Sistemas de Informação

INSTITUTO FEDERAL GOIANO

DISCIPLINA: PROGRAMAÇÃO ORIENTADA A OBJETOS

Aula 01

PROF. ANDRÉ

INSTITUTO FEDERAL
GOIANO

Programação Orientada a Objeto

POO é uma forma de analisar e projetar um projeto de software e se baseia em uma infinidade de objetos e suas interações. Uma de suas vantagens é o fato de conseguir modelar o mundo real do domínio do problema em um conjunto de componentes de software que seja o mais fiel possível na representação deste domínio.

Programação Orientada a Objeto...

Na POO, aplica-se um conjunto de classes que definem os objetos presentes no sistema de software. Estas classes determinam o comportamento e os estados possíveis de seus objetos, e também o relacionamento com outros objetos.

As linguagens C++, C# e Java são exemplos clássicos de linguagens de programação orientadas a objetos. Outras linguagens a partir de versões mais recentes também aderiram aos conceitos de orientação a objetos.

Programação Orientada a Objeto...

A orientação a objetos possui um conjunto de representações que possibilitam às linguagens representar de forma fiel algum domínio do mundo real assim como controlar as interações e as ações que ocorrem em cada objeto.

As principais representações básicas da POO são: **classes**, **objetos/instâncias**, **atributos** e **métodos**. Algumas das principais características deste conceito são: **abstração**, **encapsulamento**, **herança** e **polimorfismo**.

Programação Orientada a Objeto...

Classe → É um conjunto de objetos com características em comum. A classe define o comportamento dos objetos através de métodos, e quais estados ele é capaz de manter através de seus atributos. Um exemplo de classe pode ser uma **Pessoa**. Uma pessoa possui várias características (atributos) como altura, peso, cor dos olhos, etc.

Programação Orientada a Objeto...

Atributo → Eles são as características de um objeto. Representa a estrutura de dados que vai representar a classe. Existe também o conjunto de valores dos atributos de um determinado objeto que é chamado de estado do objeto. Como exemplo de atributos de João poderíamos citar: **altura, peso** e **cor dos olhos**. Os valores atribuídos a estas três características como 1.78m, 75kg, verde, representam o estado do objeto.

Programação Orientada a Objeto...

Método → O método define as ações em uma classe. Esta ação só ocorre quando o método é invocado através do objeto. Geralmente, uma classe possui diversos métodos, que no caso da classe **Pessoa** poderiam ser: **andar**, **correr**, **comer**, **etc.**

INSTITUTO FEDERAL
GOIANO

Programação Orientada a Objeto...

Objeto/Instância → Armazena estados através de seus atributos e reage a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Os objetos representam as características de uma classe, assim poderíamos definir como **objeto** da classe **Pessoa**, João, Tereza, etc.

Programação Orientada a Objeto...

```
import java.util.Date;

public class ImportaData {

    public static void main(String[] args){
        Date objDate = new Date();
        System.out.println("A data de hoje em mili segundos é: "+ objDate.getTime());
    }

}
```

Declarando/instanciando um objeto.

Utilizando um método da classe Date através do objeto criado.

