Aula 04

Programação orientada a objetos

Pauta

- Coletor de lixo
- A referência this
- Variáveis de instância final
- Encapsulamento
- Enumerações
- Pacotes

Pauta

- Herança
- A referência super
- Sobrescrita de métodos
- Modificadores de acesso
- Classe Object;

Revisão

- Métodos;
- Construtores;
- Classe;
- Objeto;
- Instância;
- Static;

Coletor de lixo

- Em java utilizamos os construtores para criar um novo objeto. E para finalizar o objeto, o que utilizamos?

Coletor de lixo

- Não é necessário preocupar com a finalização do objeto;
- Coletor de lixo;
- Método finalize();
- Não é possível prever o momento exato da finalização de determinado objeto;

Coletor de lixo

- É responsavel pela destruição dos objetos;
- É automático;
- System.gc
- chamando o método estático gc da classe System, você esta sugerindo para que a Virtual Machine rode o Garbage Collector naquele momento. Se sua sugestão vai ser aceita ou não,isto depende e você não tem garantias. Evite o uso deste método.

A referência this

- Utilizada para acessar membros não static da mesma classe;
- Não pode ser utilizada em métodos static;
- O método this invoca o próprio construtor da classe;
- Exemplos;

Variáveis de instância final

- Indica que o valor de uma determinada variável não poderá ser alterado;
- Principio do menor privilégio;
- Ex.: final int INCREMENTO;
- Esta variável deverá sempre ser inicializada;
- Uma vez inicializada seu valor jamais poderá ser alterado;

Encapsulamento

- Uma boa prática da programação é controlar o acesso a membros da classe com métodos get e set.
- O modificador de acesso private impede que os membros sejam acessados diretamente;
- Ex.: um membro do tipo int chamado idade:
- private int idade;
- int getIdade(){return idade;}
- void setIdade(int idade){}

Enumerações

- Palavra chave enum
- Tipos enum são implicitamente final, porque declaram constantes que não devem ser modificadas;
- Pode ser entendido como uma classe que possui um array de tipos final;

Pacotes

- Pacotes facilitam a reutilização de software;
- É uma forma de manter organizado a sua aplicação;
- Utilizamos a palavra chave package (deve vir antes de qualquer declaração);
- Há somente uma declaração package em código fonte java;
- Ex.: package br.com.ambientinformatica.treinamentos.aula

Pacotes

- Os pacotes são vistos no sistema operacional como diretórios. Ex.:
- Para executar uma classe dentro de um pacote é necessário fazer:
- java br.com.treinamento.aula04.Aclasse

Duvidas??

