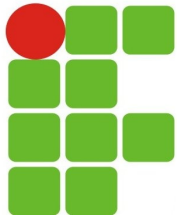


Classes e Objetos

João Paulo Q. dos Santos

joao.queiroz@ifrn.edu.br

Roteiro

- Criando objetos:
  - Operador new;
  - Passagem por Referência;
    - As áreas em memória da JVM;
    - Stack e Heap;
- Manipulando objetos:
  - Chamando métodos;
- Destruição de objetos:
  - O *garbage collector* (gc);

Criação de objetos

- Declarar variável
  - Associa variavel a tipo (classe)
  - Sintaxe
    - NomeClasse nomeVariável;
- Exemplo
  - Circulo c1;
- Criar objeto (instanciar) e fazer variável referenciar o objeto
  - c1 = new Circulo();
- Ambos em um passo
  - Circulo c1 = new Circulo();

Áreas em memória da JVM

- Uma variável é um nome associado a uma posição de memória;
- Na sua declaração, o compilador aloca um espaço para esta variável;
- É importante considerar duas áreas para a alocação e execução de dados na memória principal (RAM) as quais são utilizadas pela *Java Virtual Machine* (JVM) durante a execução de um programa: a ***stack*** e a ***heap***.

Heap (pool de memória)

- A **heap** é um local da memória que armazena todos os objetos que serão utilizados no seu programa;
- Quando um objeto é instanciado, esses objetos criados e seus respectivos parâmetros são automaticamente alocados na **heap**;
- **Quando um método que utiliza um objeto é:**
  - Finalizado;
  - Uma exceção ocorre, ou;
  - O número de referências do objeto passa a null, ou;
  - Quando *threads* que fazem uso deste são finalizadas;

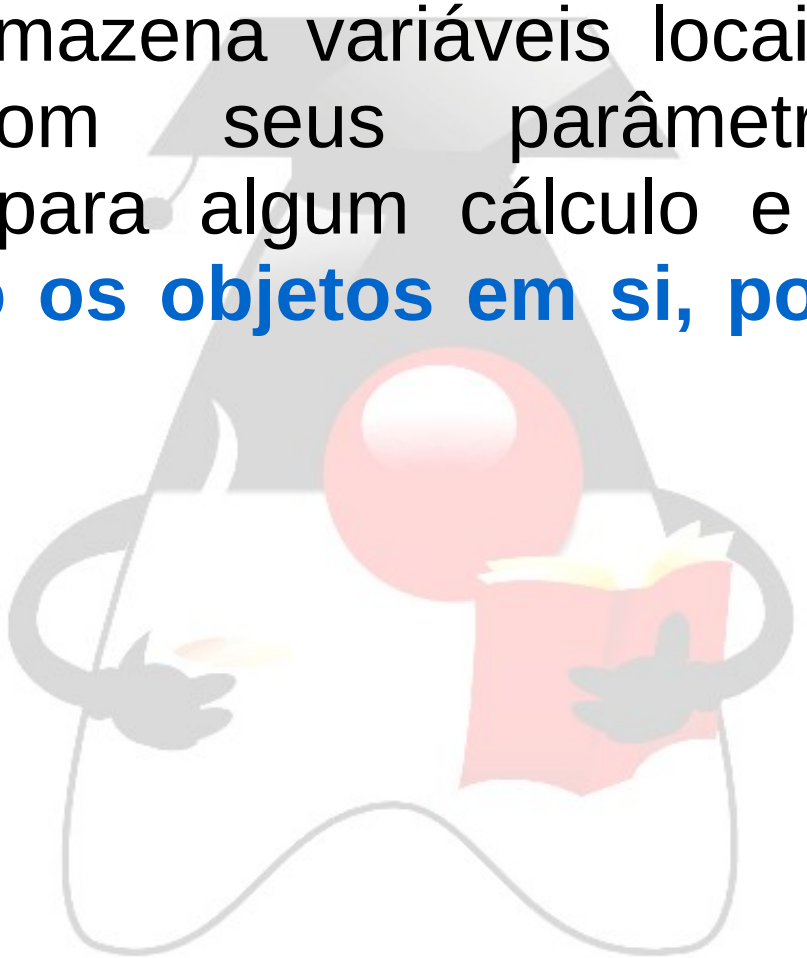
Este objeto fica passível de ser coletado pelo *Garbage Collector* (coletor de lixo)

Stack (pilha)

- A *stack* ("pilha") é uma área com suporte direto ao processador, através de seu apontador de pilha (*stack pointer* - **sp**).
- Ela empilha ("push") para criar nova área memória, e Desempilha ("pop") para liberar a área de memória criada;
- Cabe ao **sp** guardar o endereço do próximo endereço livre na stack (o topo da stack (Pilha)).