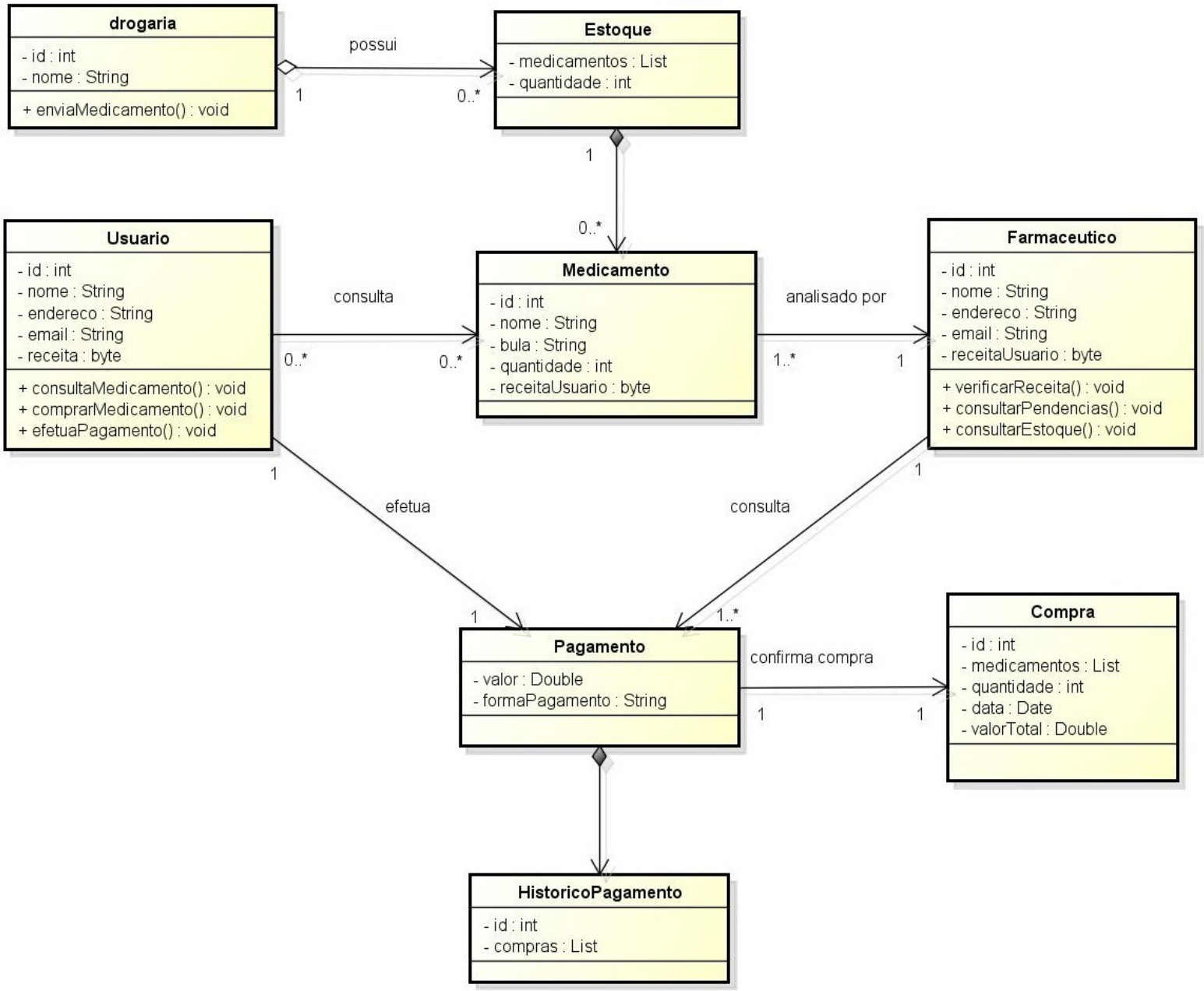


Diagrama de Classes

Programação Orientada a Objeto...

Pilar 1 - Abstração → Utilizada para a definição de entidades do mundo real. Através dela que são criadas as classes. Compreende também as formas com que as representações se comunicam e o fluxo de informação que ocorre em um certo domínio.

INSTITUTO FEDERAL
GOIANO

Programação Orientada a Objeto...

Pilar 2 - Encapsulamento → Técnica usada para esconder uma ideia, ou seja, não expôr detalhes internos para o usuário, tornando partes do sistema mais independentes possível. Ela garante que a parte de acesso dos atributos, principalmente, seja restrita somente à classe que os manipula.

```
public class Conta {
        private double Saldo = 0;
        private String Nome;

        public void deposita(double Valor){
                this.Saldo = this.Saldo + Valor + (Valor * 0.10);
        }
        public double getSaldo(){
                return this.Saldo;
        }

        public void setNome(String N){
                this.Nome = N;
        }

        public String getNome(){
                return this.Nome;
        }
}
```

```
class UsaConta {

        public static void main(String[] args){
            //instanciando duas contas, c1 e c2
            Conta c1=new Conta();
            Conta c2=new Conta();

            c1.setNome("Fulano da silva");
            c2.setNome("Beltrano de oliveira");

        //depositando
            c1.deposita(10);
            c2.deposita(50);

            System.out.println("c1 - Nome: "+c1.getNome());
            System.out.println("c1 - Saldo: "+c1.getSaldo());
            System.out.println("c2 - Nome: "+c2.getNome());
            System.out.println("c2 - Saldo: "+c2.getSaldo());
    }
}
```