AMBIENT
INFORMÁTICA

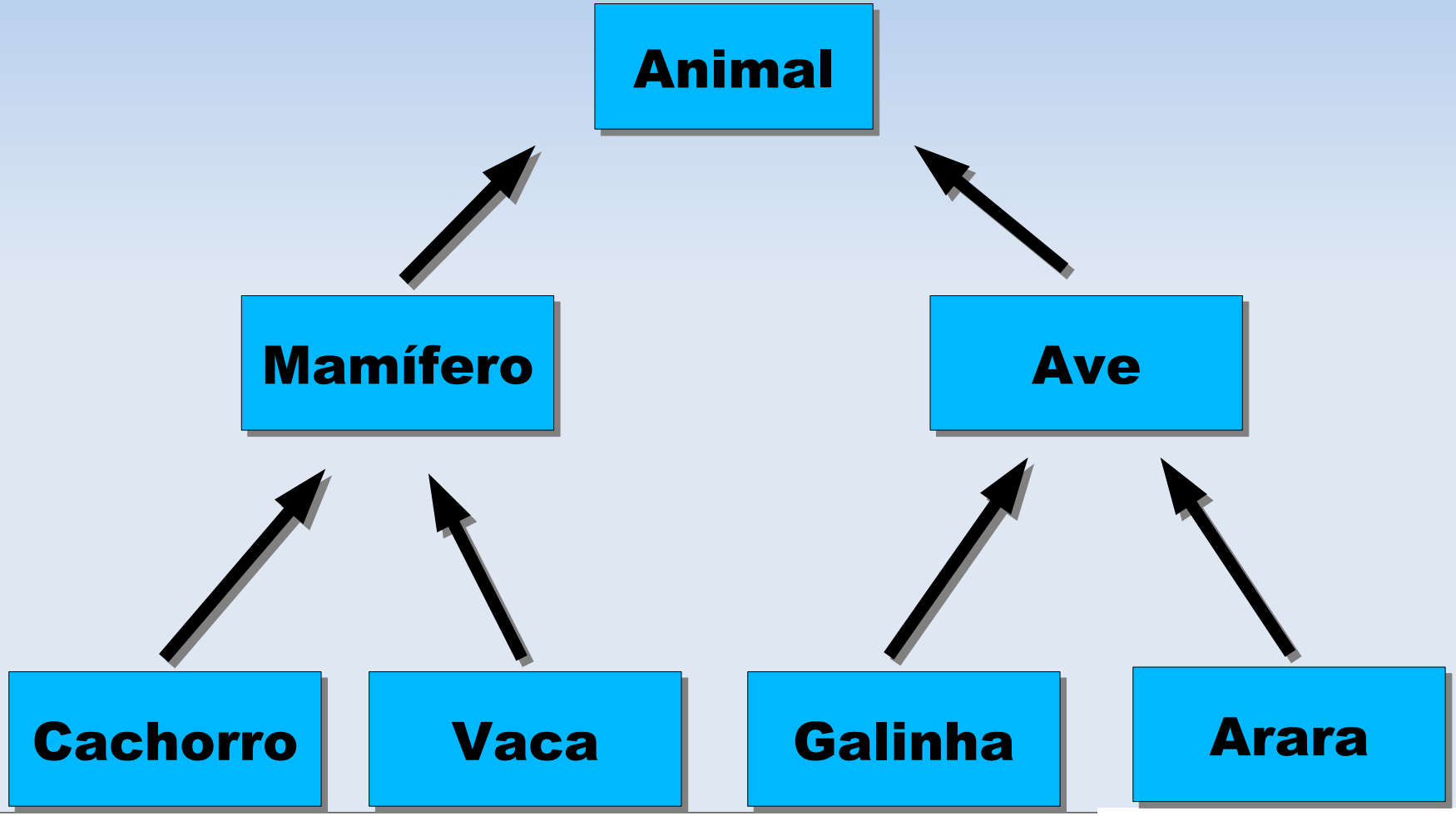
Herança

- Um recurso da orientação a objetos (POO);
- É uma forma de reutilização de software em que uma classe é criada herdando todas as características de uma outra classe;
- Há economia de tempo uma vez que pode ser reutilizado software já testado;
- Não há a necessidade de reprogramar todo o código comum a uma ou mais classes;

Herança

- Java não suporta herança múltipla;
- Superclasse direta é a classe a partir da qual a subclasse herda explicitamente;
- Superclasse indireta é qualquer superclasse acima da superclasse direta;
- Define o relacionamento é um;
- Superclasse, classe pai, ancestral
- Subclasse, classe filha

Herança



Herança

- Para trabalhar com herança no Java utlizamos a palavra chave:

```
extends
```

```
Assim:
public class SubClasse extends
  SuperClasse{
}
```

A referência super

- Fazemos referências a superclasse utilizando a palavra chave super;
- Podemos invocar o construtor da superclasse utilizando o método super(); Este método deve vir antes de qualquer código no construtor;
- Quando instanciamos uma subclasse é chamado o construtor da superclasse primeiro (utilizando super() - mesmo não tendo sido informado);

Sobrescrita de métodos

- Quando implementamos um método na subclasse com o mesmo nome e quantidade de argumentos de um método da superclasse dizemos que sobrescrevemos o método.

Modificadores de acesso

- Existem quatro modificadores de acesso no Java:
- private;
- default-pacote;
- protected;
- public;
- Obs.: para utilizar o modificador de acesso a nível de pacote(default) basta não informar o modificador;

Modificadores de acesso: private

- Proíbe o acesso fora da classe onde o membro private foi declarado.
- Ao tentarmos acessar um membro private de outra classe o compilador gera mensagens de erro declarando que esse membro private não é acessivel.

