

TI-220 Java Orientado a Objetos

ANTONIO CARVALHO - TREINAMENTOS

A solid blue horizontal bar spanning the width of the slide, located at the bottom.

Estrutura da Memória

Estrutura de Memória

A memória da JVM pode ser dividida em duas áreas simples

- **Stack**
- **Heap**

Estrutura de Memória

A área **Stack** guarda os **frames** das funções e as variáveis locais.

As chamadas de funções são empilhadas nesta área, de modo que cada nova chamada a uma função cria um novo **frame**.

As variáveis locais vivem enquanto o frame estiver criado.

Estrutura de Memória

A área **Heap** guarda as **instâncias** dos objetos.

As variáveis de instância existem enquanto a instância existir.

Estrutura de Memória

Veja como o código abaixo se comporta na memória

```
public class Pneu {  
    int aro;  
    public void rodar() {  
        System.out.println("rodando");  
    }  
}
```

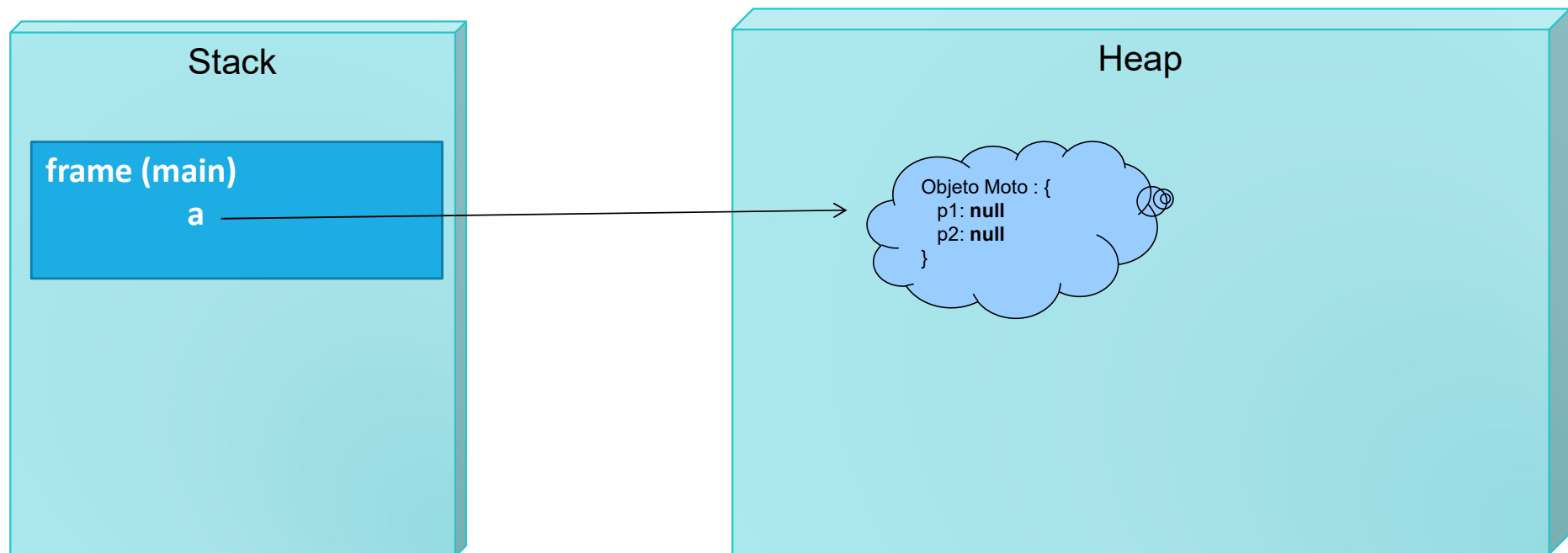
```
public class Moto {  
    Pneu p1, p2;  
    public void criar() {  
        int j = 5;  
        p1 = new Pneu();  
        p2 = new Pneu();  
    }  
}
```

```
public class Teste {  
    public static void main(String args[]) {  
        Moto a = new Moto();  
        a.criar();  
    }  
}
```

Estrutura de Memória

Comportamento dos objetos e variáveis de referência na memória.

