DIM0436 — UML I

Richard Bonichon

20140805

Sumário

Definição

- UML = Universal Modeling Language
  - visualizar
  - especificar
  - construir
  - documentar artefatos de um software em desenvolvimento
- "Universal" mas a notação é *orientada a objetos*.
- UML não é
  - uma metodologia
    - ★ não diz quem deve fazer o que, quando e como
    - ★ UML pode ser usado com diferentes metodologias, como RUP
  - uma linguagem de programação

Vantagens

- Padrão aberto

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro
  - modelagem do negócio (processos e objetos do negócio)

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro
  - modelagem do negócio (processos e objetos do negócio)
  - modelagem de requisitos alocados ao software

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro
  - modelagem do negócio (processos e objetos do negócio)
  - modelagem de requisitos alocados ao software
  - modelagem da solução de software

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro
  - modelagem do negócio (processos e objetos do negócio)
  - modelagem de requisitos alocados ao software
  - modelagem da solução de software
- Suporte de diversas áreas de aplicação

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro
  - modelagem do negócio (processos e objetos do negócio)
  - modelagem de requisitos alocados ao software
  - modelagem da solução de software
- Suporte de diversas áreas de aplicação
- Baseada na experiência e necessidades da comunidade de usuários

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro
  - modelagem do negócio (processos e objetos do negócio)
  - modelagem de requisitos alocados ao software
  - modelagem da solução de software
- Suporte de diversas áreas de aplicação
- Baseada na experiência e necessidades da comunidade de usuários
- Ferramentas

Vantagens

- Padrão aberto
  - versão 1.1 aprovada em 1997 pelo OMG
- Suporte do ciclo de vida do software inteiro
  - modelagem do negócio (processos e objetos do negócio)
  - modelagem de requisitos alocados ao software
  - modelagem da solução de software
- Suporte de diversas áreas de aplicação
- Baseada na experiência e necessidades da comunidade de usuários
- Ferramentas
- Suporte industrial

Blocos básicos

UML define 3 tipos básico de contrução:

Itens abstrações de primeira classe de um modelo

Blocos básicos

UML define 3 tipos básico de contrução:

Itens abstrações de primeira classe de um modelo

Relacionamentos definem como os itens relacionam-se

Blocos básicos

UML define 3 tipos básico de contrução:

Itens abstrações de primeira classe de um modelo

Relacionamentos definem como os itens relacionam-se

Diagramas agrupam coleções de itens

Itens

Abstrações de primeira ordem num modelo

Tipos de itens UML

- Estruturais
  - Representar elementos estáticos, concetuais ou físicos
  - Classes, interfaces, componentes, ...
- Comportamentais
  - Representar elementos dinâmicos do modelo
  - Interações, máquinas de estado
- Agrupamentos
  - Partes organizacionais do modelo
  - Pacotes, subsistemas
- Anotacionais
  - Partes explicativas e podem ser associadas a qualquer elemento
  - Notas

Relacionamentos

Associações entre itens

Tipos de relacionamentos

- Dependência
  - Relacionamento entre 2 elementos onde a alteração de um pode afetar o outro
- Associação
  - Relacionamento estrutural descrevendo um conjunto de ligações entre objetos
- Generalização
  - Relacionamento em que elementos especializados são substituíveis por elementos generalizados
- Realização
  - Relacionamento semântico entre itens que satisfazem um contrato

Diagramas

- Diagrama = representação parcial do sistema
- Diagrama = da uma compreensão específica do sistema de uma visão dada

Tipos

- Classe
- Objetos
- Casos de uso
- Sequências
- Colaborações
- Estados
- Atividades
- Componentes
- Implantação

Modelos UML

Modelos estáticos

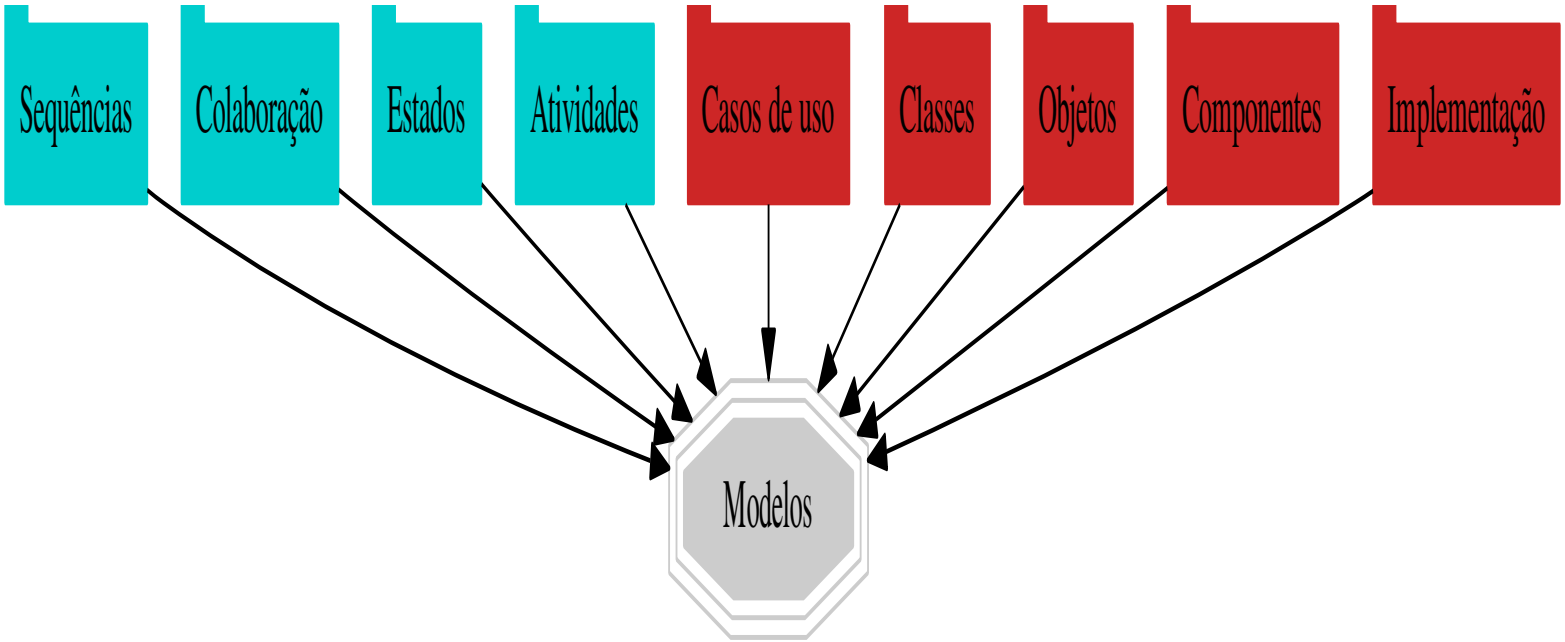
Uma notação para elementos estáticos do software: classes, atributos, relações

- Diagramas de classes
- Objetos
- Componentes
- Implementação

Modelos dinâmicos

- Diagramas de colaboração
- Diagramas de sequência
- Diagramas de estados
- Diagramas de atividades

Modelos, diagramas e visões



Antes de começar

There are lots of widgets, adornments, and whatchamajiggers in UML. There are so many that you can spend a long time becoming an UML language lawyer enabling you to do what all lawyers can – write documents nobody can understand. . . . Using too little of UML is almost always better than using too much.
– Robert C. Martin, UML for Java Programmers

Objetivos

- Modelizar uma visão estática do sistema
- Diagramas mais usados/conhecidos da UML

Uso

- Modelizar o vocabulário do sistema
  - Quais abstrações utilizadas
- Modelizar colaborações simples
- Modelizar o esquema lógico (banco de dados)

Classe

Definição

- Uma classe é uma descrição de um conjunto de elementos de objetos com os mesmos
  - atributos
  - operações
  - relacionamentos
  - semântica
- Classes são blocos básicos para elaboração de sistemas orientados a objetos.

Composição

- Nome

Classe

Definição

- Uma classe é uma descrição de um conjunto de elementos de objetos com os mesmos
  - atributos
  - operações
  - relacionamentos
  - semântica
- Classes são blocos básicos para elaboração de sistemas orientados a objetos.

Composição

- Nome
- Atributos

Classe

Definição

- Uma classe é uma descrição de um conjunto de elementos de objetos com os mesmos
  - atributos
  - operações
  - relacionamentos
  - semântica
- Classes são blocos básicos para elaboração de sistemas orientados a objetos.