Desafio

Faça um programa em java que calcule o IMC de uma pessoa e imprima o IMC e a faixa de obesidade da mesma, utilizando os conceitos de encapsulamento. O programa deverá conter:

• Uma classe para os atributos da pessoa (cpf, nome, peso e altura);

• Uma classe com 2 métodos: um para calcular o IMC e outro para verificar a faixa de obesidade;

• Uma classe para rodar o programa e exibir o IMC e a faixa de obesidade;

• Os valores deverão ser informados pelo usuário através da classe Scanner e o programa deve apresentar um menu para calcular novamente ou sair;

- IMC = peso / (altura)2

Valores e Classificação:

- 18.50 – 24.99: Peso Normal
- 25.00 – 29.99: Pré-Obesidade
- 30.00 – 34.99: Obesidade Grau I
- 35.00 – 39.99: Obesidade Grau II
- ≥40.00: Obesidade Grau III

Fonte: Organização Mundial da Saúde, 2004.

Programação Orientada a Objeto...

Pilar 3 - Herança → Na orientação a objetos é permitido que uma classe herde atributos e métodos da outra, tendo apenas uma restrição para a herança. Os modificadores de acessos das classes, métodos e atributos só podem estar com visibilidade *public* e *protected* para que sejam herdados. Utiliza-se a marcação *extends* para que uma classe herde características de outra.

A maior vantagem de usar o recurso da herança é na reutilização do código. Esse reaproveitamento pode ser acionado quando se identifica que o atributo ou método de uma classe será igual para as outras.

```java
package aula01;

public class Pessoa {

    private String nome, idade, endereco;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getIdade() {
        return idade;
    }

    public void setIdade(String idade) {
        this.idade = idade;
    }

    public String getEndereco() {
        return endereco;
    }

    public void setEndereco(String endereco) {
        this.endereco = endereco;
    }

}
```

INSTITUTO FEDERAL
GOIANO

```java
package aula01;

/**
 *
 * @author andre
 */
public class Cliente extends Pessoa {

    private String cpf;

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public void imprimirCliente(){

            System.out.println("O nome do Cliente e: "+ this.getNome());
            System.out.println("O cpf do Cliente e: "+ this.getCpf());

    }

}
```

