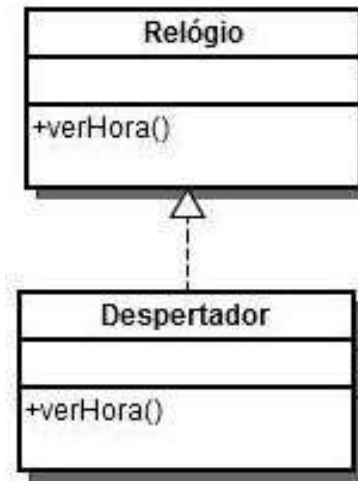




Interfaces

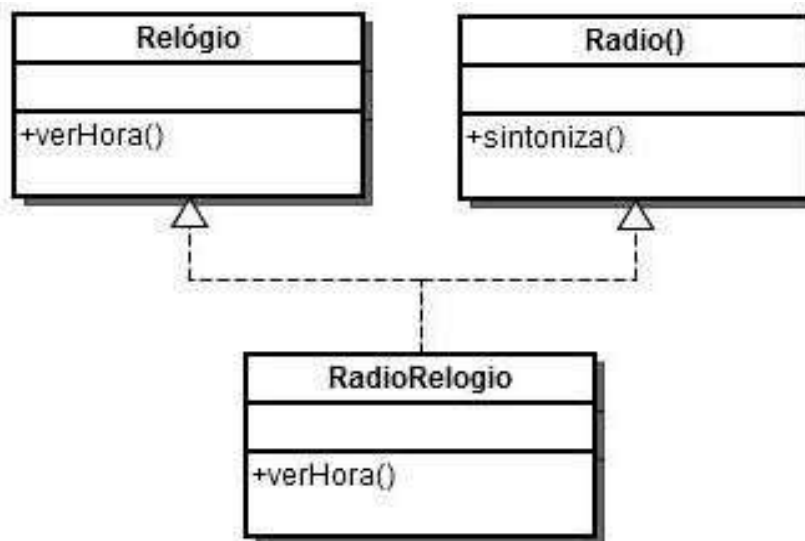
Profº Ms Gustavo Molina

MAIS UMA VEZ... RELEMBRANDO HERANÇA



```
Relógio cassio = new Relógio();
Despertador acorda = new Despertador();
...
cassio.verHora();
acorda.verHora();
// o método verHora() da classe Despertador
sobrescreve o
// método verHora() da classe Relógio
```

AGORA EU QUERO... UM RADIO-RELÓGIO



Será que uma subclasse pode ser herdeira de mais de uma superclasse?

Será que a classe **RadioRelógio** pode ser herdeira de **Relógio** e de **Radio** ao mesmo tempo?

A resposta é NÃO. A linguagem Java não permite herança múltipla

Porque Java não permite herança múltipla?

O Java não permite a herança múltipla, porque se tivermos duas classes, onde cada uma tem exatamente o mesmo método (mesma assinatura) e com corpo diferente não seria possível identificar qual deles deveria ser utilizado.

Exemplo (Este código não funciona!!!)

```
class Brasil {  
    void contarAte5() {  
        System.out.println("um, dois, três, quarto, cinco");  
    }  
}  
  
class Inglaterra  
    void contarAte5() {  
        System.out.println("one, two, three, four, five");  
    }  
}  
  
class Aluno extends Brasil, Inglaterra {  
    Aluno() {  
        this.contarAte5();  
    }  
}
```

Exemplo (Este código não funciona!!!)

```
class Brasil {
    void contarAte5() {
        System.out.println("um, dois, três, quarto, cinco");
    }
}

class Inglaterra
    void contarAte5() {
        System.out.println("one, two, three, four, five");
    }
}

class Aluno extends Brasil, Inglaterra {
    Aluno() {
        this.contarAte5();
    }
}
```

Qual dos dois métodos será utilizado?

contarAte5() da classe Brasil ou **contarAte5()** da classe Inglaterra?

O significa INTERFACE?

Novo dicionário
Aurélio

Pesquisa:

☐ Palavras iniciadas por

☐ Palavras terminadas em

☐ Procurar no texto do verbete

Mais filtros ▾

BUSCAR

interessado
interessante
interessar
interesse
interesseiro
interestadual
interestatal
interestelar
interétnico
intereuropeu
interface
interfacial

interface

[De *inter-* + *face*; ingl. *interface*.]

Substantivo feminino.

1. Dispositivo físico ou lógico que faz a adaptação entre dois sistemas.
2. Conjunto de elementos comuns entre duas ou mais áreas de conhecimento, de interesse, etc.:
A interface entre a física e a química.
3. **Comun.** Meio que promove a comunicação ou interação entre dois ou mais grupos:
O departamento de Relações Públicas funciona como interface entre a empresa e seus diversos públicos-alvos.
4. **Ecol.** Área de fronteira entre regiões adjacentes, e que constitui ponto em que interagem sistemas independentes de diversos grupos.
5. **Fis.** Superfície que separa duas fases de um sistema.
6. **Inform.** Interconexão entre dois equipamentos que possuem diferentes funções e que não se poderiam conectar diretamente, como, p. ex., o *modem* (q. v.).

