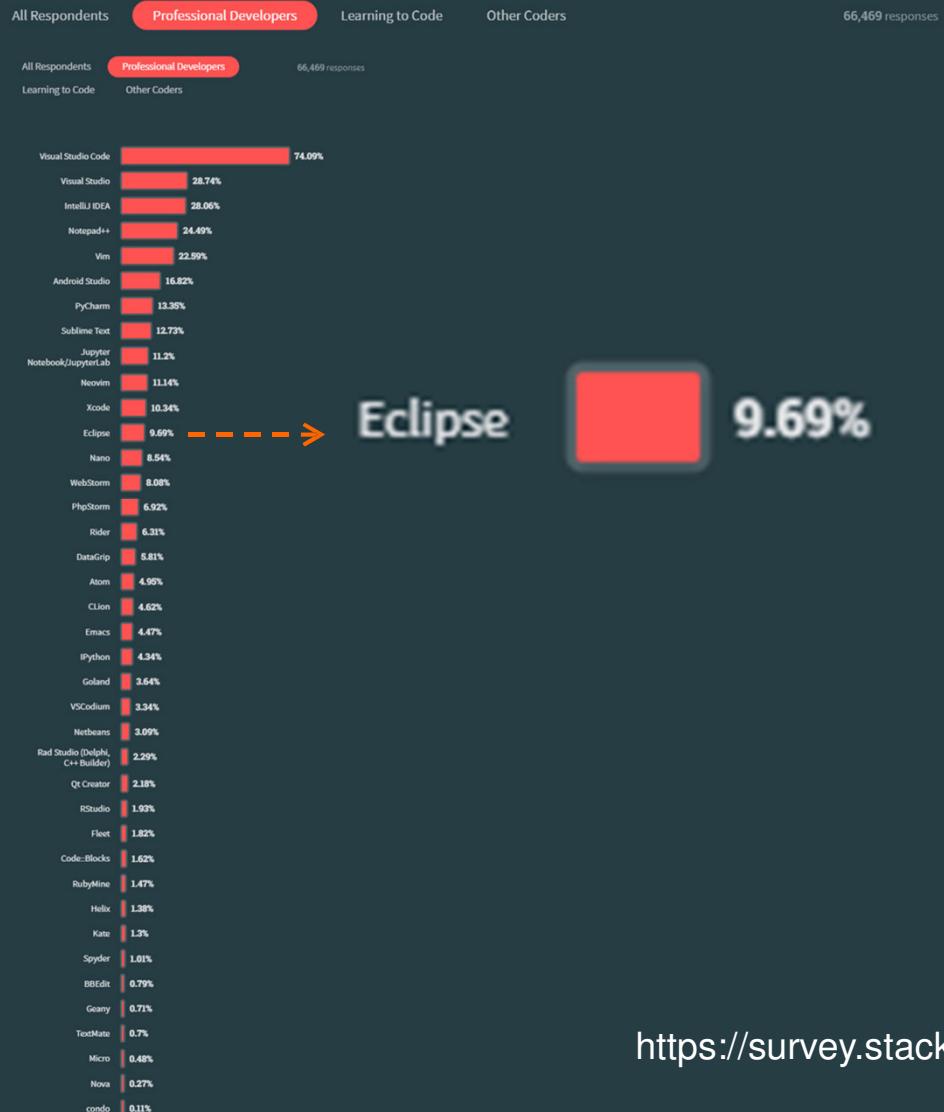
Java Release Support Timeline

The length of time updates are provided for a Feature release is outlined in the [Oracle Java SE Support Roadmap](#). Long Term Support (LTS) releases are indicated in the follow table with "LTS".

Release Family	GA	End Of Support Life (EOSL)
22	19th March 2024	September 2024
21 LTS	19th September 2023	September 2031
20	21st March 2023	September 2023
19	20th September 2022	March 2023
18	22nd March 2022	September 2022
17 LTS	14th September 2021	September 2029
16	16th March 2021	September 2021
15	15th September 2020	March 2021
14	17th March 2020	September 2020
13	17th September 2019	March 2020
12	19th March 2019	September 2019
11 LTS	25th September 2018	September 2026
10	20th March 2018	September 2018
9	21st September 2017	March 2018
8 LTS	18th March 2014	December 2030
7 LTS	11th July 2011	July 2022
6 LTS	12th December 2006	December 2018
5 LTS	30th September 2004	July 2015
4 LTS	13th February 2002	March 2013
3 LTS	8th May 2000	April 2011
2 LTS	4th December 1998	December 2003
11 LTS	28th March 1997	January 2003
1 LTS	23rd January 1996	October 2002

Integrated development environment

Visual Studio Code remains the preferred IDE across all developers, increasing its use among those learning to code compared to professional developers (78% vs. 74%).



