Desenvolvimento de Sistemas Orientados a Objeto

Programação Orientada a Objetos

Herança versus Interface

- ✓ A classe define ambos, um tipo e uma implementação desse tipo
- ✓ O tipo define apenas a interface oferecida para acessar objetos da classe
- ✓ Um objeto pode ter muitos tipos
- ✓ Classes diferentes podem implementar o mesmo tipo

Herança versus Interface

- ✓ Herança de classe ("extends") significa herança de implementação
- ✓ A sub-classe herda a implementação da superclasse
- ✓ É um mecanismo para compartilhar código e interface
- ✓ Ao fazer herança de classe, automaticamente faz também herança de interface
- ✓ Algumas linguagens não permitem definir tipos separadamente de classes

Herança versus Interface

- ✓ Herança de implementação permite reusar a funcionalidade de uma classe
- ✓ Herança de tipo oferece a habilidade de definir famílias de objetos com interfaces idênticas
- ✓ Isso é extremamente importante pois permite desacoplar um objeto de seus clientes através do polimorfismo

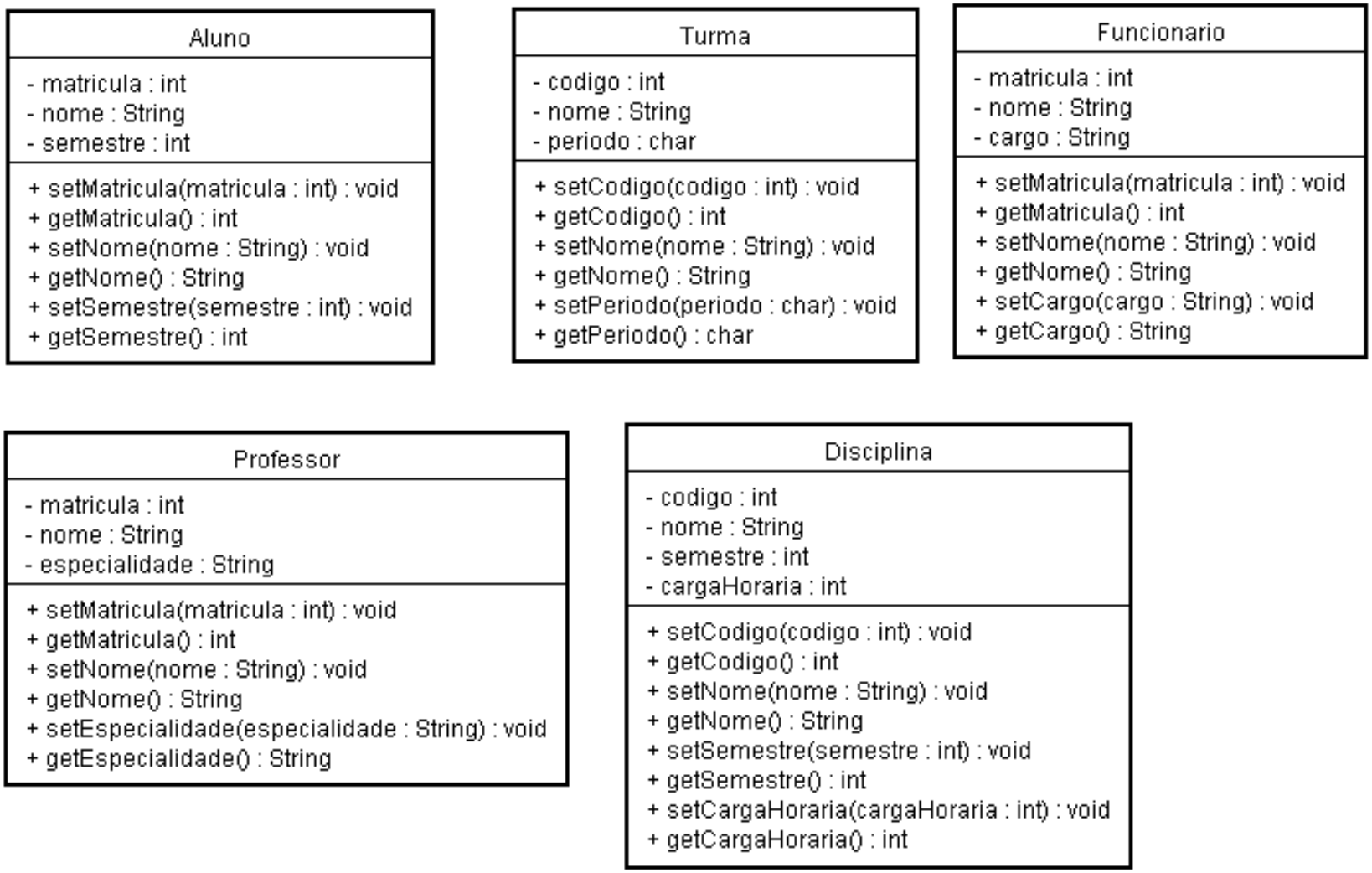
Herança versus Interface

- ✓ A herança de interface cria subtipos, permitindo o polimorfismo
- ✓ Programar em função de uma interface e não em função de uma implementação (uma classe particular) permite o polimorfismo e fornece a seguinte vantagem:
  - Clientes de um objeto, permanecem sem conhecimento das classes que o origina, bastando que as classes implementem a interface que define o tipo do objeto.

Herança versus Interface

- ✓ A flexibilidade vem da possibilidade de mudar a implementação da interface, até em tempo de execução, já que o polimorfismo é implementado em tempo de execução
- ✓ Em Java, uma classe pode implementar várias interfaces
- ✓ Isso permite ter mais polimorfismo, mesmo sem uma hierarquia

Herança versus Interface



Herança versus Interface

```
1  public class Aluno{
2
3      private int matricula;
4      private String nome;
5      private int semestre;
6
7      public void setMatricula(int matricula){
8          this.matricula = matricula;
9      }
10     public int getMatricula(){
11         return matricula;
12     }
13     public void setNome(String nome){
14         this.nome = nome;
15     }
16     public String getNome(){
17         return nome;
18     }
