Linguagem de Programação II
IMD0040

Aula 07 – Design de classes

Design de Classes

- **Regras Básicas de Design**:
  - O que é Design?
    - É uma das partes mais difíceis da programação.
    - Consiste em criar **abstrações**.
  - Isto significa três coisas:
    - Quais classes devem ser criadas?
    - Quais **responsabilidades** (métodos) devem ser assumidas por cada classe?
    - Quais são os **relacionamentos** entre tais classes e objetos dessas classes?

Design de Classes

- **Responsabilidades com os Dados**:
  - Qual é o princípio fundamental para atribuir responsabilidades?
    - Atribuir uma responsabilidade a **classe** que possui a informação necessária para executar uma tarefa.
  - Consequências:
    - **Fraco acoplamento** entre objetos e sistemas mais robustos e fáceis de manter.
    - **Alta coesão**, já que os objetos fazem tudo que é relacionado à sua própria informação.

Design de Classes

- **Fraco Acoplamento**:
  - Como **minimizar** dependências e **maximizar** o reuso?
  - O **acoplamento** é uma medida que estabelece o nível de conexão entre classes, ou seja, quando uma classe possui conhecimento ou depende de outra classe.
  - Com **fraco acoplamento**, uma classe não é dependente de muitas outras classes.

Design de Classes

- **Fraco Acoplamento**:
  - Com forte acoplamento, temos os seguintes problemas:
    - Mudanças em uma classe acarretam mudanças em outras classes.
    - A classe é mais difícil de entender isoladamente.
    - A classe é mais difícil de ser reusada, já que depende da presença de outras classes.

Exemplo de Acoplamento Forte

```
1  package forte_acoplamento;
2
3  public class Evento{
4
5      // atributos
6      private String nome;
7      private int dia, mes, ano;
8
9      public void agendar(String nome, int dia, int mes, int ano) {
10         this.nome = nome;
11         this.dia = dia;
12         this.mes = mes;
13         this.ano = ano;
14     }
15
16     public String getNome(){
17         return nome;
18     }
19
20     public int getDia(){
21         return dia;
22     }
23
24     public int getMes(){
25         return mes;
26     }
27
28     public int getAno(){
29         return ano;
30     }
31 }
```

Exemplo de Acoplamento Forte

```
1  package forte_acoplamento;
2
3  public class Evento{
4
5      // atributos
6      private String nome;
7      private int dia, mes, ano;
8
9      public void agendar(String nome, int dia, int mes, int ano) {
10         this.nome = nome;
11         this.dia = dia;
12         this.mes = mes;
13         this.ano = ano;
14     }
15
16     public String getNome(){
17         return nome;
18     }
19
20     public int getDia(){
21         return dia;
22     }
23
24     public int getMes(){
25         return mes;
26     }
27
28     public int getAno(){
29         return ano;
30     }
31 }
```

Exemplo de Acoplamento Forte

```java
package forte_acoplamento;
import java.util.Scanner;

public class Escola{
    Evento e = new Evento();
    public void cadastrarEvento(){

        Scanner ler = new Scanner(System.in);
        String nome = "";
        int dia, mes, ano;

        System.out.printf("Informe a descricao do evento:\n");
        nome = ler.next();

        System.out.printf("Informe o dia do evento:\n");
        dia = ler.nextInt();

        System.out.printf("Informe o mes do evento:\n");
        mes = ler.nextInt();

        System.out.printf("Informe o ano do evento:\n");
        ano = ler.nextInt();

        e.agendar(nome, dia, mes, ano);
    }

    public void mostrarEvento(){
        System.out.println("\n");
        System.out.println("Evento: " + e.getNome());
        System.out.println("Data: " + e.getDia() + "/" + e.getMes() + "/" + e.getAno());
    }
}
```

Exemplo de Acoplamento Forte

```java
package forte_acoplamento;

public class EscolaVisao{

    public static void main(String args []){
        Escola escola = new Escola();
        escola.cadastrarEvento();
        escola.mostrarEvento();
    }
}
```

```
Informe a descricao do evento:
Reunião
Informe o dia do evento:
15
Informe o mes do evento:
05
Informe o ano do evento:
2023


Evento: Reunião
Data: 15/5/2023
```