Composição

- Nome
- Atributos
- Operações

Atributos

Definição

- Um atributo representa alguma propriedade da classe modelizada
- Todos os objetos da classe compartilham os atributos

Composição

- Nome

Atributos

Definição

- Um atributo representa alguma propriedade da classe modelizada
- Todos os objetos da classe compartilham os atributos

Composição

- Nome
- Tipo

Atributos

Definição

- Um atributo representa alguma propriedade da classe modelizada
- Todos os objetos da classe compartilham os atributos

Composição

- Nome
- Tipo
- Visibilidade

Atributos

Definição

- Um atributo representa alguma propriedade da classe modelizada
- Todos os objetos da classe compartilham os atributos

Composição

- Nome
- Tipo
- Visibilidade
- Valor inicial

Operações

Definição

- Ações que uma classe sabe realizar
- Correspondência com métodos

Composição

- Nome

Operações

Definição

- Ações que uma classe sabe realizar
- Correspondência com métodos

Composição

- Nome
- Parâmetros

Operações

Definição

- Ações que uma classe sabe realizar
- Correspondência com métodos

Composição

- Nome
- Parâmetros
- Visibilidade

Operações

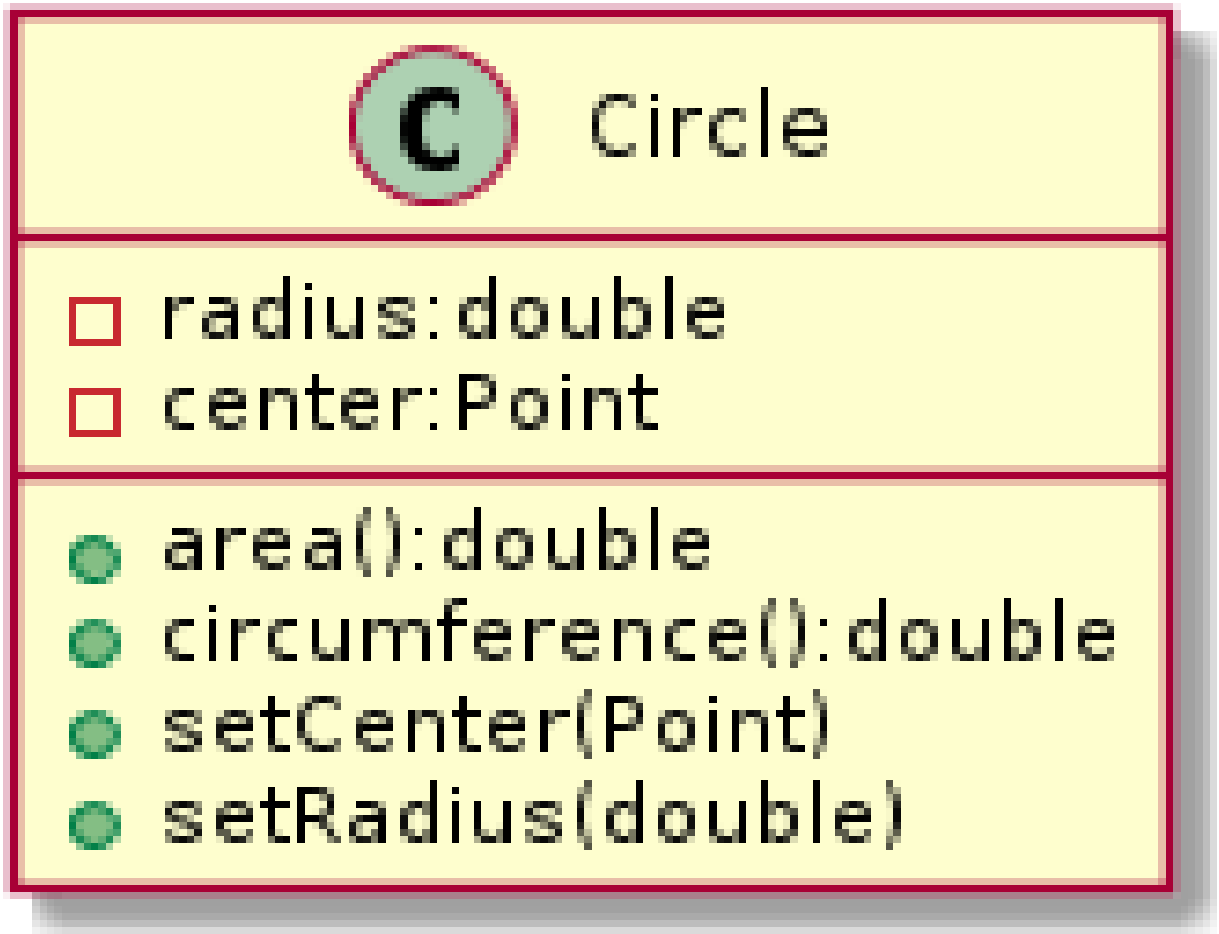
Definição

- Ações que uma classe sabe realizar
- Correspondência com métodos

Composição

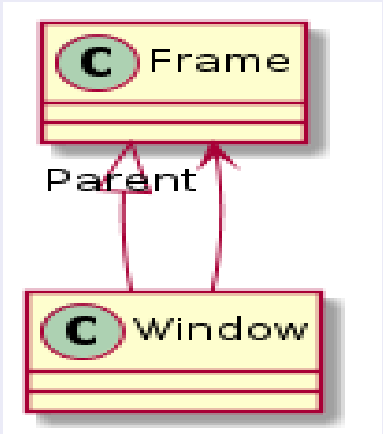
- Nome
- Parâmetros
- Visibilidade
- Tipo de retorno

Exemplo: classe `Circle`



Associação

Notação



Multiplicidade

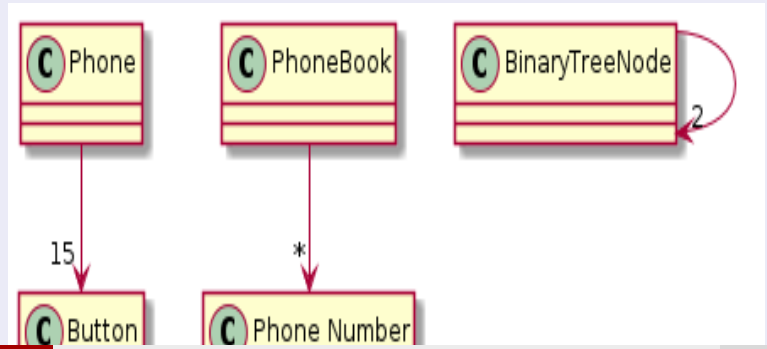
Objetivo

Representar a quantidade de objetos que podem ser conectados por uma instância da associação.