<https://survey.stackoverflow.co/2023/#section-most-popular-technologies-integrated-development-environment>

POO:
Classe - Atributo - Método Construtor- Objeto



Vamos Praticar!

UML e Codificação em Java



Situação Problema

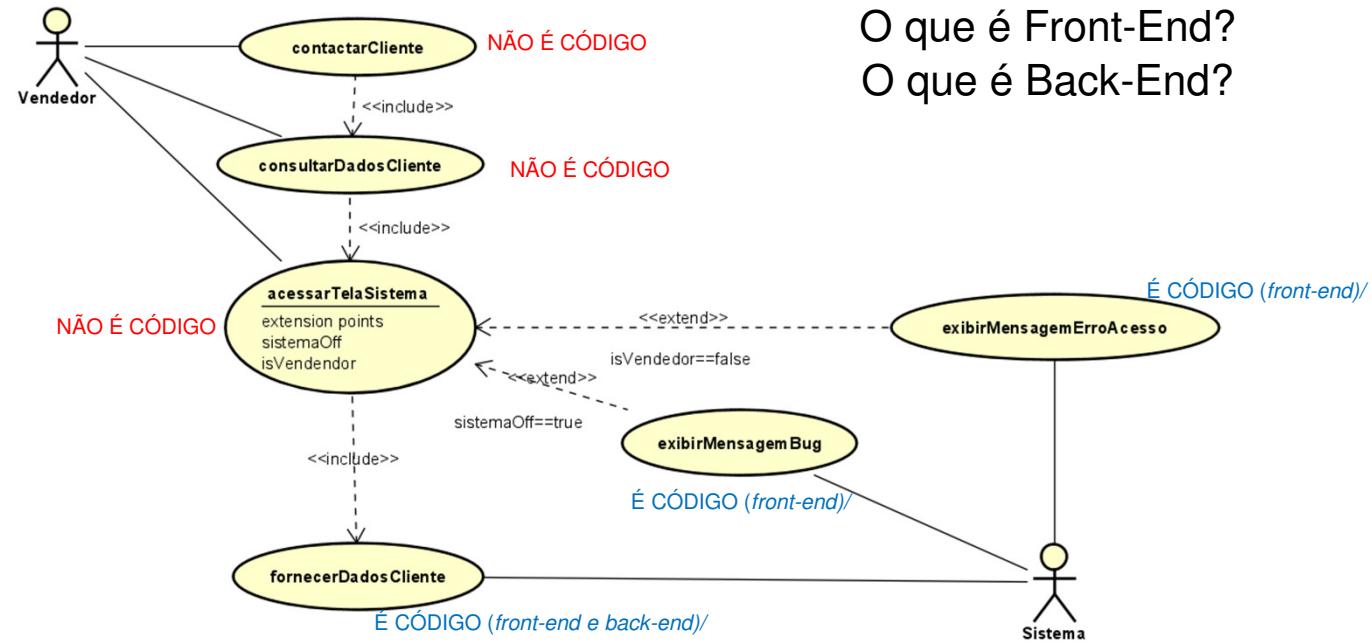




Vamos Praticar!

