```java
package interfaces;

import java.util.Date;

public interface Animal {
	/* Interfaces em Java só contem a interface dos métodos (sem corpo) e
	 * pondem conter constantes (declaradas como atributos), podem ou
	 * não ter os modificadores static e final (funcionarão como se tivessem) */
	int constanteDaInterface = 5;

	// a assinatura de um método em uma interface pode ter os seguintes modificadores:
	// <nada>, public, abstract
	void nasça(Date dataDeNascimento);
	void tipo();
	void durma();
	double peso();
}
```

```java
package interfaces;

public interface Voador {
	void voe();
	void aterrisse();
}
```

```java
package interfaces;

import java.util.Date;

public class Cachorro implements Animal{
/* A classe Cachorro PRECISA implementar todos os metodos definidos na
 * Interface Animal, além disso pode ter outros atributos e métodos  */

	private int peso;

	Cachorro(int peso){
		this.peso = peso;
	}

	public void durma() {
		System.out.println("Dormindo.");
	}

	public void nasça(Date data) {
		System.out.println("Nascendo um novo cachorro: " + data);
	}

	public double peso() {
		return peso;
	}

	public void tipo() {
		System.out.println("Este animal eh um Cachorro.");
	}
}
```

```java
package interfaces;

import java.util.Date;

/* A classe Morcego PRECISA implementar todos os metodos definidos nas
 * interfaces Animal e Voador. Note que se as duas interfaces têm métodos com
 * assinaturas/cabeçalhos idênticos isto não é um problema, porém se são métodos
 * com os mesmos nomes e parâmetros porém com tipos de retorno diferentes,
 * daí não é possível implementar as duas interfaces.
 */
public class Morcego implements Animal, Voador {

	public Date dataDeNascimento;

	public void durma() {
		System.out.println("Dormindo de cabeça para baixo.");
	}

	public void nasça(Date data) {
		dataDeNascimento = data;
		System.out.println("Nascendo um novo morcego");
	}

	public double peso() {
		return 1.5*constanteDaInterface;
	}

	public void tipo() {
		System.out.println("Este animal eh um Morcego.");
	}

	public void aterrisse() {
		System.out.println("Aterrissando.");
	}

	// Ao implementar métodos de uma interface, o método NÃO pode ser estático
	public void voe() {
		System.out.println("Levantando voo.");
	}
}
```

```java
package interfaces;

import java.util.Date;

public class ExecutaZoologico {

	public static void main(String[] args) {
		Animal zoologico[] = new Animal[5];
		zoologico[0] = new Cachorro(10);
		zoologico[1] = new Morcego();
		zoologico[2] = new Cachorro(20);
		zoologico[3] = new Morcego();
		zoologico[4] = new Cachorro(30);
		for (int cont=0;cont<5;cont++){
			zoologico[cont].tipo();
			zoologico[cont].durma();
			System.out.println();
		}
		System.out.println(Animal.constanteDaInterface);
	}

}

/* RESULTADO DA EXECUÇÃO
Este animal eh um Cachorro.
Dormindo.

Este animal eh um Morcego.
Dormindo de cabeça para baixo.

Este animal eh um Cachorro.
Dormindo.

Este animal eh um Morcego.
Dormindo de cabeça para baixo.

Este animal eh um Cachorro.
Dormindo.

5
*/
```