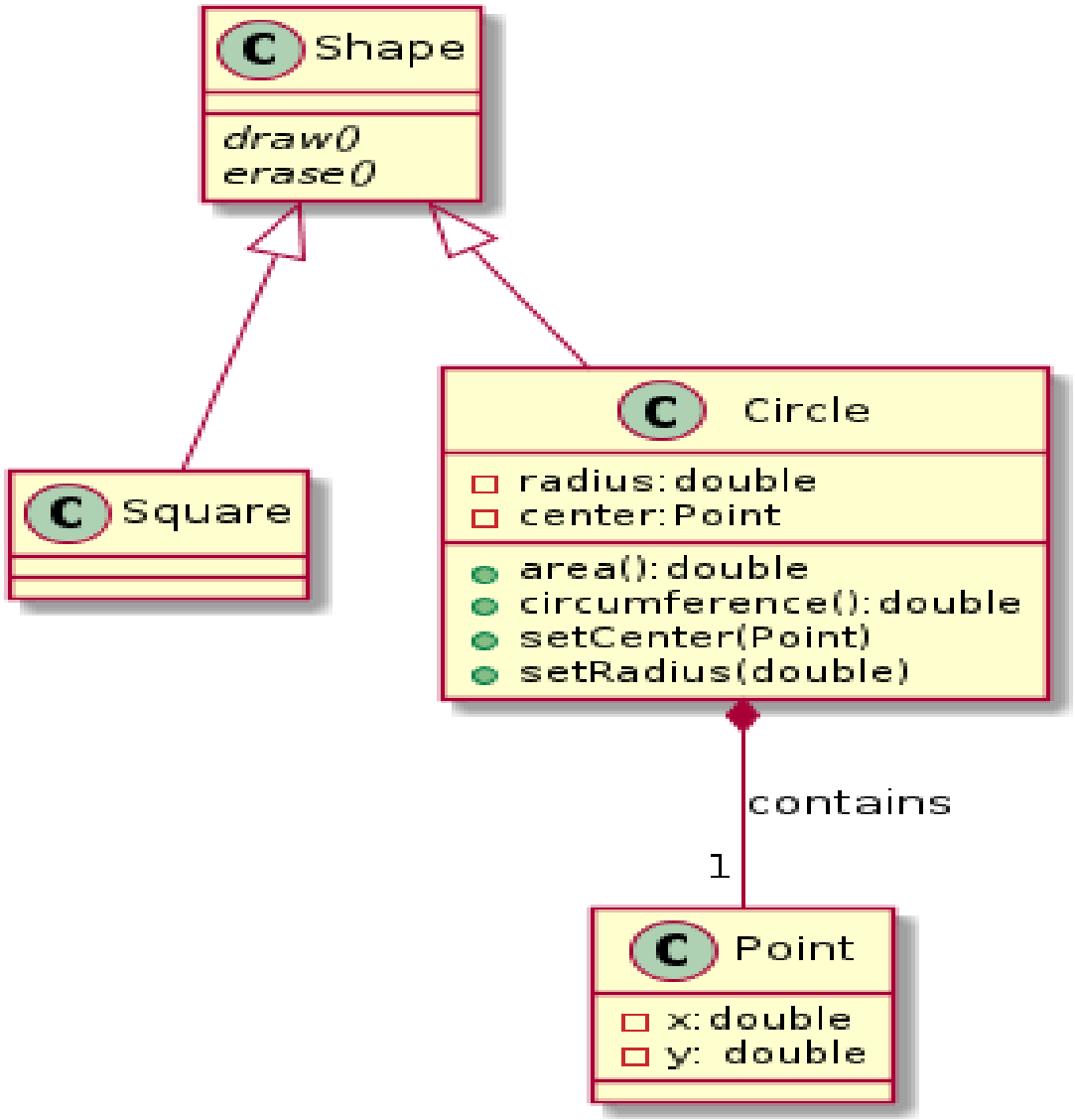
Formas

Notação	Semântica
* ou 0..*	zero to many
0..1	zero ou um
1..*	one to many
3..5	três a cinco

Exemplo



Generalização (Herança)

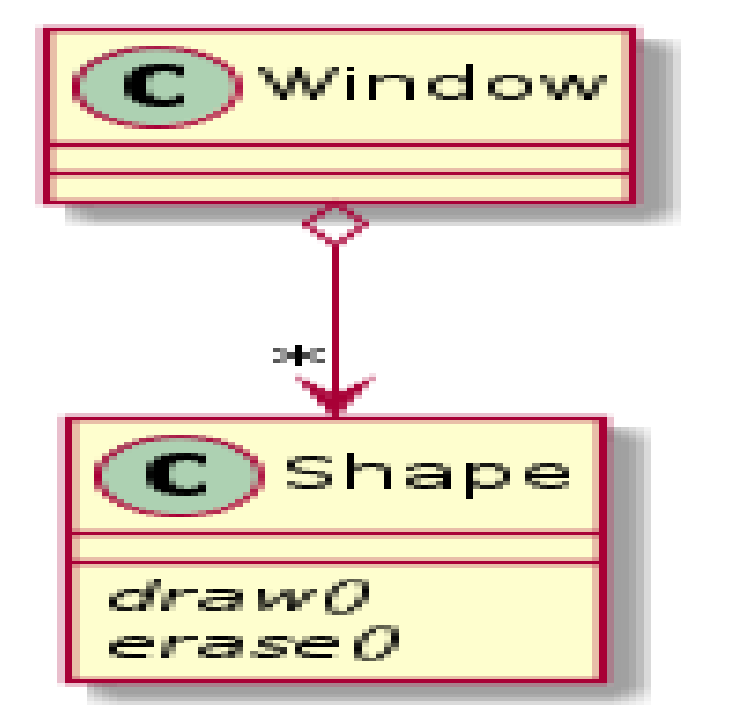


Agregação

Definição

- A classe agregada (com ◇) é o *todo* é a outra classe da relação faz *parte* do todo.
- Um todo não podo fazer parte dele mesmo (i.e. não tem ciclos)

Notação



Semi-formalidade

Diferença

- Agregação = relação parte/todo
- Associação não

Citações

. . . if you don't understand aggregation don't use it
– UML 0.8

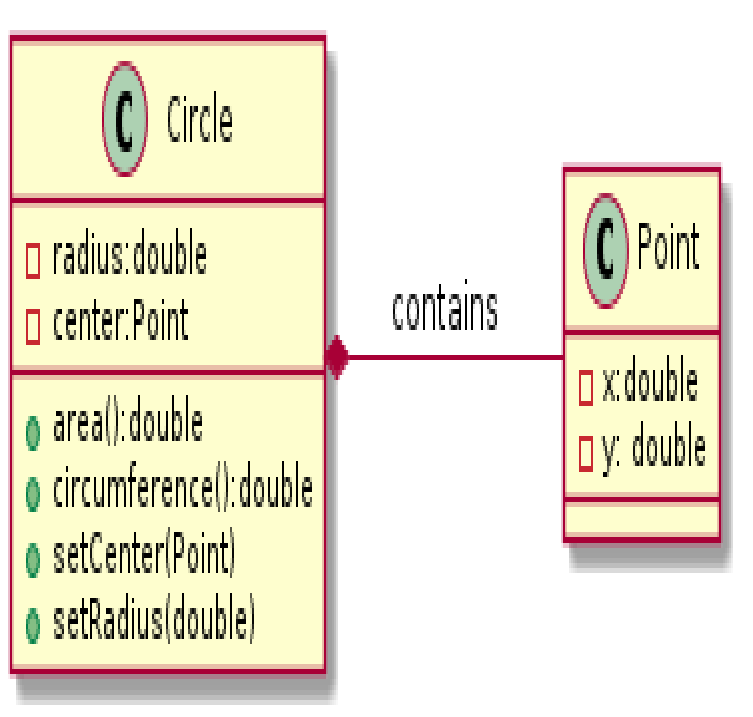
Unfortunately, UML does not provide a strong definition for this relationship. This eads to confusion because various programmers and analysts adopt their own pet definitions for the relationship. For that reason, I don't use the relationship at all; and I recommend that you avoid it as well.
– Robert C. Martin

Composição

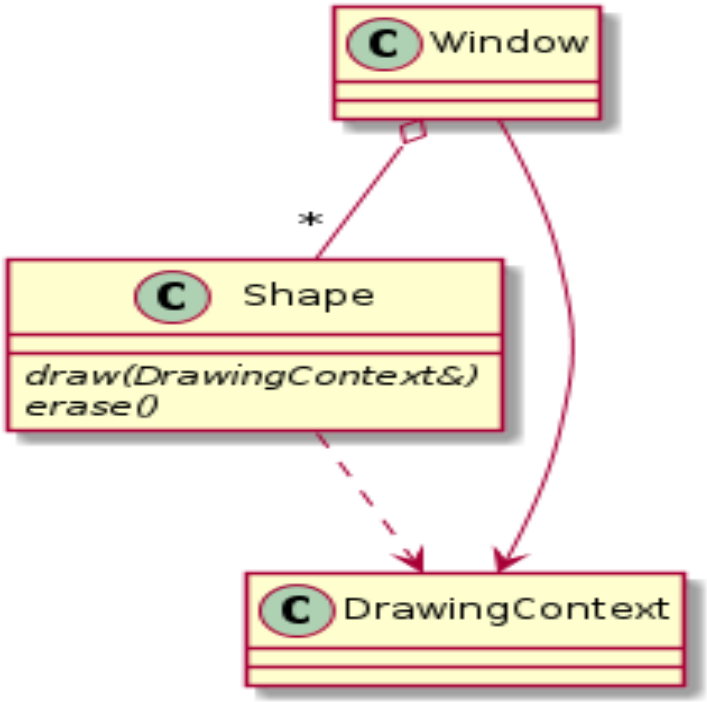
Composição

- Instâncias do tipo `Circle` contem uma instância do tipo `Point`.
- Esta relação é chamada de composição
- É um tipo especial de agregação

