Modelação e Design 15: Diagrama de Classes

Leonor Melo

leonor@isec.pt

1

Diagrama de classes

- Classes abstratas
- Interfaces
- Templates

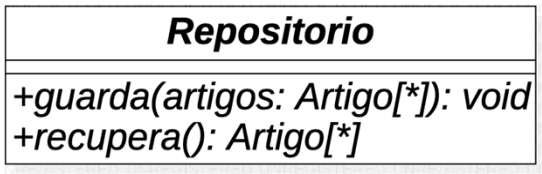
2

Classes abstratas

- Aplica-se a
  - classes mais genéricas
  - desenhadas para serem reutilizadas
    - agregam atributos e comportamentos comuns às subclasses
- Algum do comportamento da classe mais genérica é desconhecido
  - Sei que o comportamento deve existir
  - Mas implementação depende exclusivamente da classe derivada

3

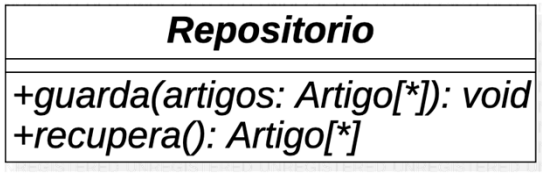
Classes abstratas



- Métodos abstratos
  - assinatura escrita em itálico
  - não contém implementação
  - "espaço reservado"
    - a implementação será feita pelas subclasses

4

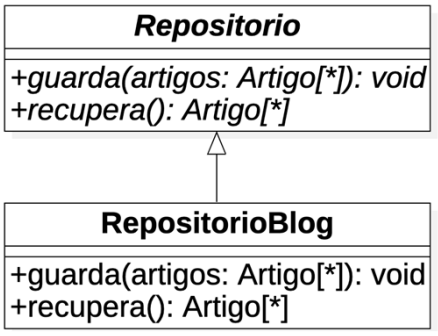
Classes abstratas



- Se algum dos métodos for abstrato
  - a classe é abstrata
- Classe abstrata
  - Não pode ser instanciada

5

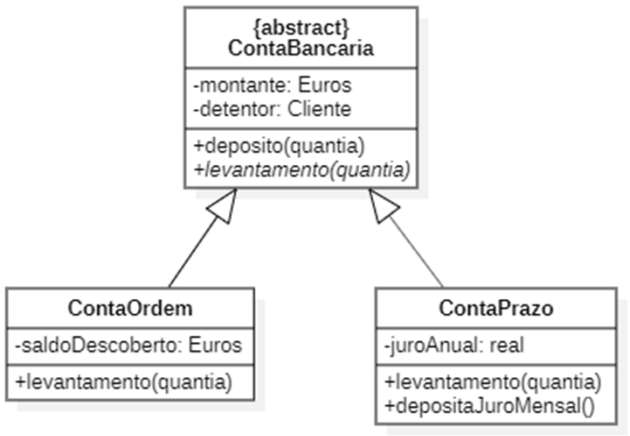
Classes abstratas



- Subclasse de classe abstrata
  - Ou implementa os todos os métodos abstratos
  - Ou é também abstrata

6

Classes abstratas



- normalmente
  - tem atributos
  - apenas parte das operações são abstratas

7

Classes abstratas

- Classes abstratas
  - Mecanismo que promove a reutilização
    - Define-se atributos e operações que deverão existir / ser garantidos
      - Ainda que os detalhes venham a ser definidos pelas sub- classes
  - Usadas frequentemente em design patterns
  - Usa relação de herança
    - Acoplamento forte
      - Usar com cuidado