19     public void setSemestre(int semestre){
20         this.semestre = semestre;
21     }
22     public int getSemestre(){
23         return semestre;
24     }
25 }
```

Herança versus Interface

```
1  public class Disciplina{
2
3      private int codigo;
4      private String nome;
5      private int semestre;
6      private int cargaHoraria;
7
8      public void setCodigo(int codigo){
9          this.codigo = codigo;
10     }
11     public int getCodigo(){
12         return codigo;
13     }
14     public void setNome(String nome){
15         this.nome = nome;
16     }
17     public String getNome(){
18         return nome;
19     }
20     public void setSemestre(int semestre){
21         this.semestre = semestre;
22     }
23     public int getSemestre(){
24         return semestre;
25     }
26     public void setCargaHoraria(int cargaHoraria){
27         this.cargaHoraria = cargaHoraria;
28     }
29     public int getCargaHoraria(){
30         return cargaHoraria;
31     }
32 }
```

Herança versus Interface

```
1  public class Funcionario{
2
3      private int matricula;
4      private String nome;
5      private String cargo;
6
7      public void setMatricula(int matricula){
8          this.matricula = matricula;
9      }
10     public int getMatricula(){
11         return matricula;
12     }
13     public void setNome(String nome){
14         this.nome = nome;
15     }
16     public String getNome(){
17         return nome;
18     }
19     public void setCargo(String cargo){
20         this.cargo = cargo;
21     }
22     public String getCargo(){
23         return cargo;
24     }
25 }
```

Herança versus Interface

```
1  public class Professor{
2
3      private int matricula;
4      private String nome;
5      private String especialidade;
6
7      public void setMatricula(int matricula){
8          this.matricula = matricula;
9      }
10     public int getMatricula(){
11         return matricula;
12     }
13     public void setNome(String nome){
14         this.nome = nome;
15     }
16     public String getNome(){
17         return nome;
18     }
19     public void setEspecialidade(String especialidade){
20         this.especialidade = especialidade;
21     }
22     public String getEspecialidade(){
23         return especialidade;
24     }
25 }
```