Modificadores de acesso: pacote

- Permite o acesso a membros do mesmo pacote;
- Não possui palavra chave, utilizamos quando não informamos explicitamente o modificador de acesso;
- Ao tentarmos acessar um membro default de uma classe em outro pacote o compilador gera mensagens de erro declarando que esse membro default não é acessível.

Modificadores de acesso: protected

- Permite o acesso a membros do mesmo pacote (como acesso default);
- Os membros declarados protected em uma superclasse são acessiveis por todas as subclasses (mesmo estas estando em pacotes diferentes)

Modificadores de acesso: public

- Permite o acesso público;
- Qualquer membro de qualquer classe estando em qualquer pacote podem acessar membros public.

A classe Object

- Todas as classes java herdam da classe object (mesmo não sendo informado.);
- Alguns métodos especiais de Object:
- finalize()
- toString()
- equals(Object obj)
- Outros métodos de object são: clone, getClass, hashCode, notify, notifyAll, wait;

Revisão

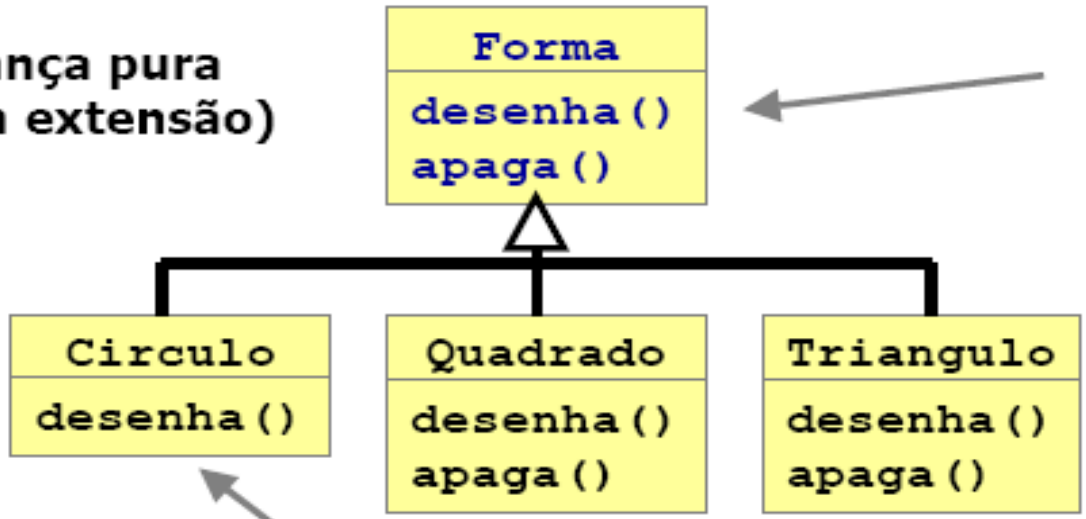
- Herança
- Sobrescrita de métodos
- Modificadores de acesso
- Classe Object;

Dúvidas?

AMBIENT
INFORMÁTICA

...entendendo melhor a Herança

Herança pura
(sem extensão)



interface original é automaticamente duplicada nas classes derivadas

Campos de dados da classe base também são duplicados

Se membro derivado não for redefinido, implementação original é usada

```
class Forma {
  public void desenha() {
    /*...*/
  }
  public void apaga() {
    /*...*/
  }
}
```

```
class Circulo extends Forma {
  public void desenha() {
    /*...*/
  }
}
```

*Assinatura do método **tem que ser igual** ou sobreposição não ocorrerá (poderá ocorrer sobrecarga)*

...entendendo melhor Modificadores de acesso

public

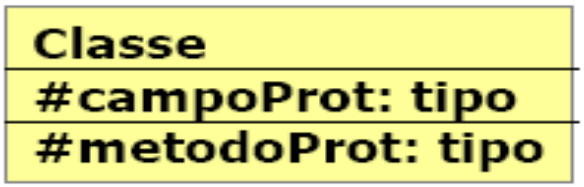
- *Acessível*
  - *na própria classe*
  - *nas subclasses*
  - *nas classes do mesmo pacote*
  - *em todas as outras classes*