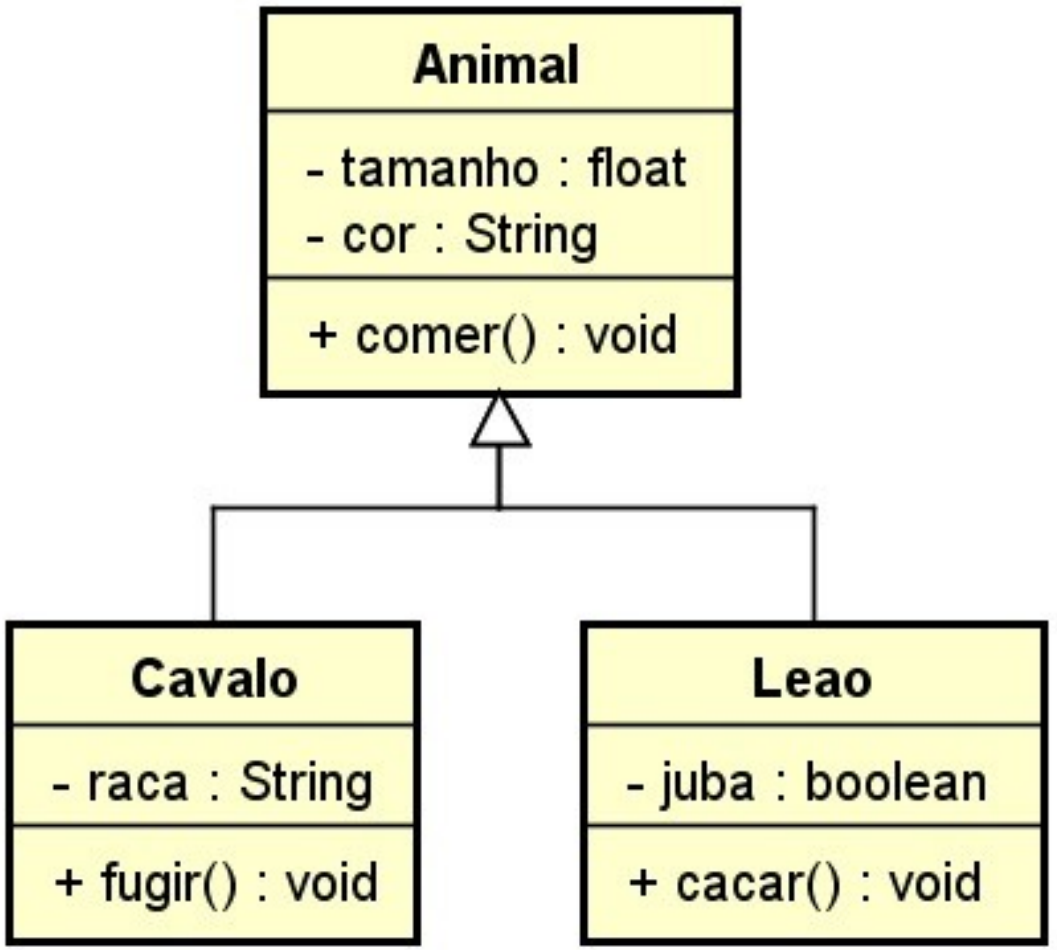
```java
package aula01;

/**
 *
 * @author andre
 */
public class Fornecedor extends Pessoa {

    private String cnpj;

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    public void imprimirFornecedor(){

            System.out.println("O nome do fornecedor e: "+ this.getNome());
            System.out.println("O cnpj do fornecedor e: "+ this.getCnpj());

    }


}
```

INSTITUTO FEDERAL
GOIANO

```java
package aula01;

/**
 *
 * @author andre
 */
public class PessoaPrincipal {

    public static void main(String[] args) {

        Cliente cliente = new Cliente();
        cliente.setNome("Primeiro Cliente");
        cliente.setIdade("27");
        cliente.setEndereco("Rua 1 do Cliente");
        cliente.setCpf("876.654.543-23");

        cliente.imprimirCliente();

        Fornecedor fornecedor = new Fornecedor();
        fornecedor.setNome("Primeiro Fornecedor");
        fornecedor.setIdade("45");
        fornecedor.setEndereco("Rua 2 do Fornecedor");
        fornecedor.setCnpj("87.543.654/0001-34");

        fornecedor.imprimirFornecedor();


    }


}
```

INSTITUTO FEDERAL
GOIANO

Vamos exercitar?



INSTITUTO FEDERAL
GOIANO

Baseado no diagrama de classes acima, construa as 3 classes que representam a abstração e uma classe principal que invoca os métodos de cada uma das classes filhas, informando a comida que os 2 animais comeram, o local para onde o cavalo fugiu e o animal que o leão caçou. Utilize a classe Scanner para informar os parâmetros dos métodos.

Programação Orientada a Objeto...

Pilar 4 - Polimorfismo → É o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação, assinatura, mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse.

De forma genérica, polimorfismo significa várias formas. No caso da Orientação a Objetos, polimorfismo denota uma situação na qual um objeto pode se comportar de maneiras diferentes ao receber uma mensagem, dependendo do seu tipo de criação.

Polimorfismo

```java
package aula01;

/**
 *
 * @author andre
 */
public class OperacaoMatematica {

    public double calcular(double x, double y){

        return 0;
    }

}
```

INSTITUTO FEDERAL
GOIANO

Polimorfismo

```java
package aula01;

/**
 *
 * @author andre
 */
public class Soma extends OperacaoMatematica {

    @Override
    public double calcular(double x, double y) {
        return x + y;
    }

}
```

INSTITUTO FEDERAL
GOIANO

Polimorfismo

```java
package aula01;

/**
 *
 * @author andre
 */
public class Subtrai extends OperacaoMatematica {

    @Override
    public double calcular(double x, double y) {
        return x - y;
    }

}
```