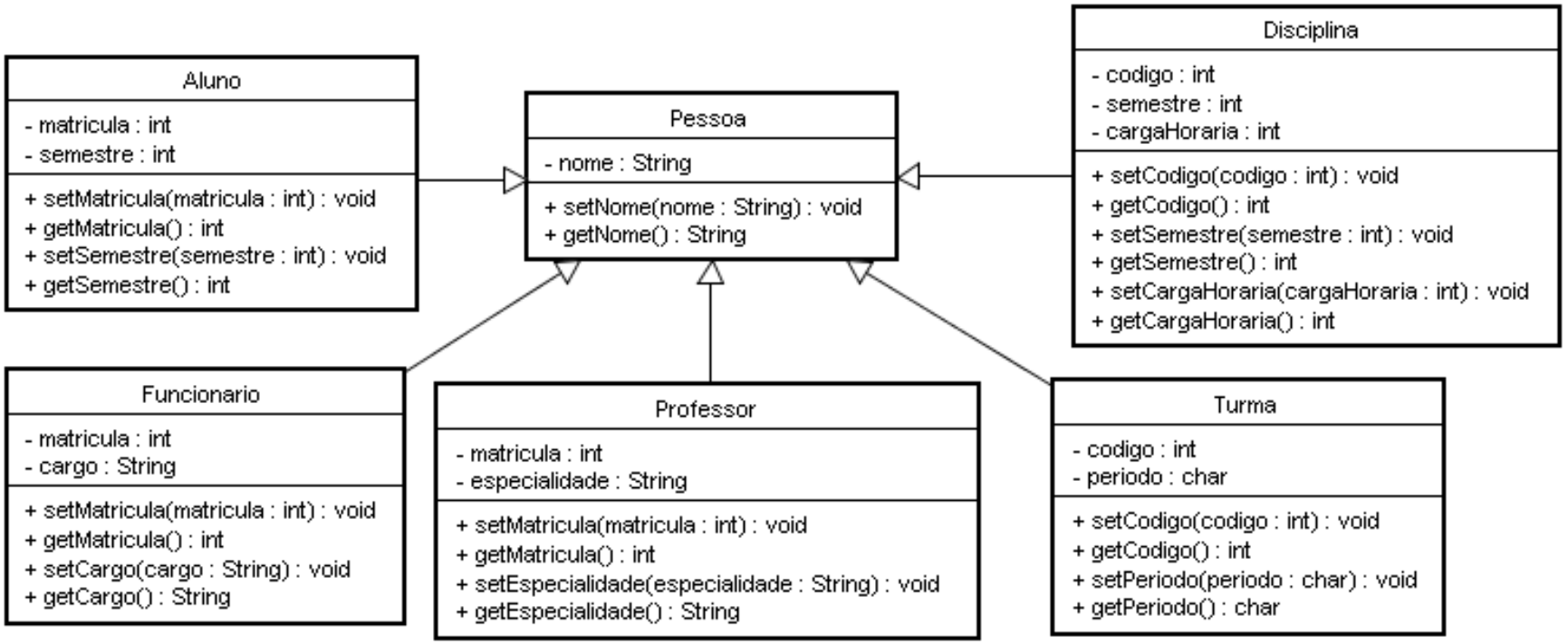
Herança versus Interface

```
1  public class Turma{
2
3      private int codigo;
4      private String nome;
5      private char periodo;
6
7      public void setCodigo(int codigo){
8          this.codigo = codigo;
9      }
10     public int getCodigo(){
11         return codigo;
12     }
13     public void setNome(String nome){
14         this.nome = nome;
15     }
16     public String getNome(){
17         return nome;
18     }
19     public void setPeriodo(char periodo){
20         this.periodo = periodo;
21     }
22     public char getPeriodo(){
23         return periodo;
24     }
25 }
```