Diagrama



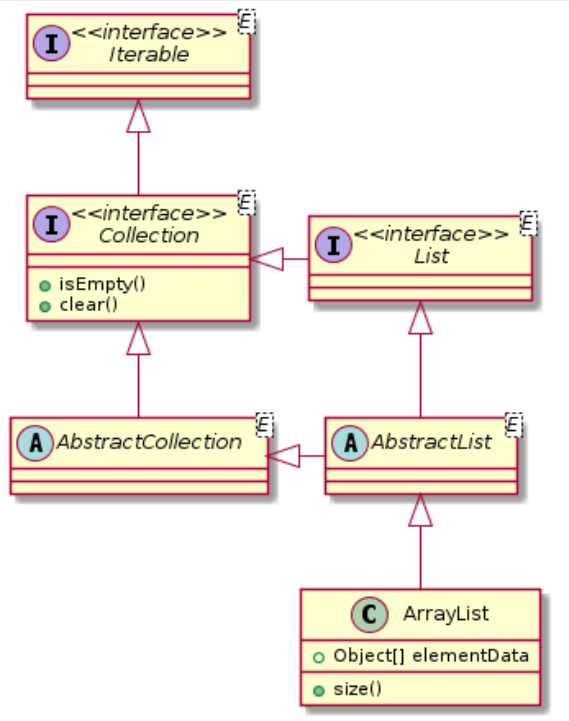
Dependência



Interface

Algumas classes são só puras funções virtuais. Em `Java`, é chamado de `interface`

Notação



Exemplo de código

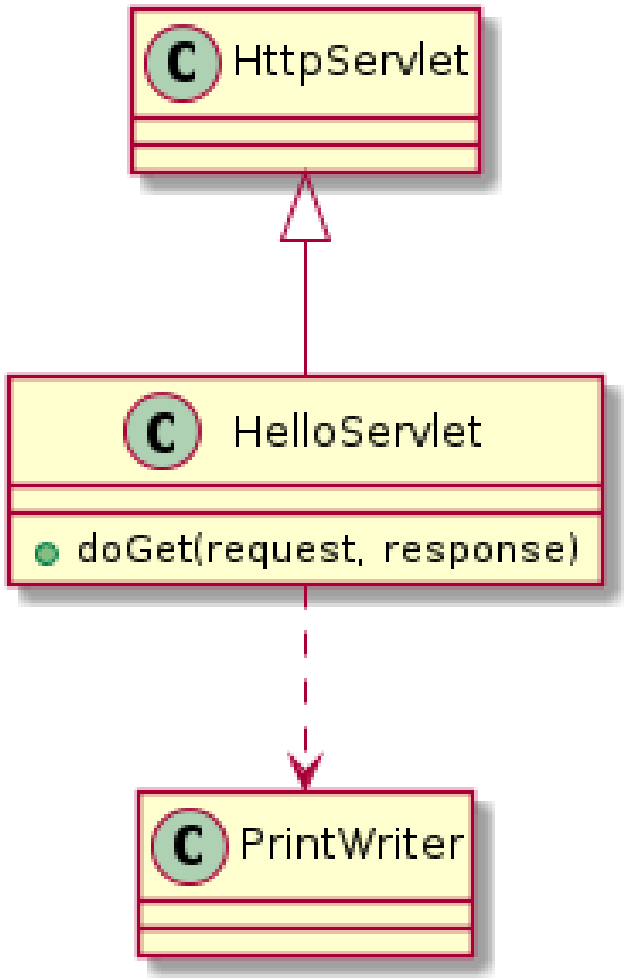
```java
package servlet.test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException
  {
    response.setContentType("text/plain");
    PrintWriter out = response.getWriter();
    out.println("Hello World");
  }
}
```

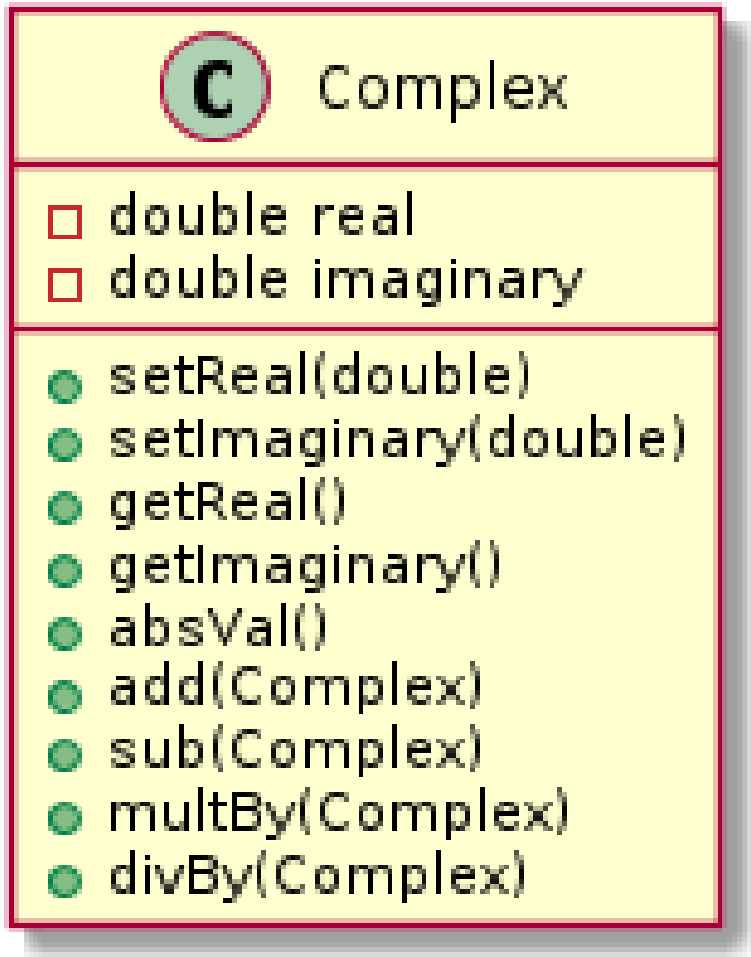
Diagrame de classes



Exercício

- Desenhe uma diagrama de classe para a classe `Complex`.
- Um objeto `Complex` tem
  - partes real e imaginária privadas (do tipo `double`)
  - e pode executar
    - adição,
    - subtração,
    - multiplicação e
    - divisão por um outro número complexo.

Solução


Complex
double real
double imaginary
setReal(double)
setImaginary(double)
getReal()
getImaginary()
absVal()
add(Complex)
sub(Complex)
multBy(Complex)
divBy(Complex)

Propriedades

Objetivos

- Descrever o comportamento do sistema
- Descrição conforme o ponto de vista do usuário
- Modelizar o contexto do sistema
- Modelizar os requisitos do sistema
- Principais funcionalidades do sistema são nestes diagramas
- Comunicação cliente/analistas

Notação

- Atores

Notação

- Atores
- Casos de uso

Notação

- Atores
- Casos de uso
- Relacionamentos entre elementos

Notação

- Atores
- Casos de uso
- Relacionamentos entre elementos
  - Associação atores/casos de uso

Notação

- Atores
- Casos de uso
- Relacionamentos entre elementos
  - Associação atores/casos de uso
  - Generalização de atores

Notação

- Atores
- Casos de uso
- Relacionamentos entre elementos
  - Associação atores/casos de uso
  - Generalização de atores
  - Generalização de casos de uso

Notação

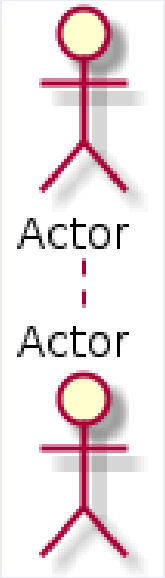
- Atores
- Casos de uso
- Relacionamentos entre elementos
  - Associação atores/casos de uso
  - Generalização de atores
  - Generalização de casos de uso
  - Dependências de inclusão e extensão (*estereótipos*)

Ator

Definição

- Um ator representa um conjunto coerente de papéis que os usuários fazem quando usam o sistema.
- Pode ser um humano ou um outro sistema