Resultado Prática

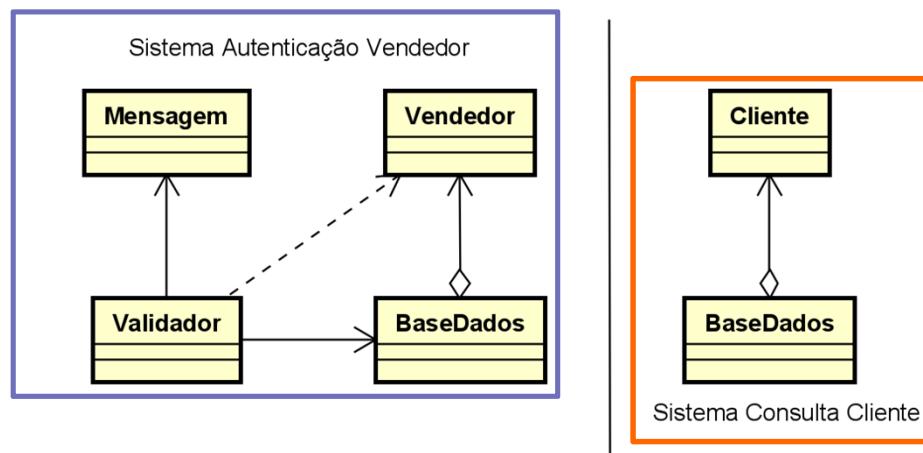
■ Modelagem: □ Use Case





Vamos Praticar! Resultado Prática

- Modelagem:
 - Modelo de Domínio



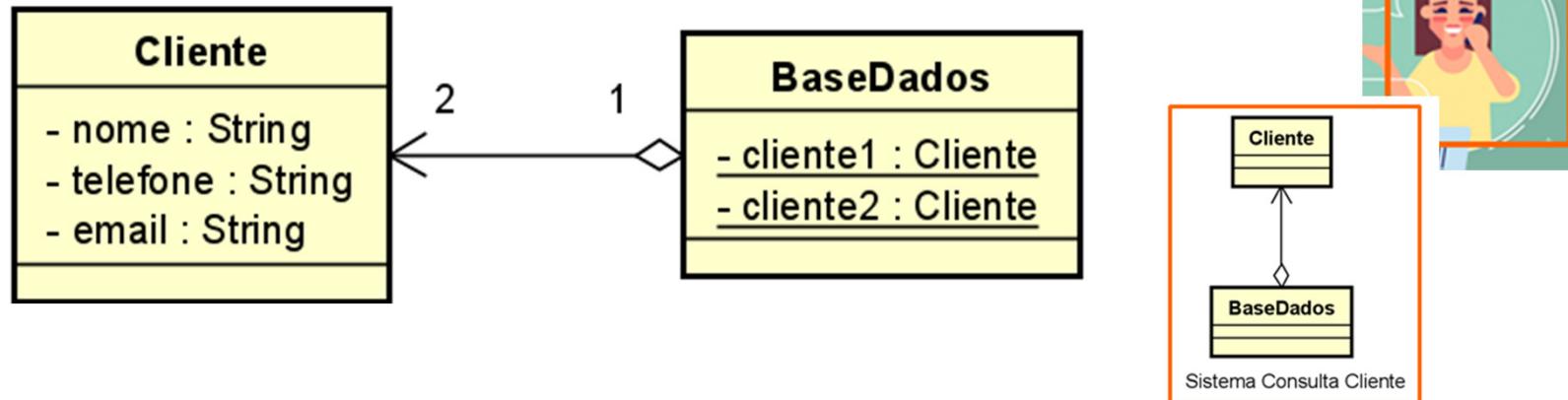


Vamos Praticar!

Resultado Prática

■ Modelagem:

- Diagrama de Classes (atualizando o Modelo de Domínio)





Vamos Praticar!

Resultado Prática

- Modelagem:
 - Codificação em Java

```
Cliente.java ×  
1 public class Cliente {  
2     String nome;  
3     String telefone;  
4     String email;  
5  
6     public Cliente() {}  
7  
8     public Cliente(String nome, String telefone, String email) {  
9         this.nome = nome;  
10        this.telefone = telefone;  
11        this.email = email;  
12    }  
13 }
```

Use a geração de Código:

Construtor:

*Na linha do programa acionar
Alt+Shift+S “depois” O*

continua...

(vide src da Aula e material complementar)



Vamos Praticar!

Resultado Prática

- Modelagem:
 - Codificação em Java

```
App.java x
1 public class App {
2     public static void main(String[] args) {
3         Cliente cliente1 = new Cliente();
4         cliente1.nome="João Silva";
5         cliente1.telefone="(87)99999-9999";
6         cliente1.email="joao@gmail.com";
7
8         Cliente cliente2 = new Cliente("Maria Santos", "(87)99999-9999", "maria@gmail.com");
9     }
10 }
```

Use a geração de Código para autocomplete:
Na linha do programa acione:
Ctrl + space

continua...
(vide src da Aula e material complementar)

Dicas de *debug*





Vamos Praticar! Debug no Eclipse

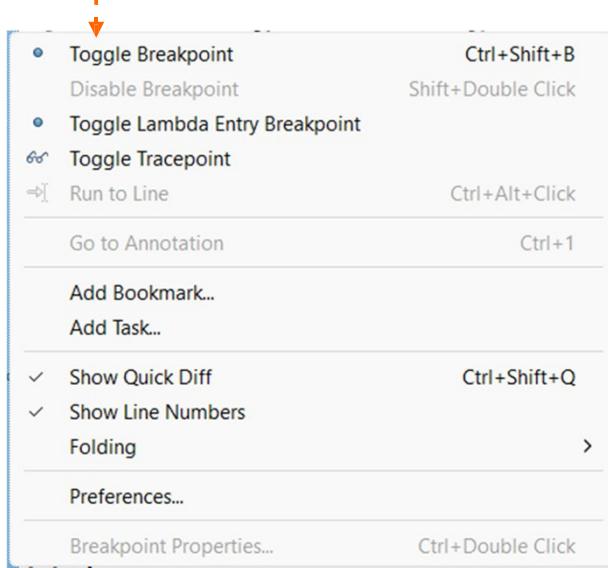
1)