Herança versus Interface

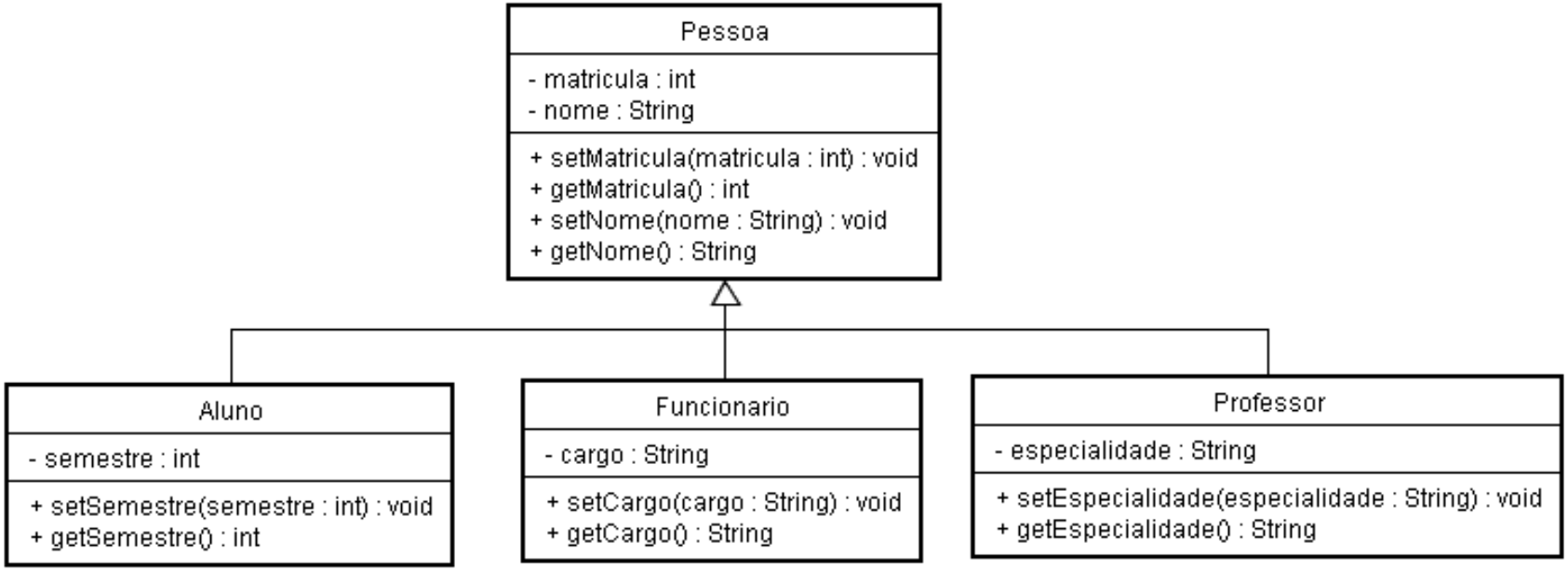
- ✓ Existem elementos comuns entre as classes anteriores?
- ✓ Qual(is) seria(m) a(s) superclasse(s)?

Proposta 01



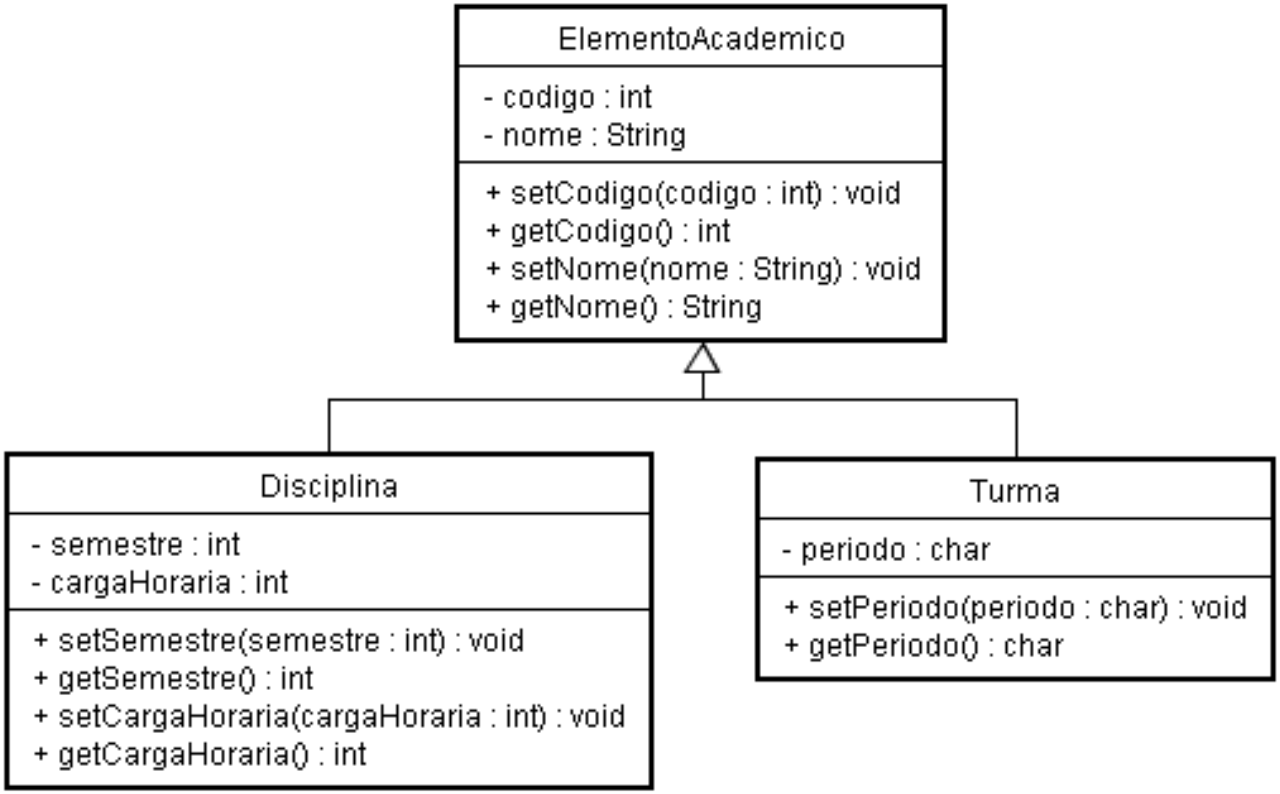
Uso incorreto de herança, Turma e Disciplina não tem ligação semântica com Pessoa, apenas reuso de código.