O significa INTERFACE?

A INTERFACE é um meio na relação entre dois elementos



INTERFACE Controle Remoto

- Tem uma série de operações disponíveis (ligar, desligar, etc.)
- Não funciona sozinho
- A televisão, por sua vez, deve estar preparada para receber os sinais do controle remoto e executar as ações ligar, desligar, etc.
- Seria como se a televisão implementasse aquilo que o controle remoto só indicou que deve ser feito
- O que aconteceria se a televisão não estivesse preparada para executar as operações disponíveis no controle remoto? O controle não serviria para nada...
- Todas as operações previstas no controle remoto devem estar “implementadas” na televisão

INTERFACES x CLASSES

Escrever uma Interface é similar à escrita de uma classe, porém, existem conceitos diferentes:

- Uma **classe** é um conjunto de atributos e comportamento de um objeto;
- Uma **interface** contém o comportamento que uma classe deverá implementar (o controle remoto tem um botão desliga, que a televisão, por sua vez, precisa implementar seu reconhecimento e ações relacionadas...)
- Uma interface permite a implementação de **comportamentos polimórficos**

INTERFACES em Java (I)

- Pode conter qualquer quantidade de métodos;
- Uma interface é escrita em um arquivo com uma extensão .java e o nome da interface deve ser o mesmo nome do arquivo;
- Não é possível instanciar uma interface, ou seja, criar um objeto através do operador **new**;
- Uma interface não tem métodos construtores;
- Todos os métodos de uma interface são **abstratos** (os prefixos **abstract** e **public** podem ser omitidos);

INTERFACES em Java (II)

- Uma interface não pode conter variáveis de instância (atributos). As únicas variáveis que podem existir em uma interface devem ser declaradas com **static** e **final** (o prefixo **static** normalmente é omitido).
- Uma interface não é estendida por uma classe (**extends**). Ela é implementada (**implements**).
- Pode ser criada como “subinterface” de outra interface já existente, usando **extends**, como as classes.

EXEMPLO – ZOOLOGICO

Será criada uma **interface** para manipular os diferentes animais de um zoológico. Todos os animais do zoológico, independentemente de seu tipo, **nascem**, se **movimentam** e **dormem**. Estas ações, comuns a todos os animais são as **operações** da interface Animal. Veja o código:

```
public interface Animal {  
    void nasce();  
    void movimentar();  
    void dormir();  
}
```

EXEMPLO – ZOOLOGICO

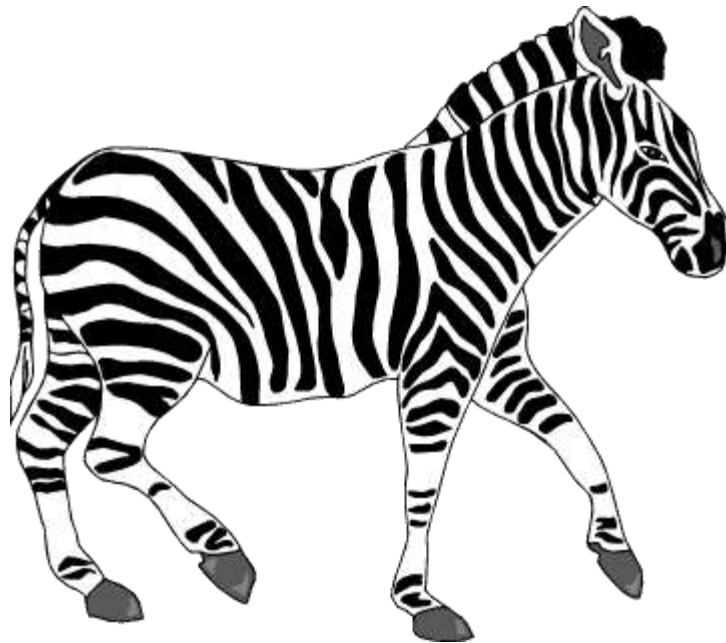
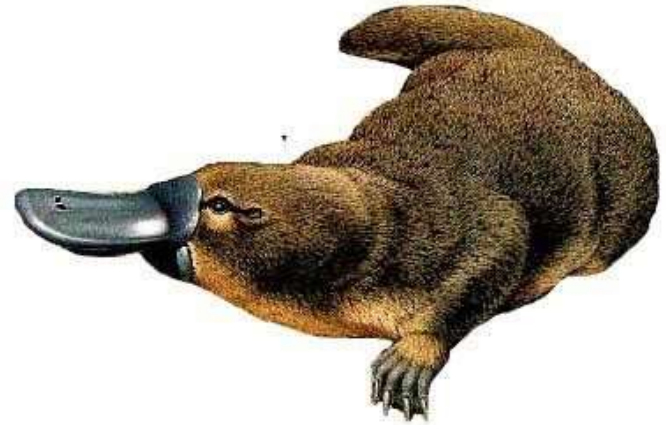
Cada animal que for implementado no projeto deverá informar explicitamente que irá usar a interface Animal, ou seja, irá conter as operações (métodos) `nasce()`, `movimenta()` e `dorme()`.