Representação

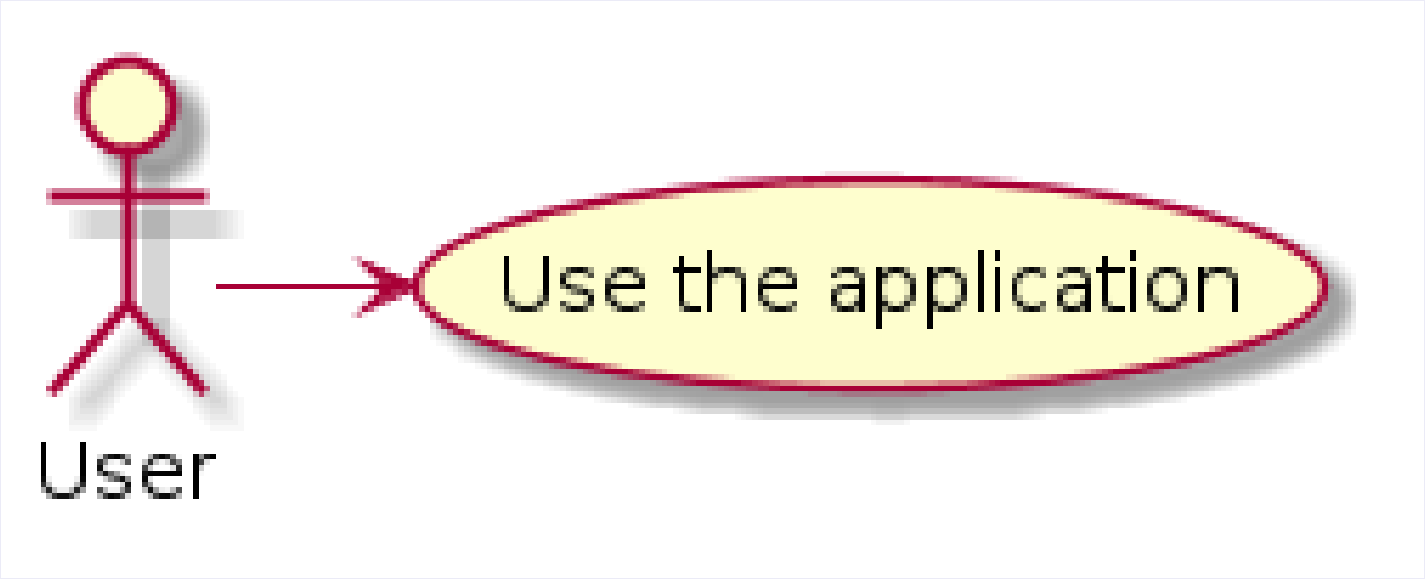


Casos de uso

Objetivo

Definir uma funcionalidade do sistema do ponto de vista do ator

Representação



Relacionamentos

Objetivos

- Descrição dos casos de uso
  - ator/caso de uso
  - caso de uso/caso de uso
- Formas
  - Generalização
  - Associação
  - Dependências (inclusões/exclusões)

Robert C. Martin

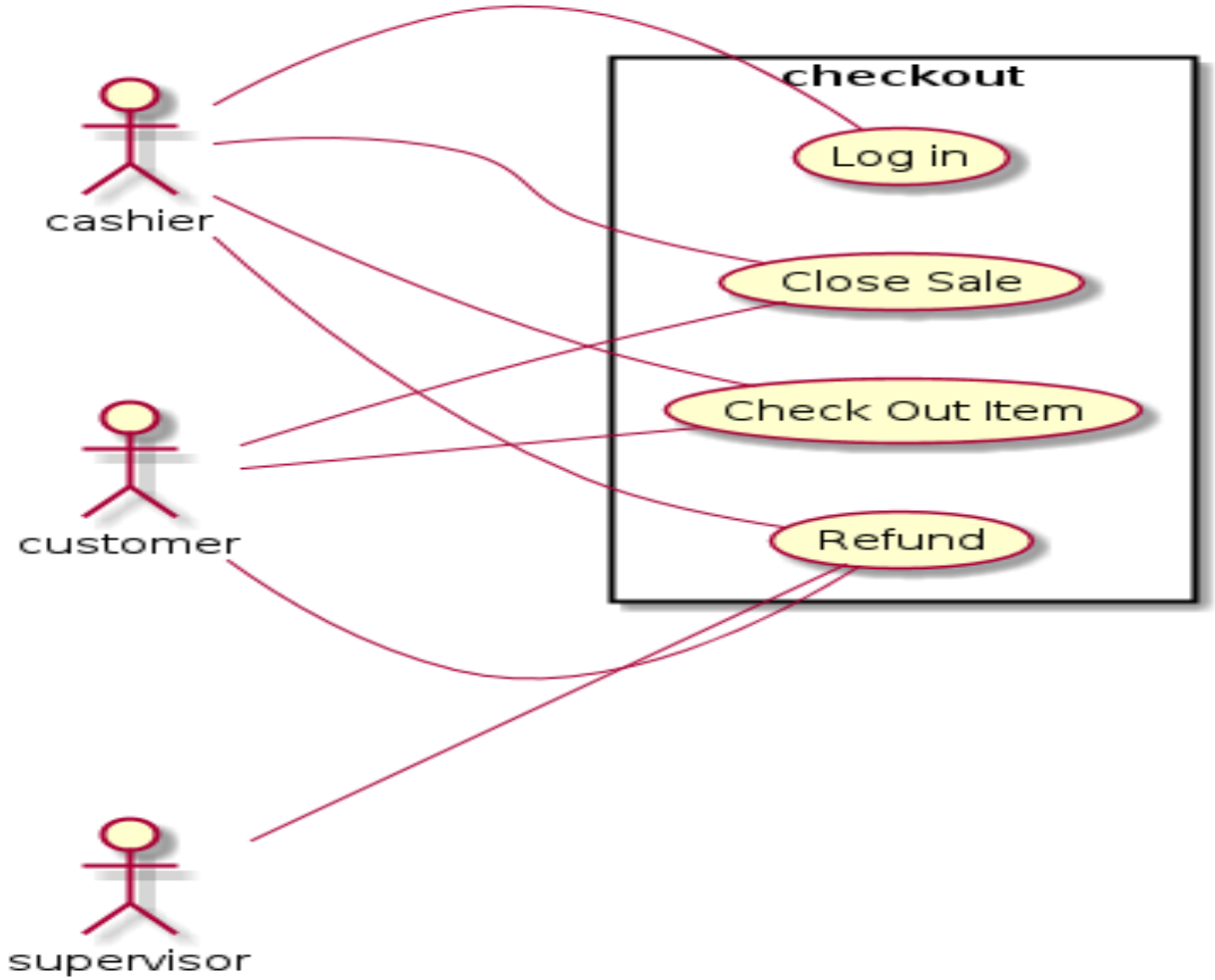
Use case relationships fall into the category of things that "seemed like a good idea at the time". I suggest that you actively ignore them. They'll add no value to your use cases, or to your understanding of the system, and they will be the source of many never ending debates about whether or not to use «extends» or «generalization».

Exemplos: sistema de vendas

Check Out Item

1. Cashier swipes product over scanner, scanner reads UPC code.
2. Price and description of item, as well as current subtotal appear on the display facing the customer. The price and description also appear on the cashier's screen.
3. Price and description are printed on reciept.
4. System emits an audible "acknowledgement" tone to tell the cashier that the UPC code was correctly read.