Proposta 02



Uso correto de herança, Aluno, Funcionário e Professor possuem uma ligação semântica com pessoa.

Proposta 02



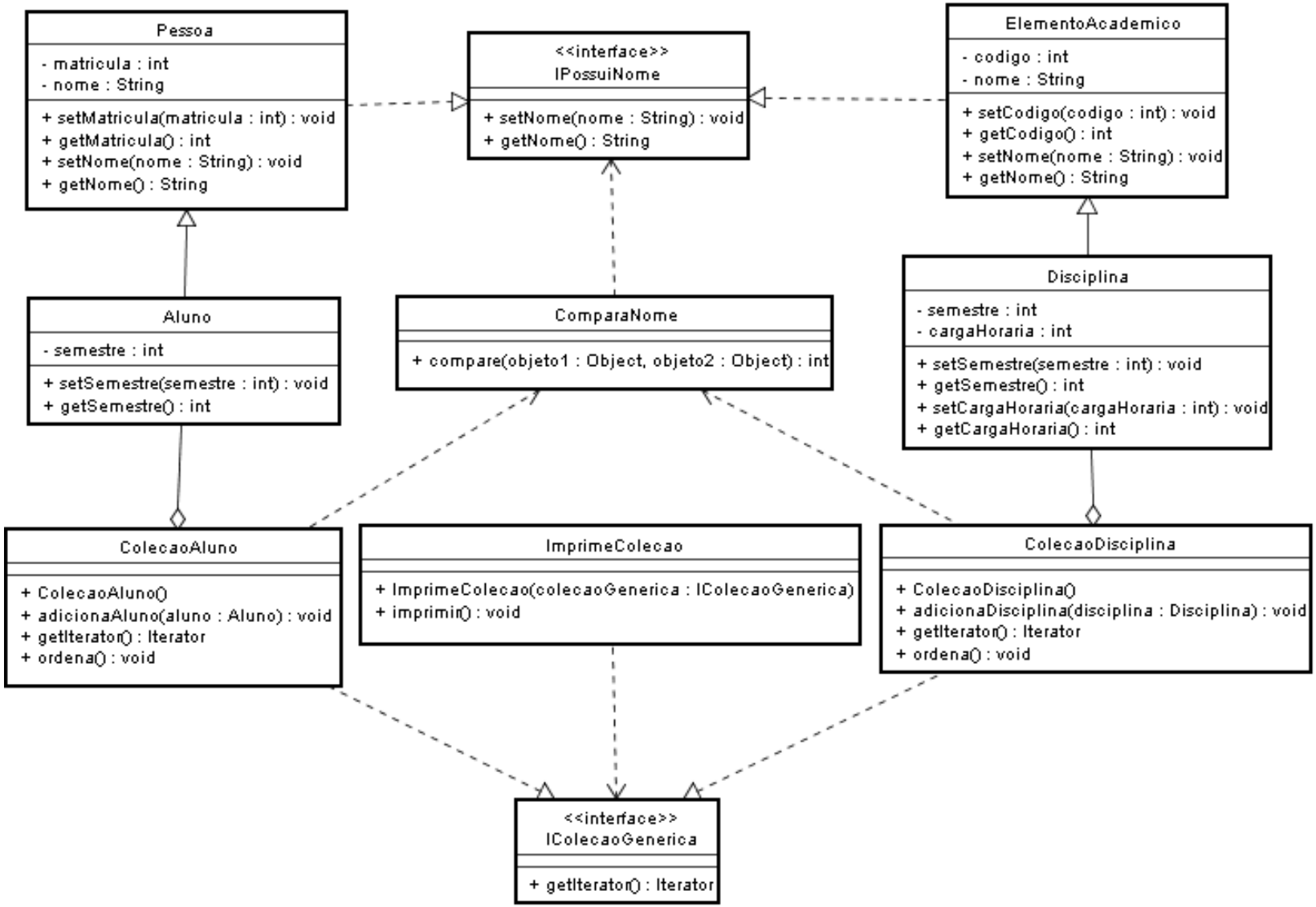
Uso correto de herança, Aluno, Funcionário e Professor possuem uma ligação semântica com pessoa