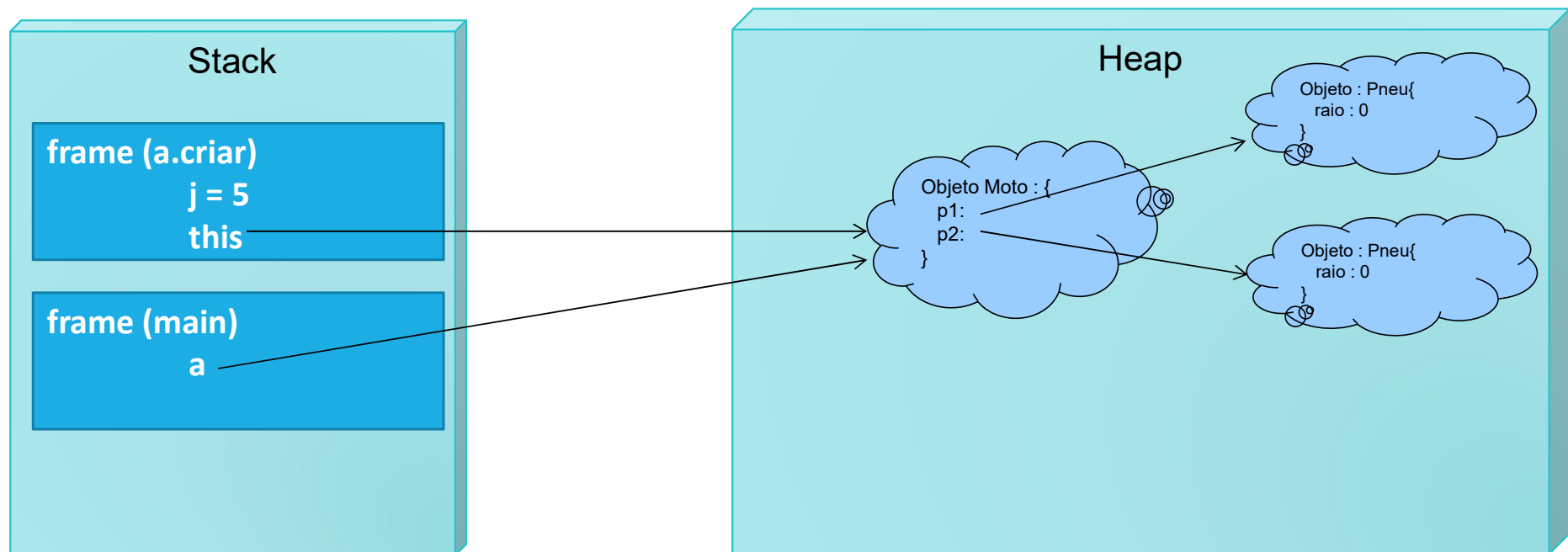
Ao executar o método **main**



Estrutura de Memória

Comportamento dos objetos e variáveis de referência na memória.

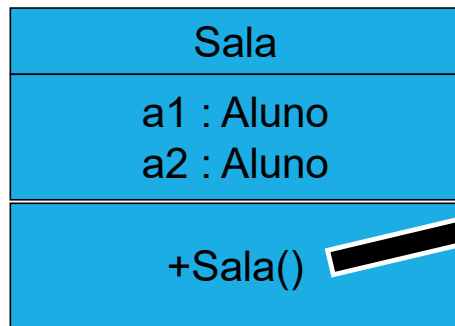
Quando o método **criar** é acionado



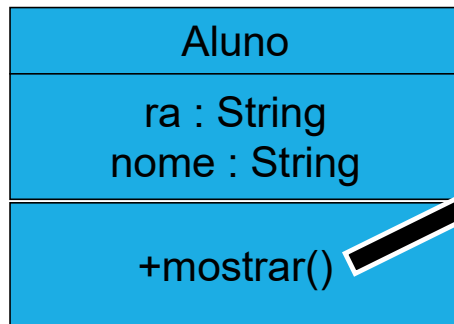
Dúvidas



Exercício



```
public Sala() {  
    int valor = 12;  
    a1 = new Aluno();  
    a2 = new Aluno();  
    a1.mostrar();  
}  
public void mostrar() {  
    System.out.println( ra + "-" + nome );  
}
```



Faça:

1. O código completo das classes **Aluno** e **Sala**
2. Diagrama de memória do Stack e Heap, mostrando todos os frames que haverá na memória quando ocorrer a chamada ao método **a1.mostrar()**