INSTITUTO FEDERAL
GOIANO

Polimorfismo

```java
package aula01;

/**
 *
 * @author andre
 */
public class Multiplica extends OperacaoMatematica {

    @Override
    public double calcular(double x, double y) {
        return x * y;
    }


}
```

INSTITUTO FEDERAL
GOIANO

Polimorfismo

```java
package aula01;

/**
 *
 * @author andre
 */
public class Divide extends OperacaoMatematica {

    @Override
    public double calcular(double x, double y) {
        return x / y;
    }

}
```

INSTITUTO FEDERAL
GOIANO

Polimorfismo

```java
package aula01;

/**
 *
 * @author andre
 */
public class ProgramaOperacaoMatematica {

    //UTILIZANDO O POLIMORFISMO
    //Declarando como static para poder ser usado na propria classe
    public static void calculaOperacao(OperacaoMatematica operacao, double x, double y) {
        System.out.println(operacao.calcular(x, y));
    }

    public static void main(String[] args) {
        calculaOperacao(new Soma(), 10, 10);
        calculaOperacao(new Subtrai(), 10, 10);
        calculaOperacao(new Multiplica(), 10, 10);
        calculaOperacao(new Divide(), 10, 10);

    }

}
```

Programação Orientada a Objeto...

Classe *abstract* → Pode-se dizer que as classes abstratas servem como “modelo” para outras classes que dela herdem, não podendo ser instanciada por si só. Para ter um objeto de uma classe abstrata é necessário criar uma classe mais especializada herdando dela e então instanciar essa nova classe. Os métodos abstratos da classe abstrata devem obrigatoriamente serem sobrescritos nas classes filhas.

```
public abstract class Animal {
	public abstract double obterCotaDiariaDeLeite();
}

public class Gato extends Animal {
	public double obterCotaDiariaDeLeite(){
		return 20.0;
	}
}

public class Rato extends Animal {
	public double obterCotaDiariaDeLeite() {
		return 0.5;
	}
}

public class ProgramaAnimal {
	public static void main(String args[]){
		System.out.println("Polimorfismo\n");
		Animal animal1 = new Gato();
		System.out.println("Cota diaria de leite do gato: " +
animal1.obterCotaDiariaDeLeite());
		Animal animal2 = new Rato();
		System.out.println("Cota diaria de leite do rato: " +
animal2.obterCotaDiariaDeLeite());
	}
}
```

INSTITUTO FEDERAL
GOIANO

Programação Orientada a Objeto...

Interface → É um conjunto de métodos abstratos que define um contrato que as classes que a implementam devem obrigatoriamente seguir. Elas são semelhantes a classes, mas só podem ter assinaturas de métodos, atributos e métodos padrão. Permite implementação múltipla, quando uma classe pode implementar várias interfaces.

INSTITUTO FEDERAL
GOIANO

Interface

```
public interface Veiculo {
    public String placa = "";
    public float velMax;
    public void iniciar();
    public void parar();
    default void buzinar(){
      System.out.println("Buzinando");
   }
}
```

INSTITUTO FEDERAL
GOIANO

Interface

```
public class Carro implements Veiculo {
    public void iniciar() {
        System.out.println("ligando o motor...");
    }
    public void parar() {
        System.out.println("parando o motor...");
    }
}

Veiculo veiculo = new Carro();
veiculo.iniciar();
```

INSTITUTO FEDERAL
GOIANO

Interface

```
public interface GPS {
    public void obterCoordenadas();
}

public interface Radio {
    public void ligarRadio();
    public void pararRadio();
}

public class Smartphone implements GPS,Radio {
    public void obterCoordenadas() {
        // retorna coordenadas
    }
    public void ligarRadio() {
      // liga o rádio
    }
    public void pararRadio() {
        // desliga o rádio
    }
}
```

INSTITUTO FEDERAL
GOIANO

Vamos exercitar?

Utilizando a linguagem Java e os conceitos que foram trabalhados na aula, desenvolva um programa que leia o nome e a nota de uma quantidade X de alunos (informada pelo usuário), da disciplina POO, do curso de Sistemas de informação, e ao final imprima:

- o nome e a nota de cada aluno;
- a maior nota;
- a menor nota;
- a média normal das notas;