- *Use para*
  - *construtores e métodos que fazem parte da interface do objeto*
  - *métodos estáticos utilitários*
  - *constantes (estáticas) utilitárias*
- *Evite usar em*
  - *construtores e métodos de uso restrito*
  - *campos de dados de objetos*

...entendendo melhor
Modificadores de acesso

protected

- *Acessível*
  - *na própria classe*
  - *nas subclasses*
  - *nas classes do mesmo pacote*
- *Use para*
  - *construtores que só devem ser chamados pelas subclasses (através de super())*
  - *métodos que só devem ser usados se sobrepostos*
- *Evite usar em*
  - *construtores em classes que não criam objetos*
  - *métodos com restrições à sobreposição*
  - *campos de dados de objetos*



...entendendo melhor Modificadores de acesso

package-private

- *Modificador ausente*
  - *se não houver outro modificador de acesso, o acesso é "package-private".*
- *Acessível*
  - *na própria classe*
  - *nas classes e subclasses do mesmo pacote*
- *Use para*
  - *construtores e métodos que só devem ser chamados pelas classes e subclasses do pacote*
  - *constantes estáticas úteis apenas dentro do pacote*
- *Evite usar em*
  - *construtores em classes que não criam objetos*
  - *métodos cujo uso externo seja limitado ou indesejável*
  - *campos de dados de objetos*

Classe
~campoAmigo: tipo
~metodoAmigo: tipo

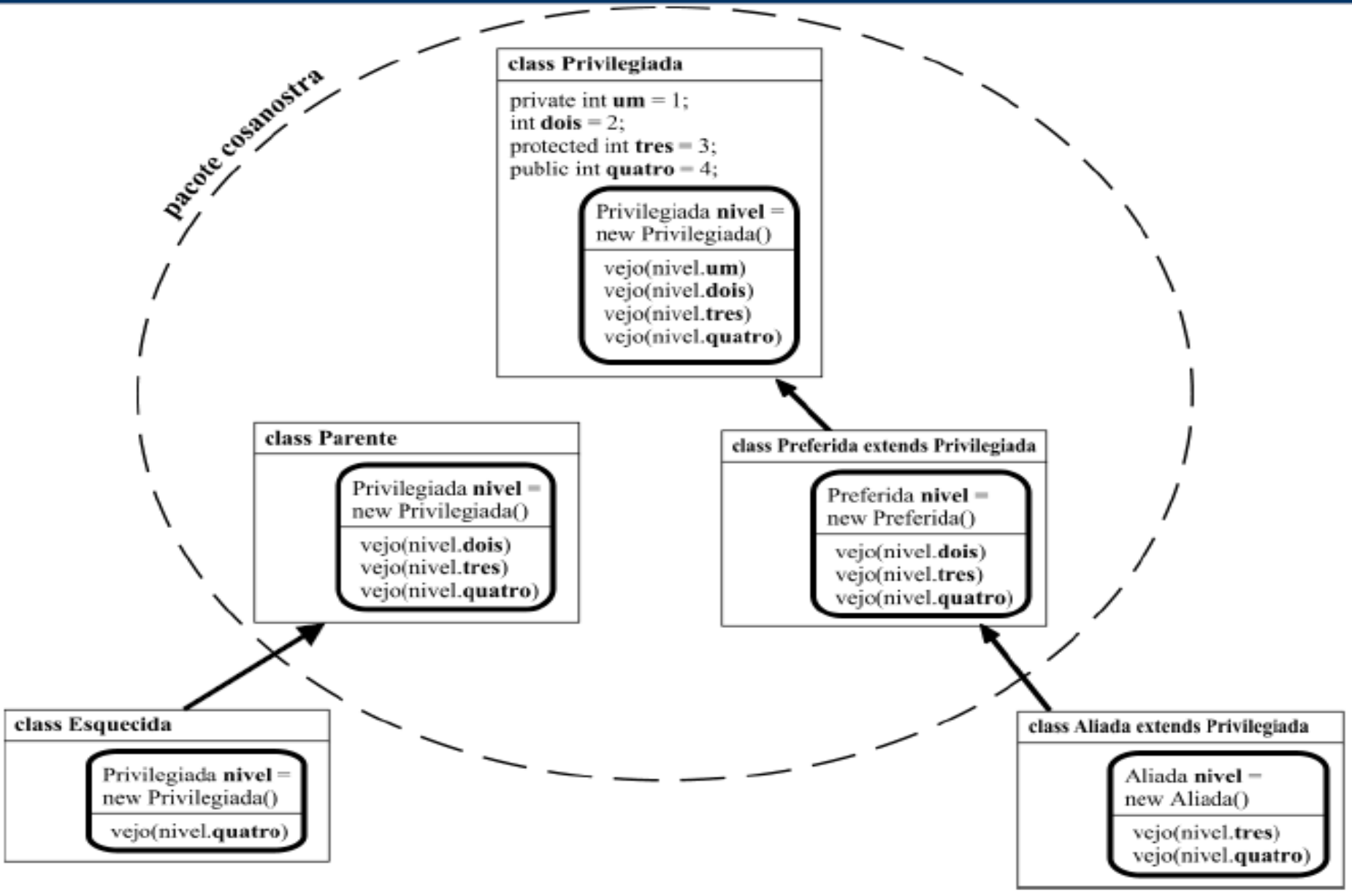
...entendendo melhor
Modificadores de acesso