Exemplo de Acoplamento Forte

```
1  package forte_acoplamento;
2
3  public class Evento{
4
5      // atributos
6      private String nome;
7      private int dia, mes, ano;
8
9      public void agendar(String nome, int dia, int mes, int ano) {
10         this.nome = nome;
11         this.dia = dia;
12         this.mes = mes;
13         this.ano = ano;
14     }
15
16     public String getNome(){
17         return nome;
18     }
19
20     public int getDia(){
21         return dia;
22     }
23
24     public int getMes(){
25         return mes;
26     }
27
28     public int getAno(){
29         return ano;
30     }
31 }
```

Vamos mudar para tipo Date!

Exemplo de Acoplamento Forte

```
 1 package forte_acoplamento;
 2
 3 import java.util.Date;
 4
 5 public class Evento02 {
 6
 7     // atributos
 8     private String nome;
 9     private Date data;
10
11     public void agendar(String nome, Date dt) {
12         this.nome = nome;
13         this.data = dt;
14     }
15
16     public String getNome(){
17         return nome;
18     }
19
20     public Date getData() {
21         return data;
22     }
23 }
```

Exemplo de Acoplamento Forte

```
1  package forte_acoplamento;
2
3  import java.util.Date;
4
5  public class Evento02 {
6
7      // atributos
8      private String nome;
9      private Date data;
10
11     public void agendar(String nome, Date dt) {
12         this.nome = nome;
13         this.data = dt;
14     }
15
16     public String getNome(){
17         return nome;
18     }
19
20     public Date getData() {
21         return data;
22     }
23 }
```

Qual o problema???

Exemplo de Acoplamento Forte

```
1  package forte_acoplamento;
2  import java.util.Scanner;
3
4  public class Escola{
5      Evento02 e = new Evento02();
6      public void cadastrarEvento(){
7
8          Scanner ler = new Scanner(System.in);
9          String nome = "";
10         int dia, mes, ano;
11
12         System.out.printf("Informe a descricao do evento:\n");
13         nome = ler.next();
14
15         System.out.printf("Informe o dia do evento:\n");
16         dia = ler.nextInt();
17
18         System.out.printf("Informe o mes do evento:\n");
19         mes = ler.nextInt();
20
21         System.out.printf("Informe o ano do evento:\n");
22         ano = ler.nextInt();
23
24         e.agendar(nome, dia, mes, ano);
25     }
26
27     public void mostrarEvento(){
28         System.out.println("\n");
29         System.out.println("Evento: " + e.getNome());
30         System.out.println("Data: " + e.getDia() + "/" + e.getMes() + "/" + e.getAno());
31     }
32 }
```

Acoplamento Forte

Como vamos resolver?

Refatoração das Classes

```
1   import java.util.Date;
2
3   public class EventoMelhorado{
4
5       // atributos
6       private String descricao;
7       private Date data;
8
9       public String getDescricao(){
10          return descricao;
11      }
12
13      public Date getData(){
14          return data;
15      }
16
17      public void setDescricao(String str){
18          this.descricao = str;
19      }
20
21      public void setData(Date data){
22          this.data = data;
23      }
24  }
```

Refatoração das Classes

```
1
2  public class EscolaMelhorada{
3
4      private EventoMelhorado evento;
5
6      public void setEvento(EventoMelhorado evt){
7          this.evento = evt;
8      }
9
10     public EventoMelhorado getEvento(){
11         return evento;
12     }
13 }
```

Refatoração das Classes

```
1  import java.util.Scanner;
2  import java.util.Date;
3  import java.text.SimpleDateFormat;
4  import java.text.ParseException;
5
6  public class EscolaVisaoMelhorada{
7
8      public static void main(String args []) throws ParseException{
9
10         EscolaMelhorada escola = new EscolaMelhorada();
11         EventoMelhorado evento = new EventoMelhorado();
12
13         Scanner ler = new Scanner(System.in);
14
15         String descricao = "";
16         String data = "00-00-0000";
17
18         System.out.printf("Informe a descricao do evento:\n");
19         descricao = ler.next();
20
21         System.out.printf("Informe a data do evento:\n");
22         data = ler.next();
23
24         Date dt = new Date();
25         SimpleDateFormat formato = new SimpleDateFormat("dd-MM-yyyy");
26         dt = formato.parse(data);
```