Herança versus Interface

- ✓ Como generalizar e tratar essas duas hierarquias de maneira comum?

- ✓ Situação:
  - Caso seja necessário desenvolver uma classe para imprimir o nome do Aluno, da Disciplina, do Professor, da Turma ou do Funcionário a partir de uma lista desses objetos, qual seria a saída para generalizar esse grupo de objetos?
  - Polimorfismo – com herança ou com interface?

Herança versus Interface



Herança versus Interface

```java
public interface IPossuiNome {

    public void setNome(String nome);

    public String getNome();
}
```

```java
public class Pessoa implements IPossuiNome{

    private int matricula;
    private String nome;

    public void setMatricula(int matricula){
        this.matricula = matricula;
    }
    public int getMatricula(){
        return matricula;
    }
    public void setNome(String nome){
        this.nome = nome;
    }
    public String getNome(){
        return nome;
    }
}
```

Herança versus Interface

```
1  public class ElementoAcademico implements IPossuiNome{
2
3      private int codigo;
4      private String nome;
5
6      public void setCodigo(int codigo){
7          this.codigo = codigo;
8      }
9
10     public int getCodigo(){
11         return codigo;
12     }
13
14     public void setNome(String nome){
15         this.nome = nome;
16     }
17
18     public String getNome(){
19         return nome;
20     }
21 }
```

Herança versus Interface

```java
public class Disciplina extends ElementoAcademico {

    private int semestre;
    private int cargaHoraria;

    public void setSemestre(int semestre){
        this.semestre = semestre;
    }
    public int getSemestre(){
        return semestre;
    }
    public void setCargaHoraria(int cargaHoraria){
        this.cargaHoraria = cargaHoraria;
    }
    public int getCargaHoraria(){
        return cargaHoraria;
    }
}
```

```java
public class Aluno extends Pessoa {

    private int semestre;

    public void setSemestre(int semestre){
        this.semestre = semestre;
    }

    public int getSemestre(){
        return semestre;
    }
}
```

Herança versus Interface

```
1  import java.util.Iterator;
2  public interface IColecaoGenerica {
3
4      public Iterator getIterator();
5
6  }
```

```
1  import java.util.Comparator;
2  public class ComparaNome implements Comparator {
3      public int compare(Object objeto1, Object objeto2) {
4          String nome1 = ((IPossuiNome)objeto1).getNome();
5          String nome2 = ((IPossuiNome)objeto2).getNome();
6          return nome1.compareTo(nome2);
7      }
8  }
```

Herança versus Interface

```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3  import java.util.Collections;
4
5  public class ColecaoAluno implements IColecaoGenerica {
6
7      ArrayList listaAluno;
8
9      public ColecaoAluno(){
10         listaAluno = new ArrayList();
11     }
12
13     public void adicionaAluno(Aluno aluno){
14         listaAluno.add(aluno);
15     }
16
17     public Iterator getIterator(){
18         return listaAluno.iterator();
19     }
20
21     public void ordena(){
22         Collections.sort(listaAluno, new ComparaNome());
23     }
24 }
```