private

- *Acessível*
  - *na própria classe (nos métodos, funções estáticas, blocos estáticos e construtores)*
- *Use para*
  - *construtores de classes que só devem criar um número limitado de objetos*
  - *métodos que não fazem parte da interface do objeto*
  - *funções estáticas que só têm utilidade dentro da classe*
  - *variáveis e constantes estáticas que não têm utilidade ou não podem ser modificadas fora da classe*
  - *campos de dados de objetos*

...estou querendo dizer que...

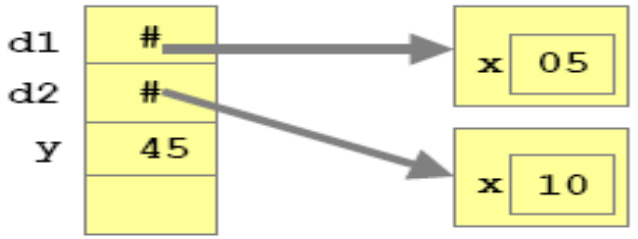
Exemplo



..voltando em

Mais sobre 'static'

- *Variáveis declaradas como 'static' existem antes de existir qualquer objeto da classe*
  - *Só existe uma variável static, independente do número de objetos criado com a classe*
  - *Podem ser chamadas externamente pelo nome da classe*
    Color.red, **System**.out, **BorderLayout**.NORTH
  - *Podem também ser chamadas através da referência de um objeto (evite usar este método)*




```
class Duas {
    int x;
    static int y;
}
```

```
(...)
Duas d1 = new Duas();
Duas d2 = new Duas();
d1.x = 5;
d2.x = 10;
//Duas.x = 60; // ilegal!
d1.y = 15;
d2.y = 30;
Duas.y = 45;
(...)
```

mesma variável!

Classe
campoStatic: tipo
metodoStatic: tipo

use esta notação apenas

..voltando em

Classe que só permite um objeto (Singleton pattern)

```
public class Highlander {
  private Highlander() {}
  private static Highlander instancia;
  public static Highlander criarInstancia() {
    if (instancia == null)
      instancia = new Highlander();
    }
    return instancia;
  }
}
```

Esta classe implementa o design pattern Singleton

```
public class Fabrica {
  public static void main(String[] args) {
    Highlander h1, h2, h3;
    //h1 = new Highlander(); // nao compila!
    h2 = Highlander.criarInstancia();
    h3 = Highlander.criarInstancia();
    if (h2 == h3) {
       System.out.println("h2 e h3 são mesmo objeto!");
    }
  }
}
```

*Esta classe cria apenas **um** objeto Highlander*

Exercícios

- agora:
- Escreva uma hierarquia de classes para as classes Pessoa, Aluno, Professor; Especifique as variáveis de instância e os métodos para cada classe. As variáveis de instância para a classe Pessoa deve ser nome e idade, para a classe Aluno nota e para a classe Professor salário. Sobrescreva o método toString em cada classe. Escreva um programa que instancia objetos Pessoa,

Fim

AMBIENT
INFORMÁTICA