Refatoração das Classes

```
28          evento.setDescricao(descricao);
29          evento.setData(dt);
30
31          escola.setEvento(evento);
32          mostrarEvento(escola);
33      }
34
35      public static void mostrarEvento(EscolaMelhorada escola){
36          System.out.println("\n");
37          System.out.println("Evento: " + escola.getEvento().getDescricao());
38          System.out.println("Data: " + escola.getEvento().getData());
39      }
40  }
```

Refatoração das Classes

```
Informe a descricao do evento:
Reunião
Informe a data do evento:
05-09-2023


Evento: Reunião
Data: Tue Sep 05 00:00:00 BRT 2023
```

Design de Classes

- Número de **tarefas** pelas quais uma única unidade é responsável.
- "Se cada unidade é responsável por uma única tarefa lógica, dizemos que ela tem **alta coesão**".
- Coesão se aplica a **classes** e **métodos**.
- Nosso objetivo: **alta coesão**.

Design de Classes

- **Alta Coesão**:
  - Problema:
    - Como gerenciar a complexidade?
    - A coesão mede o quão relacionadas ou focadas estão as responsabilidades da classe (coesão funcional).
    - Uma classe com **baixa coesão** faz muitas coisas não relacionadas e leva aos seguintes problemas:
      - Difícil de entender;
      - Difícil de reusar;
      - Difícil de manter;
      - "Delicada": constantemente sendo afetada por outras mudanças

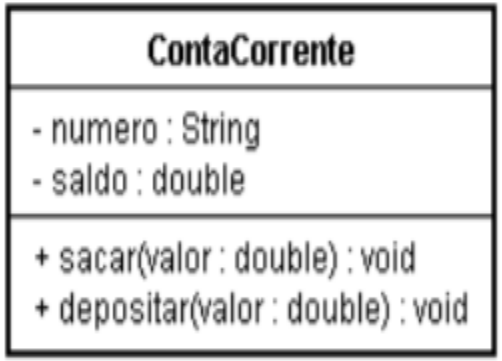
Design de Classes

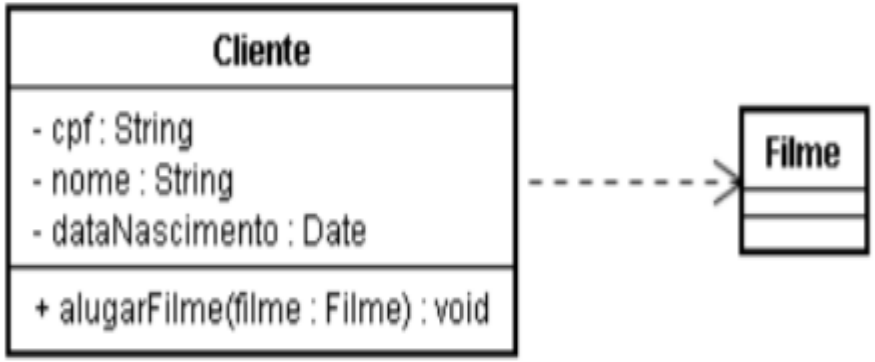
- Alta Coesão - possibilita:
  - Entender o que uma classe ou método faz.
  - Usar nomes descritivos.
  - Reusar classes ou métodos.