Exemplos


checkout
Log in
Close Sale
Check Out Item
Refund
cashier
customer
supervisor

Observações

Dicas para casos de uso UML

- KISS

Observações

Dicas para casos de uso UML

- KISS
- Tomorrow they are going to change

Para terminar

Of all the diagrams in UML, use case diagrams are the most confusing, and the least useful.
– Robert C. Martin, UML for Java Programers

Observações

Dicas para casos de uso UML

- KISS
- Tomorrow they are going to change
- Casos de uso = requisitos *Just-In-Time*

Para terminar

> *Of all the diagrams in UML, use case diagrams are the most confusing, and the least useful.*
> *– Robert C. Martin, UML for Java Programers*

Definição

Objetivo

- Descrição do fluxo de mensagens entre objetos
- Focado nas relações entre objetos
- Visualizar como objetos colaboram para efetuar o trabalho

Observação

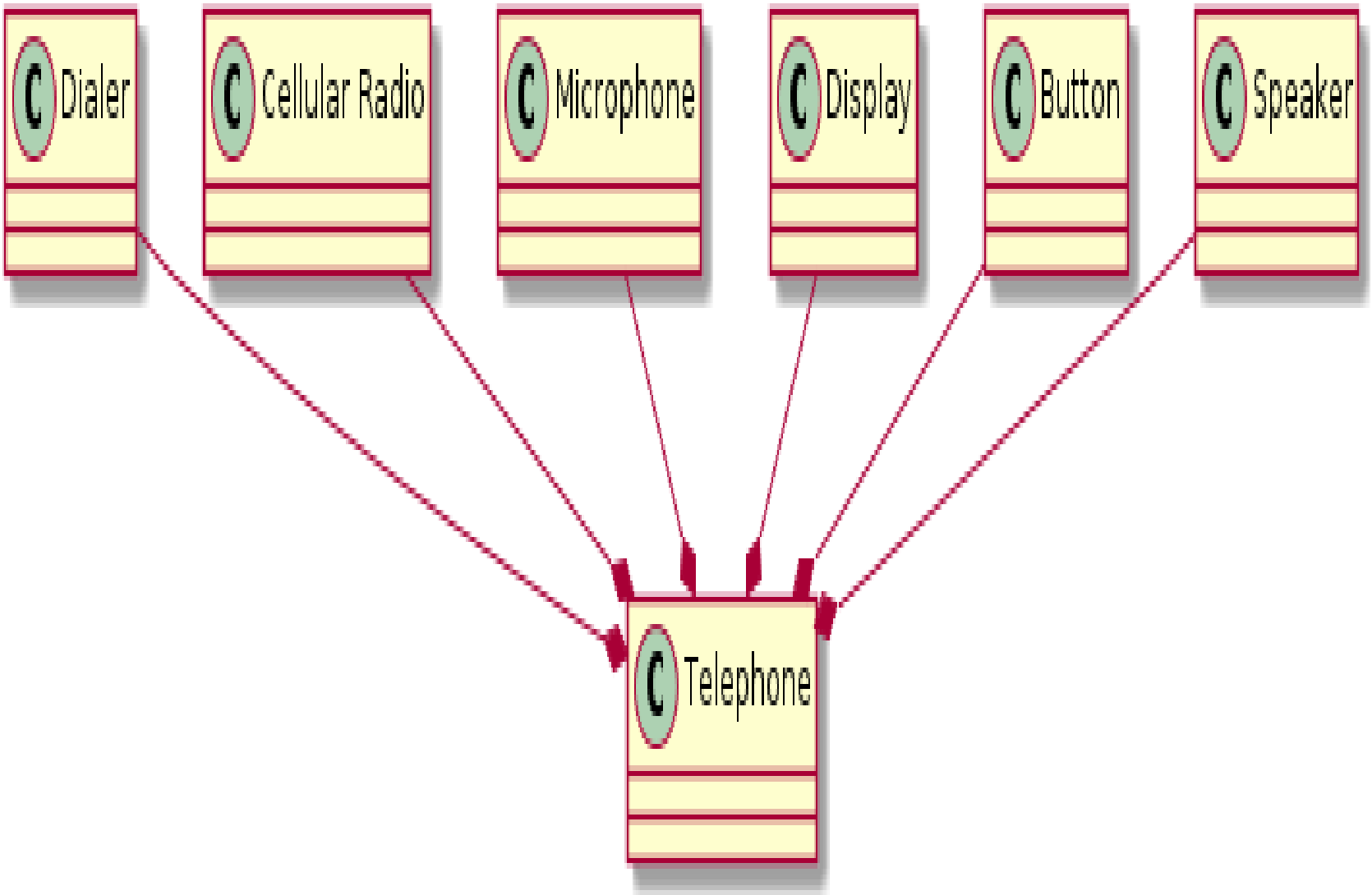
Diagramas de colaboração e diagramas de sequências podem ser traduzido um para o outro.

O software de um celular

Definição (Um celular muito simples)

- Teclas para digitar
- Tecla "Enviar" para iniciar uma chamada
- Hardware para "discar"
- Software para reunir os dígitos a serem discados, e emitir os tons apropriados
- Tem um rádio celular para tratar a conexão à rede celular
- Tem um microfone, um alto-falante e uma tela.

Diagrama de classe

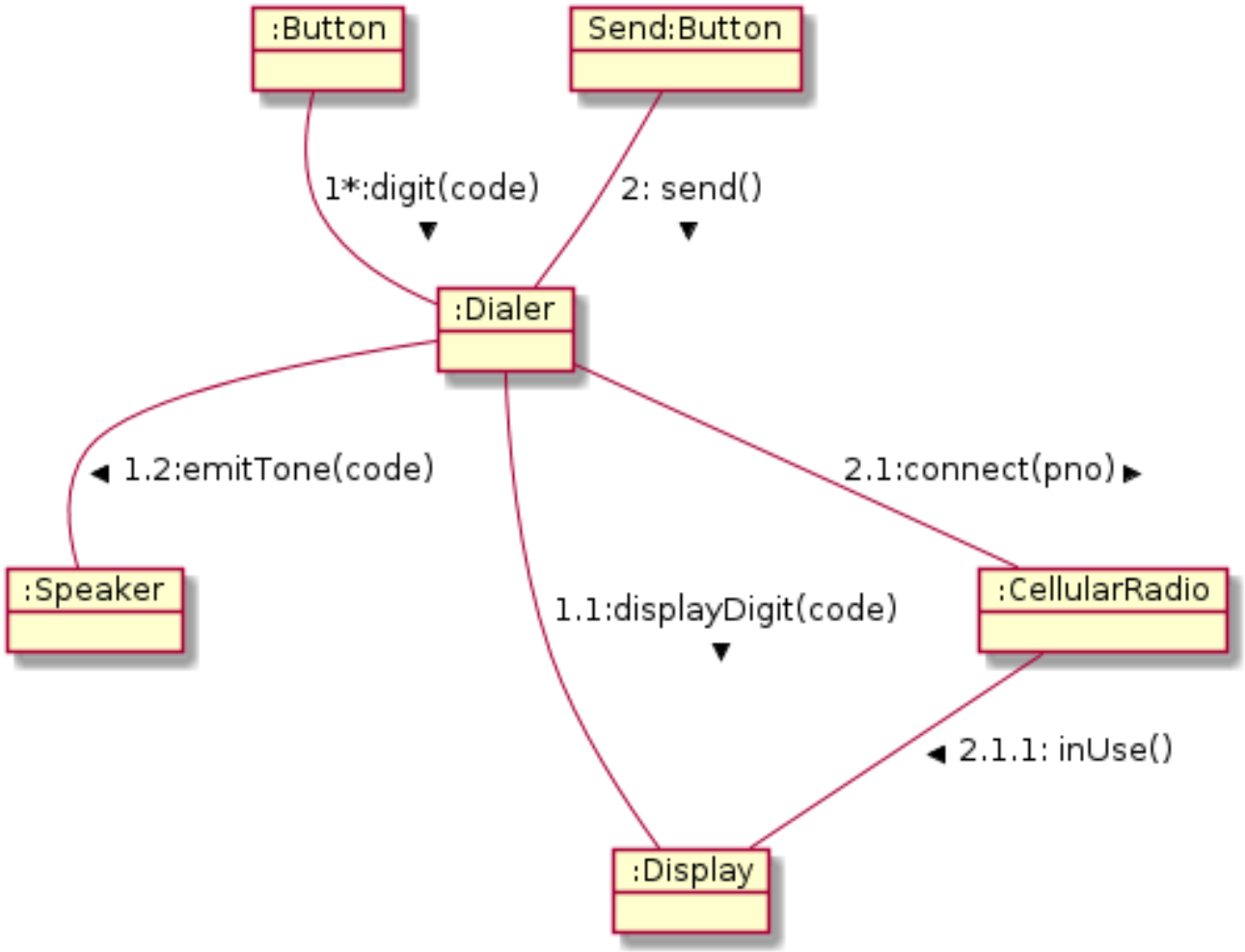


Especificação: celular

Use case: Make Phone Call

1. User presses the digit buttons to enter the phone number.
2. For each digit, the display is updated to add the digit to the phone number.
3. For each digit, the dialer generates the corresponding tone and emits it from the speaker.
4. User presses "Send"
5. The "in use" indicator is illuminated on the display
6. The cellular radio establishes a connection to the network.
7. The accumulated digits are sent to the network.
8. The connection is made to the called party.