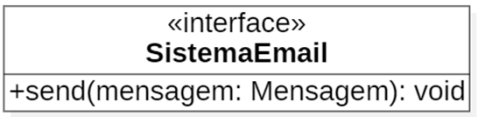
8

Interfaces

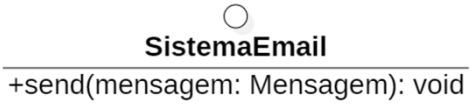
- Representa um tipo de comportamento que determinada classe pederá, ou não, ser capaz de fornecer
- Conjunto de operações
  - sem implementação dos métodos
- "Contrato" que indica
  - a assinatura das operações que devem existir
- Raramente contém atributos
  - e apenas estáticos ou constantes
- Não pode ser instanciada

9

Notação UML



Notação com esteriótipo

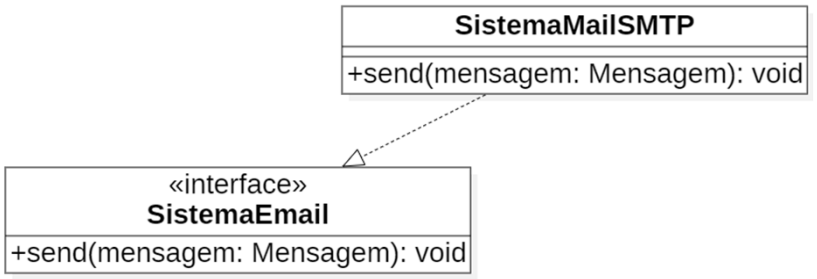


Notação com "bola"

10

Relação de realização

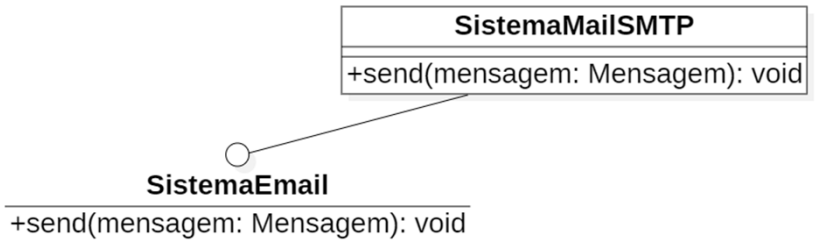
- "relação de realização"
  - Estabelecida entre uma classe e um interface
    - A classe esté em conformidade com o contrato especificado pela interface



11

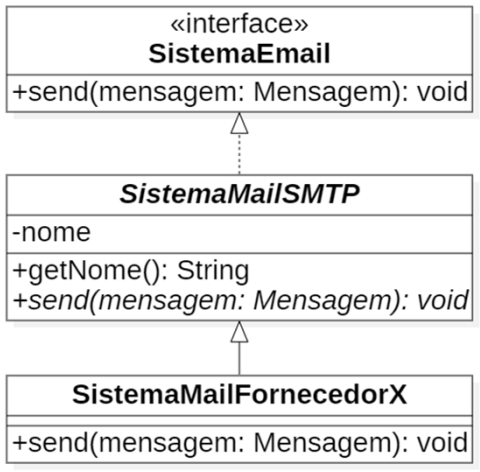
Relação de realização

- representação UML
  - na notação de bola
    - linha de associação
  - Notação com esteriótiopos
    - seta de realização



12

Relação de realização



- Uma classe que realize um interface mas não implemente todos os métodos do interface tem de ser declarada abstrata

13

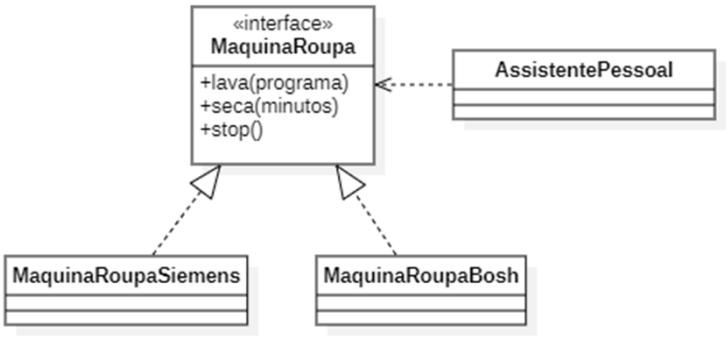
Interface: relação de dependencia

- Os interfaces existem para que as classes dependam dos interfaces e não de classes específicas (abstração, redução de acoplamento)
- Uma classe A depende de um interface se
  - é indiferente os detalhes da classe que na realizade vai executar os métodos
  - a classe A sabe que a classe que eventualmente implementar o interface está preparada para receber mensagens com determinada assinatura