Design de Classes

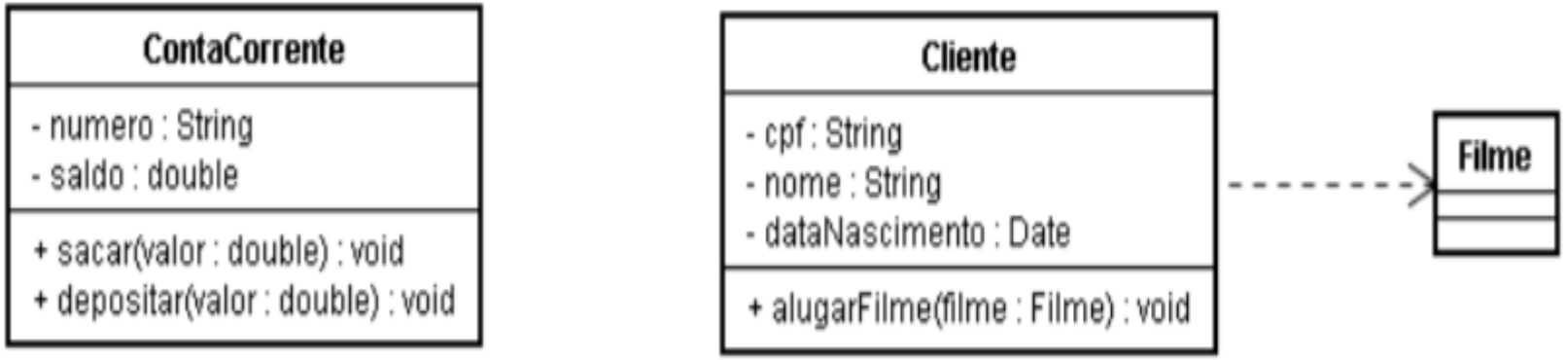
- Coesão de métodos:
  - Um método deve ser responsável por apenas uma tarefa bem definida.
- Coesão de classes:
  - As classes devem representar uma entidade única, bem definida.

Exemplos de Coesão



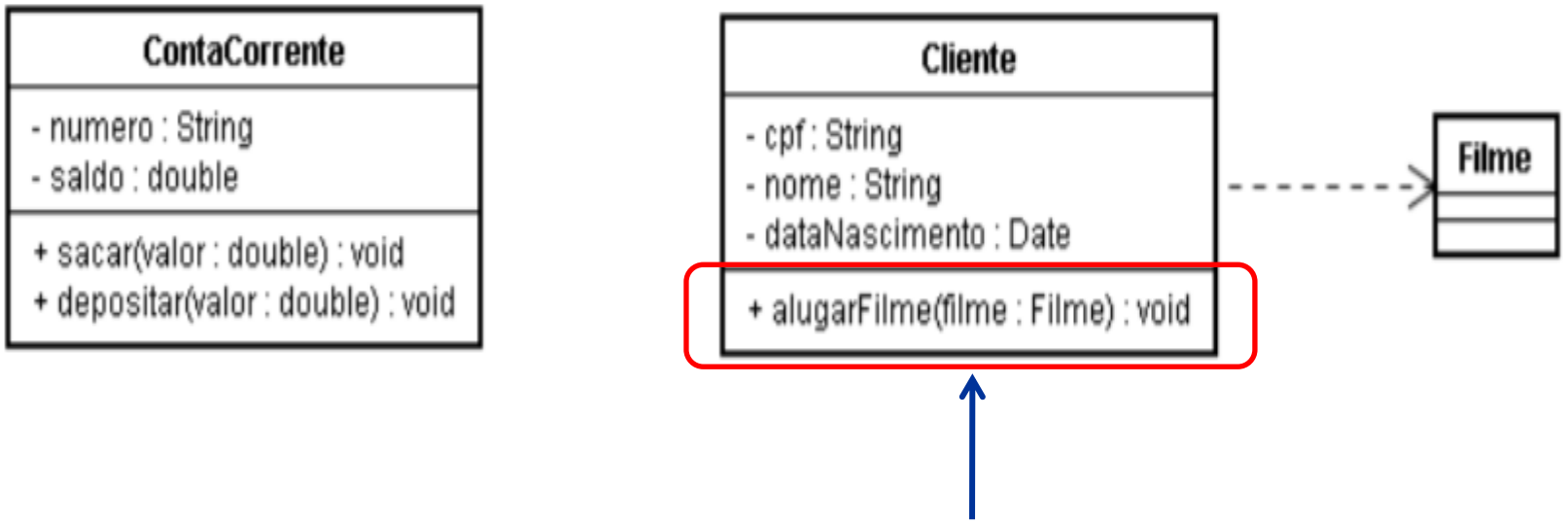


Exemplos de Coesão



- Qual a classe com menor **coesão**?
  - Por que?

Exemplos de Coesão



Qualidade de código

- Acoplamento fraco;
- Coesão forte.

Duplicação de código

- Indicador de Design ruim.
- Dificulta a manutenção.
- Pode levar à introdução de erros durante a manutenção.

Perguntas ...

Obrigado!!!