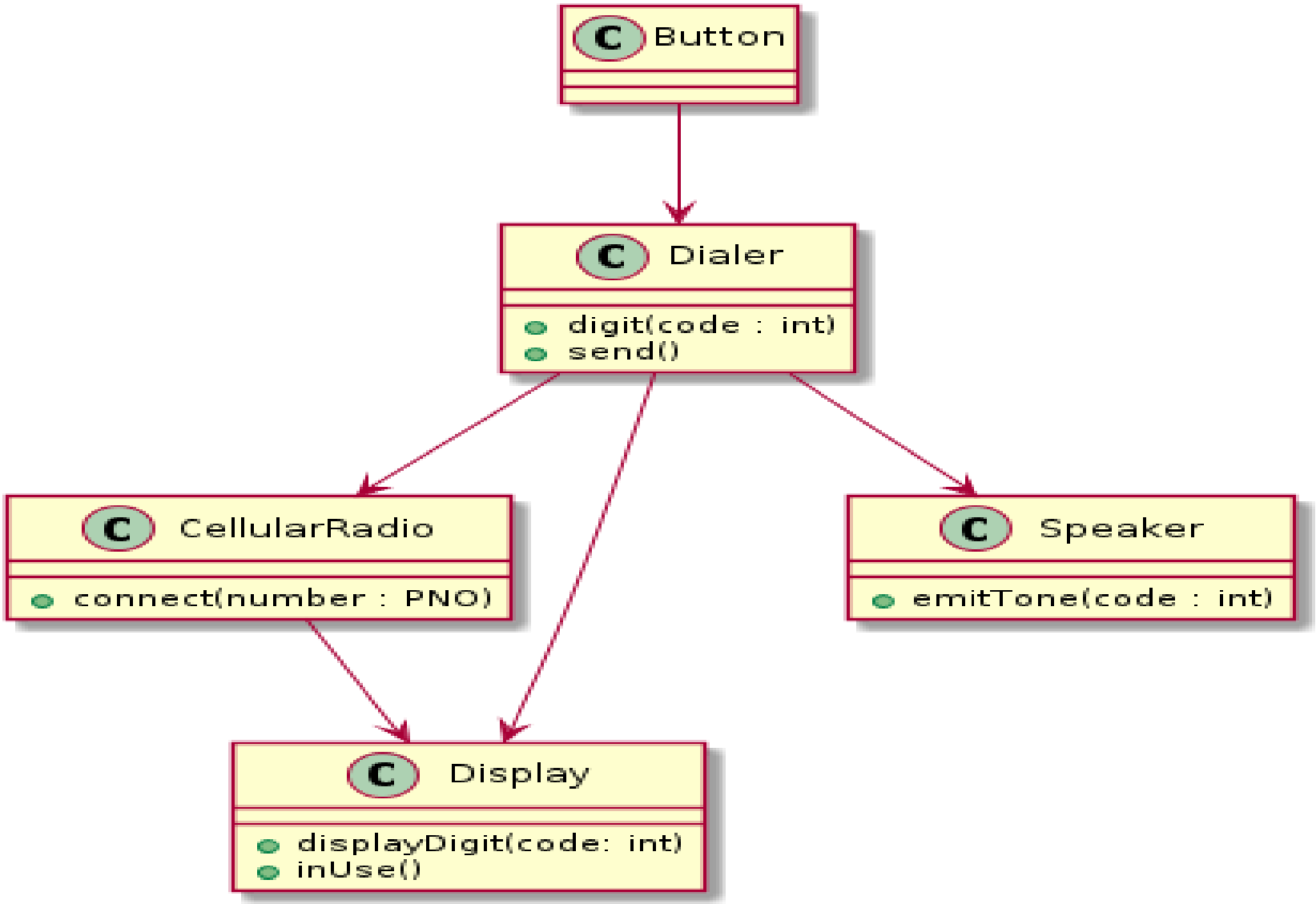
Diagrama


:Button
Send:Button
1*:digit(code)
2: send()
:Dialer
1.2:emitTone(code)
2.1:connect(pno)
:Speaker
1.1:displayDigit(code)
:CellularRadio
2.1.1: inUse()
:Display

Sintaxe

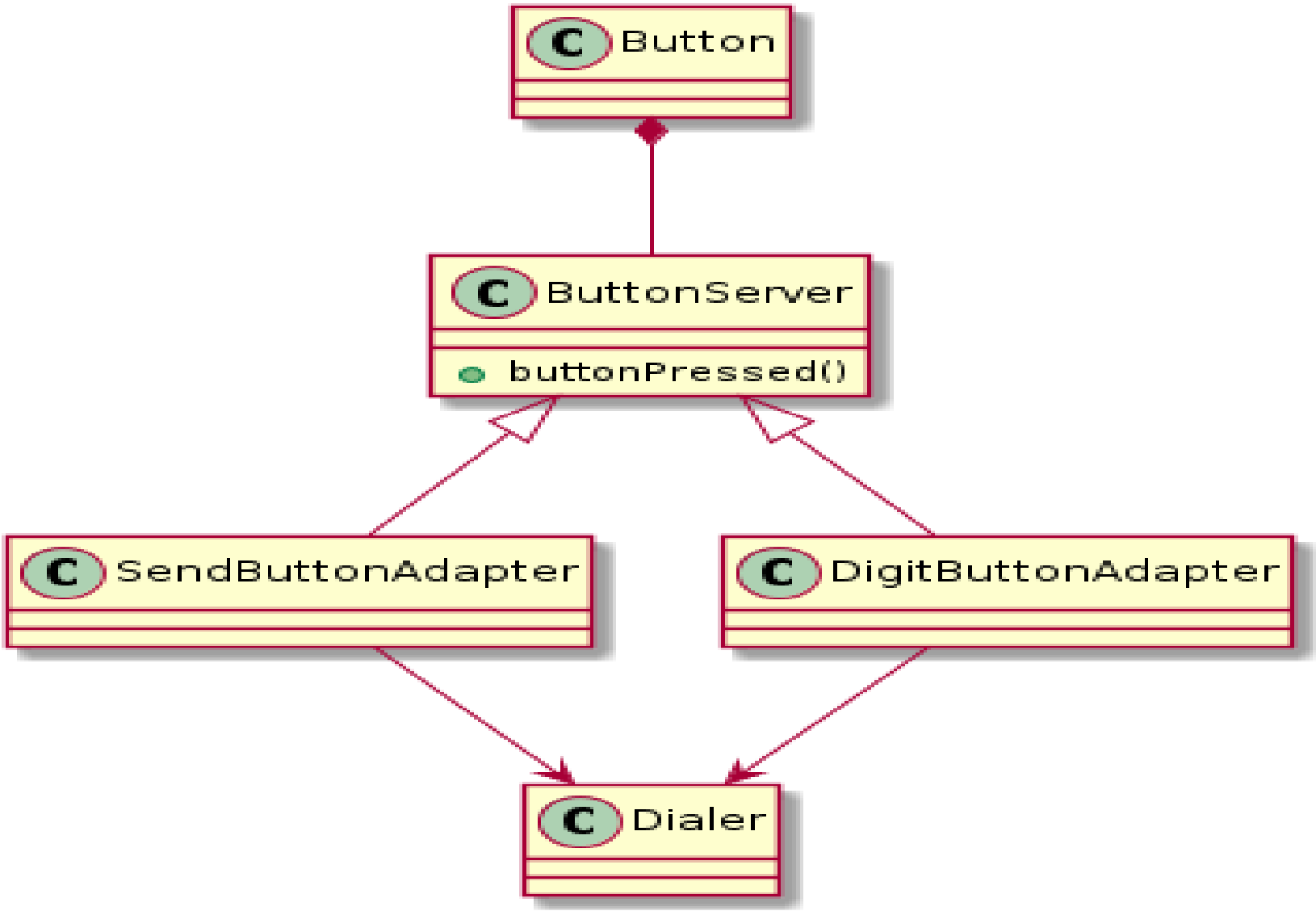
- Elementos do diagrama são instâncias de objetos e não classes.
  - `Send:Button` = objeto `Send` da classe `Button`
  - `:Button` = objeto anônimo da classe `Button`
- Linhas = links
  - links são instâncias de associações
  - se não tiver associação no diagrama de classe , não botar link no diagrama colaboração.
- Setas = mensagens com um rótulo dando o nome, o número na sequência e os parâmetros