14

Interface: relação de dependencia

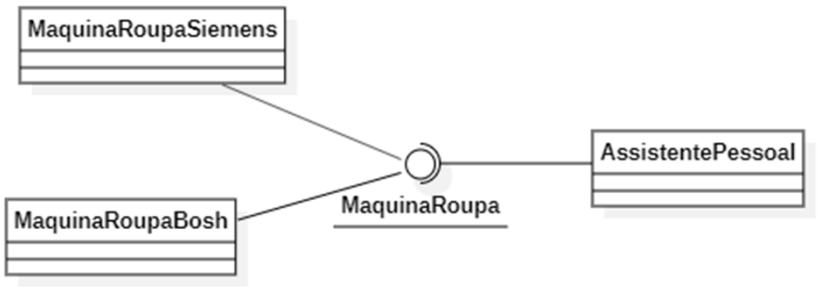
- O assitente sabe como comunicar com uma máquina da roupa, desde que essa máquina realize o interface MaquinaRoupa



15

Interface: relação de dependencia

- Notação de bola:
  - Tanto pode comunicar com uma máquina Bosh como com uma Siemens (ou outras que entretanto realizem o interface)



16

Interfaces

- Uma classe pode realizar vários interfaces
- Evita problemas da generalização múltipla
  - porquê?
- Semelhantes a classes abstratas
  - mas com acoplamento mais fraco
  - usadas para "desacoplar" classes
- Usadas frequentemente em design patterns

17

Interfaces

- Separar o comportamento que a classe deve ter
  - da implementação desse comportamento
- Se uma classe implementar um interface
  - objetos dessa classe podem ser referidos pelo nome do interface
- As outras classes podem passar a depender do interface
  - e não da classe que o implementa

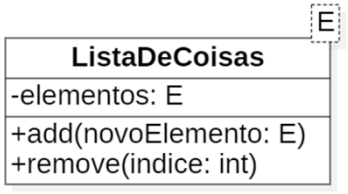
18

Templates

- classe parameterizada
  - útil quando se quer adiar a decisão sobre com que classe(s) a nossa classe irá trabalhar
- sei que a minha classe irá trabalhar com outras
  - mas não quero ter de especificar exatamente com qual

19

Templates



- E não é nenhuma classe existente no modelo
  - está apenas no lugar onde aparecerá a classe cujos objetos serão guardados na lista

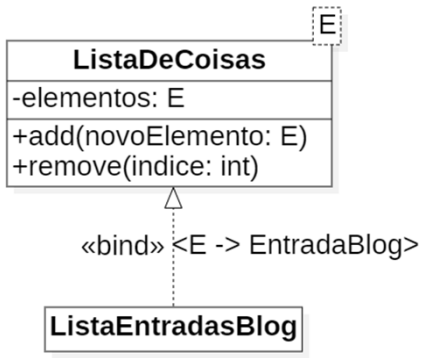
20

Usar Templates

- usar uma classe template
  - fazer o *bind* (ligar) dos parâmetros
- Fazer o *bind*:
  - *binding* por subclasse
  - *binding* em *runtime*

21

Usar Templates: *binding* por subclase



- ListaEntradasBlog:
  - comportamento genérico de ListaDeCoisas
    - apenas para instancias de EntradaBlog

22

Usar Templates: *binding* em *runtime*

- O *template* só conhece o tipo de dados com que vai trabalhar quando é criado
  - em *runtime*
- *Binding* que diz respeito aos objetos
  - representado usando diagramas de objeto