Add breakpoint:

Clicar no botão direito do mouse e
selecionar Toggle Breakpoint

ou

Na linha do programa acionar
Ctrl+Shift+B



```
App.java ×
1 public class App {
2     public static void main(String[] args) {
3         Cliente cliente1 = new Cliente();
4         cliente1.nome="João Silva";
5         cliente1.telefone="(87)99999-9999";
6         cliente1.email="joao@gmail.com";
7
8     }
9 }
10 }
```



Vamos Praticar!

Debug no Eclipse

2)

The screenshot shows the Eclipse IDE interface with the title bar "workspaceMPOO2024.2 - Aula2_Slide/src/App.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations. The left sidebar shows the "Package Expl..." view with a tree structure: Aula2_Slide > JRE System Library [Java] > src > (default package) > App.java and Cliente.java. The main editor window displays the following Java code:

```
1 public class App {  
2     public static void main(String[] args) {  
3         Cliente cliente1 = new Cliente();  
4         cliente1.nome="João Silva";  
5         cliente1.telefone="(87)99999-9999";  
6         cliente1.email="joao@gmail.com";  
7  
8         Cliente cliente2 = new Cliente("Maria Santos", "(87)99999-9999");  
9     }  
10 }
```

A dashed orange rectangle highlights the code area. To the right of the code, there are two annotations:

- "Depois de adicionar os breakpoints" (After adding the breakpoints) with an arrow pointing to the code.
- "Clicar sobre o ícone debug do Eclipse para rodar a App pretendida." (Click on the Eclipse debug icon to run the intended application.) with an arrow pointing to the toolbar.
- "ou" (or)
- "Tecla de Atalho Alt+Shift+D, J" (Shortcut key Alt+Shift+D, J)



Vamos Praticar!

Debug no Eclipse

3)

Modo Debug Ativado

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspaceMPOO2024.2 - Aula2_Slide/src/App.java - Eclipse IDE
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Various icons for file operations, search, and navigation.
- Left Sidebar:** Shows the project structure with "App (1) Java Application" and a thread named "Thread [main] (S)".
- Central Editor:** Displays the code for "App.java":

```
1 public class App {  
2     public static void main(String[] args) {  
3         Cliente cliente1 = new Cliente();  
4         cliente1.nome="João Silva";  
5         cliente1.telefone="(87)99999-9999";  
6         cliente1.email="joao@gmail.com";  
7  
8         Cliente cliente2 = new Cliente("Maria :");  
9     }  
10 }
```
- Right Side Panels:**
 - Variables View:** Shows a variable "cliente1" with the value "<error(s)_during_the_evaluation>".
 - Breakpoints View:** Shows a list of breakpoints.
 - Expressions View:** Shows an option to "Add new expression".
- Bottom Status Bar:** Janelas acompanhamento debug



Vamos Praticar!

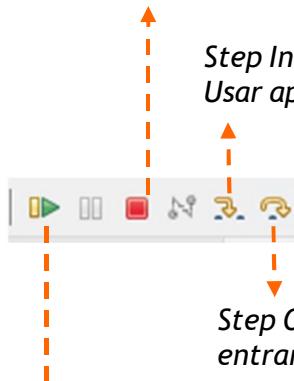
Debug no Eclipse

4)

Uso de controles versus Janelas Debug

Controles

Terminate (Ctrl + F2): Para execução



Step Into (F5): Executa linha e entra na definição.
Usar apenas quando tiver src do método/instrução

Step Over (F6): Executa linha a linha sem
entrar na definição da instrução.

Resume (F8): Executa linhas até o
próximo breakpoint

Janelas Debug

*Enquanto o debug acontece vai-se
acompanhando o resultado da instrução*

Variables - exibe as variáveis do contexto da depuração

Breakpoints - exibe os locais dos breakpoints definidos

Expressions - permite inspecionar algo de interesse

Name	Value
cliente1	(id=24)
email	"joao@gmail.com" (id=37)
nome	"João Silva" (id=27)
telefone	"(87)99999-9999" (id=35)
Add new expression	



Vamos Praticar! *Debug no Eclipse*

Vide mais em:

https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php



Prof. Dr. Richarlyson Alves D'Emery
grupo: http://groups.google.com/group/mpoo_uast
email grupo: mpoo_uast@googlegroups.com
contato: richarlyson.demery@ufrpe.br