Voltando ao modelo estático

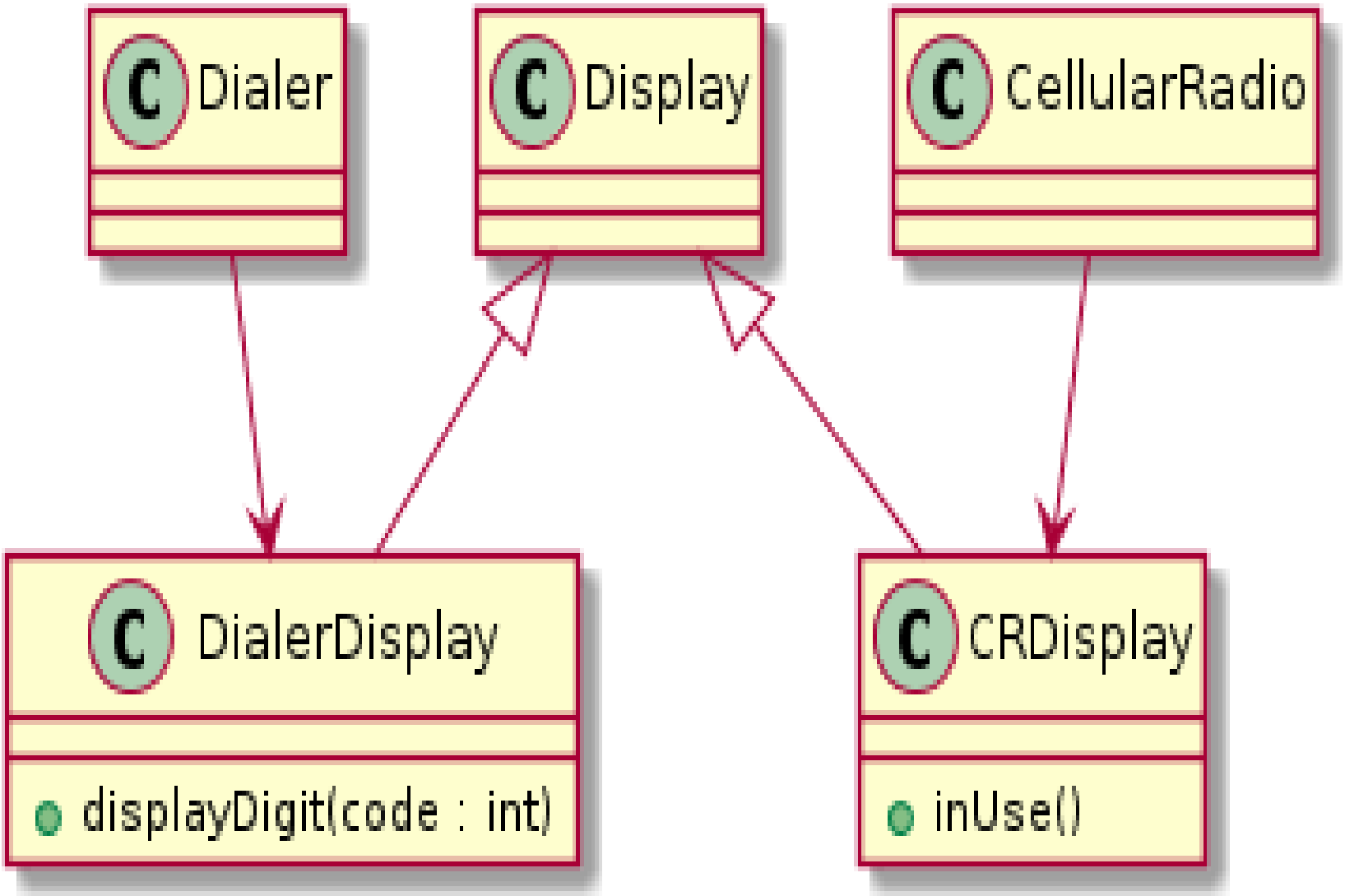


Investigando o modelo estático

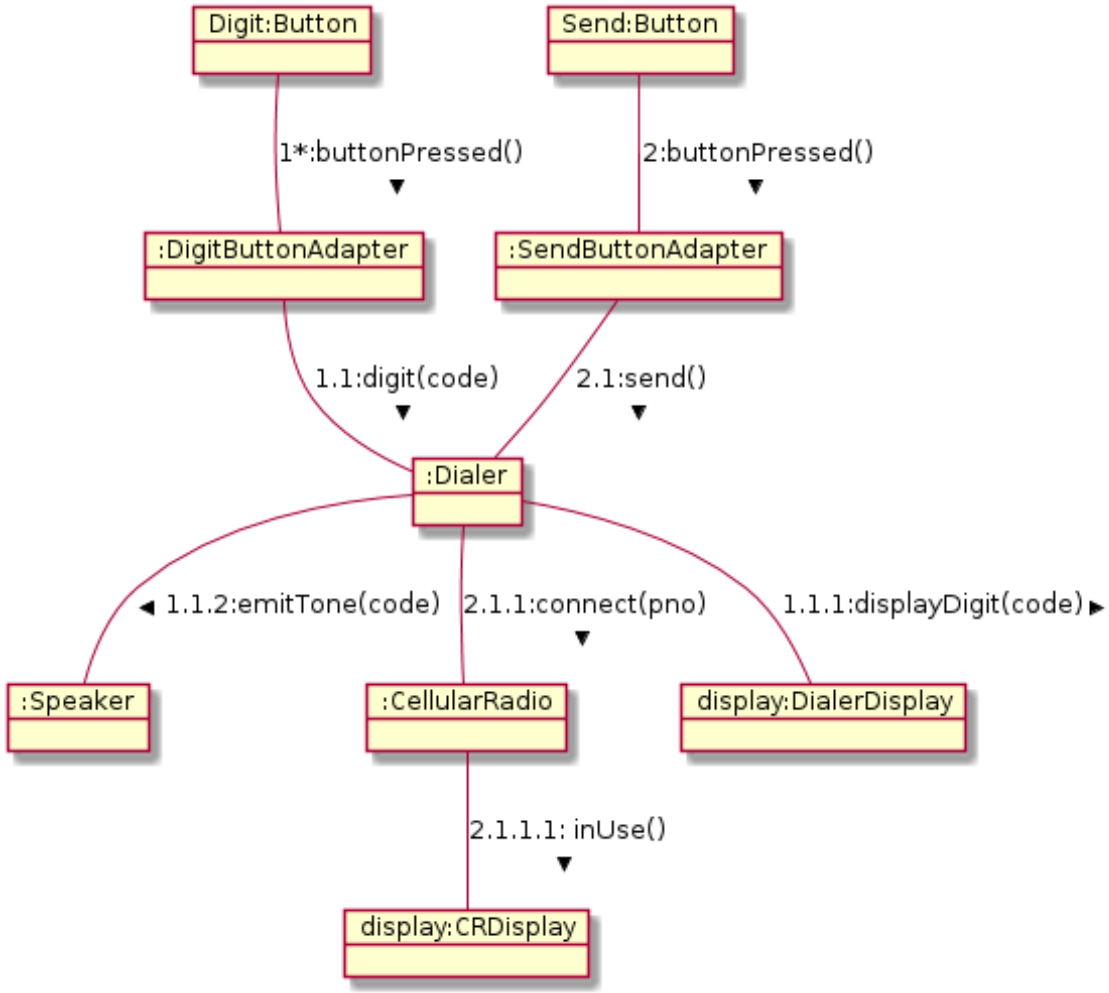
- `Button` deveria ser reusável em outros programas sem `Dialers` ?



Separação de Display



Iteração no modelo dinâmico



Ferramentas

- Livre
  - PlantUML
  - ArgoUML
  - StarUML 2
  - Eclipse UML2 Tools
- Proprietário
  - Borland Together
  - Rational Rose Modeler

Resumo

- Introdução
- Diagramas de classes
- Diagramas de colaborações
- Diagramas de sequências
- Diagramas de estados
- Ferramentas

Referências

- *Site da UML*, `http://www.uml.org`.
- Grady Booch, James Rumbaugh, and Ivar Jacobson, *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*, Addison-Wesley Professional, 2005.
- Martin Fowler, *UML Distilled (3rd Ed.): A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley, Boston, MA, USA, 2003.
- R.C. Martin, *UML for Java programmers*, Robert C. Martin series, Prentice Hall PTR, 2003.

Perguntas ?

http://dimap.ufrn.br/~richard/dim0436