Herança versus Interface

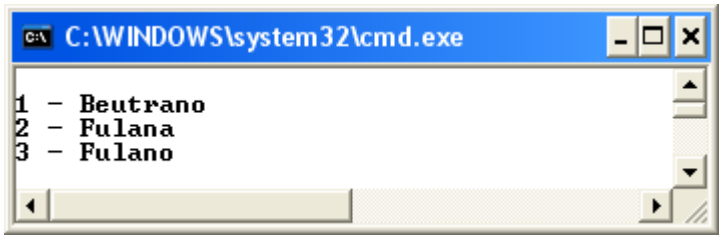
```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3  import java.util.Collections;
4
5  public class ColecaoDisciplina implements IColecaoGenerica {
6
7      ArrayList listaDisciplina;
8
9      public ColecaoDisciplina(){
10         listaDisciplina = new ArrayList();
11     }
12
13     public void adicionaDisciplina(Disciplina disciplina){
14         listaDisciplina.add(disciplina);
15     }
16
17     public Iterator getIterator(){
18         return listaDisciplina.iterator();
19     }
20
21     public void ordena(){
22         Collections.sort(listaDisciplina, new CompNome());
23     }
24 }
```

Herança versus Interface

```
1  import java.util.Iterator;
2  public class ImprimeColecao {
3
4      Iterator colecao;
5
6      public ImprimeColecao(IColecaoGenerica colecaoGenerica){
7          this.colecao = colecaoGenerica.getIterator();
8      }
9
10     public void imprimir(){
11         int x=0;
12         System.out.print("\n\n");
13         while (colecao.hasNext()){
14             IPossuiNome objeto = (IPossuiNome) colecao.next();
15             System.out.println((++x)+" - "+ objeto.getNome());
16         }
17         System.out.print("\n\n");
18     }
19 }
```

Herança versus Interface

```
1  public class TesteColecoes {
2
3      public static void main(String args[]){
4
5          Aluno aluno;
6          ColecaoAluno colecaoAluno = new ColecaoAluno();
7
8          ImprimeColecao relatorio;
9
10         aluno = new Aluno();
11         aluno.setMatricula(10);
12         aluno.setNome("Fulano");
13         aluno.setSemestre(1);
14         colecaoAluno.adicionaAluno(aluno);
15
16         aluno = new Aluno();
17         aluno.setMatricula(20);
18         aluno.setNome("Fulana");
19         aluno.setSemestre(2);
20         colecaoAluno.adicionaAluno(aluno);
21
22         aluno = new Aluno();
23         aluno.setMatricula(30);
24         aluno.setNome("Beutrano");
25         aluno.setSemestre(2);
26         colecaoAluno.adicionaAluno(aluno);
27
28         colecaoAluno.ordena();
29         relatorio = new ImprimeColecao(colecaoAluno);
30         relatorio.imprimir();
31     }
32 }
```



Herança versus Interface

```
1  public class TesteColecoes2 {
2
3      public static void main(String args[]){
4
5          Disciplina disciplina;
6          ColecaoDisciplina colecaoDisciplina = new ColecaoDisciplina();
7
8          ImprimeColecao relatorio;
9
10         disciplina = new Disciplina();
11         disciplina.setCodigo(1010);
12         disciplina.setNome("Programacao orientada a objetos");
13         disciplina.setSemestre(2);
14         disciplina.setCargaHoraria(60);
15         colecaoDisciplina.adicionaDisciplina(disciplina);
16
17         disciplina = new Disciplina();
18         disciplina.setCodigo(1020);
19         disciplina.setNome("Analise orientada a objetos");
20         disciplina.setSemestre(5);
21         disciplina.setCargaHoraria(60);
22         colecaoDisciplina.adicionaDisciplina(disciplina);
23
24         disciplina = new Disciplina();
25         disciplina.setCodigo(1030);
26         disciplina.setNome("Projeto orientado a objetos");
27         disciplina.setSemestre(6);
28         disciplina.setCargaHoraria(40);
29         colecaoDisciplina.adicionaDisciplina(disciplina);
30
31         colecaoDisciplina.ordena();
32         relatorio = new ImprimeColecao(colecaoDisciplina);
33         relatorio.imprimir();
34     }
35 }
```