Em cada classe, porém, estes métodos deverão estar com sua implementação completa, já que na interface, só foi feita uma referência à assinatura do método.

```
public interface Animal {  
    void nasce();  
    void movimenta();  
    void dorme();  
}
```

Assinatura do
método

ANIMAIS DO ZOOLÓGICO

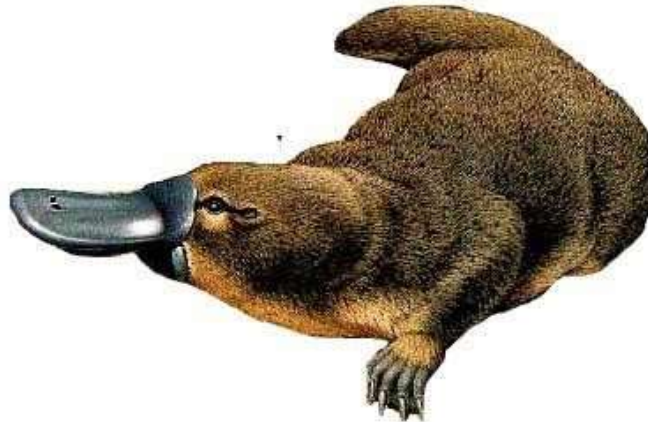


CLASSE MORCEGO



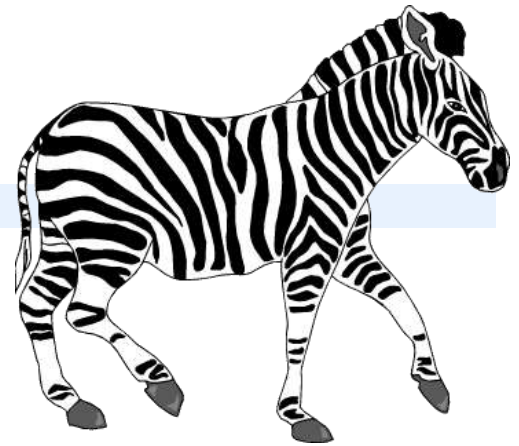
```
public class Morcego implements Animal {  
  
    public void nasce() {  
        System.out.println("Nasce um lindo morcego...");  
    }  
  
    public void movimenta() {  
        System.out.println("O morcego voa de lá para cá...");  
    }  
  
    public void dorme() {  
        System.out.println("E dorme de cabeça para baixo!");  
    }  
  
}
```


CLASSE ORNITORRINCO



```
public class Ornitorrinco implements Animal {  
  
    public void nasce() {  
        System.out.println("O ornitorrinco nasce de um ovo...");  
    }  
  
    public void movimenta() {  
        System.out.println("Anda de um lado para o outro...");  
    }  
  
    public void dorme() {  
        System.out.println("E dorme durante o dia dentro de túneis");  
    }  
}
```

CLASSE ZEBRA



```
public class Zebra implements Animal{

    int qtdeListas;

    Zebra(int qtdeListas) {
        this.qtdeListas = qtdeListas;
    }

    public void nasce() {
        System.out.print("A zebra nasce cheia de listas...");
    }

    public void movimenta() {
        System.out.println("Fica galopando pelo campo...");
    }

    public void dorme() {
        System.out.println("E dorme em pé!");
    }

    public void mostraListas() {
        System.out.println("Esta zebra tem " + qtdeListas + " listas");
    }
}
```

No método main()....

```
public class TestaZoo {  
    public static void main(String[] args) {  
        System.out.println ("Vamos passear pelo Zoológico?");  
        System.out.println ("\nPrimeiro vamos conhecer o morcego");  
        Morcego animal1 = new Morcego();  
        animal1.nasce();  
        animal1.movimenta();  
        animal1.dorme();  
  
        System.out.println ("\nAgora é a vez do ornitorrinco!");  
        Ornitorrinco animal2 = new Ornitorrinco();  
        animal2.nasce();  
        animal2.movimenta();  
        animal2.dorme();  
  
        System.out.println ("\nPor último, a zebra!");  
        Zebra animal3 = new Zebra(10);  
        animal3.nasce();  
        animal3.mostraListas();  
        animal3.movimenta();  
        animal3.dorme();  
    }  
}
```

TELA DE SAÍDA

Vamos passear pelo Zoológico?

Primeiro vamos conhecer o morcego

Nasce um lindo morcego...

O morcego voa de lá para cá...

E dorme de cabeça para baixo!

Agora é a vez do ornitorrinco!

O ornitorrinco nasce de um ovo...

Anda de um lado para o outro...

E dorme durante o dia dentro de túneis

Por último, a zebra!

A zebra nasce cheia de listas...Esta zebra tem 10 listas

Fica galopando pelo campo...

E dorme em pé!

SIMULANDO HERANÇA MÚLTIPLA

Usamos este recurso quando queremos que uma classe tenha um comportamento igual a duas ou mais classes.

Uma classe pode ser herdeira de apenas uma classe, porém, pode **implementar várias interfaces**.

```
public class classe0 extends classe1 implements  
                                     classe2, classe3, classe4 {  
    ...  
}
```

EXEMPLO