Stack (pilha)

- Em geral, armazena variáveis locais, chamadas a métodos com seus parâmetros, variáveis temporárias para algum cálculo e **referências a objetos** (**não os objetos em si, pois estes ficam na heap**).



Stack e Heap em funcionamento

- O que ocorre em memória quando um objeto é alocado? Dada uma classe **Pessoa**, cria-se uma instância **p** em memória de **Pessoa**.

```
Pessoa p = new Pessoa ();
```

Instanciando um objeto tipo Pessoa

É utilizada uma área em memória da stack (com a variável de referência "p") para armazenar o endereço de memória heap, aonde foi criada (logicamente) a estrutura do objeto Pessoa.

Importante: Stack e Heap

- Um aspecto a ser observado é que, ao ser feita a declaração na **stack (**de Pessoa p**;)** o objeto ainda não existe em memória:
  - Exemplo: Pessoa p**;**
- Isso porque, esse objeto só irá efetivamente existir em memória quando for utilizado o operador new Pessoa (), o qual alocará esse objeto na área heap (aonde os objetos residem):

```
p = new Pessoa();
```

- Ambos em um passo:

```
Pessoa p = new Pessoa();
```

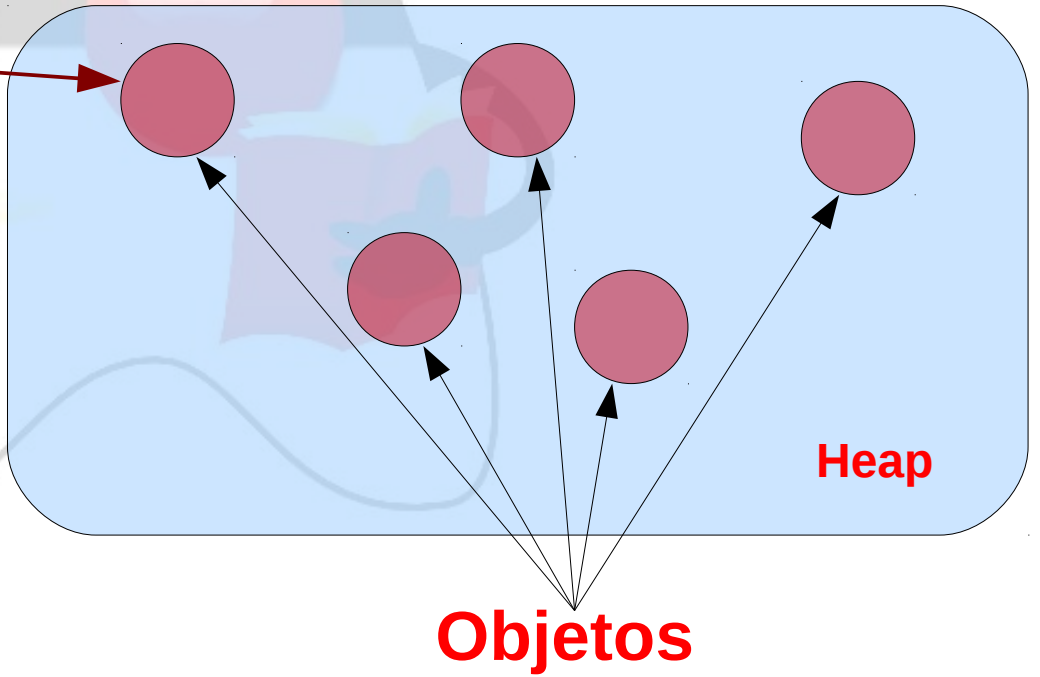
Criação de objetos

```
Classe obj1;

obj1 = new Classe();
```

Declara obj1 como referência para objeto da classe Classe.

Cria objeto e faz obj1 referenciar objeto recém-criado.

Obj1

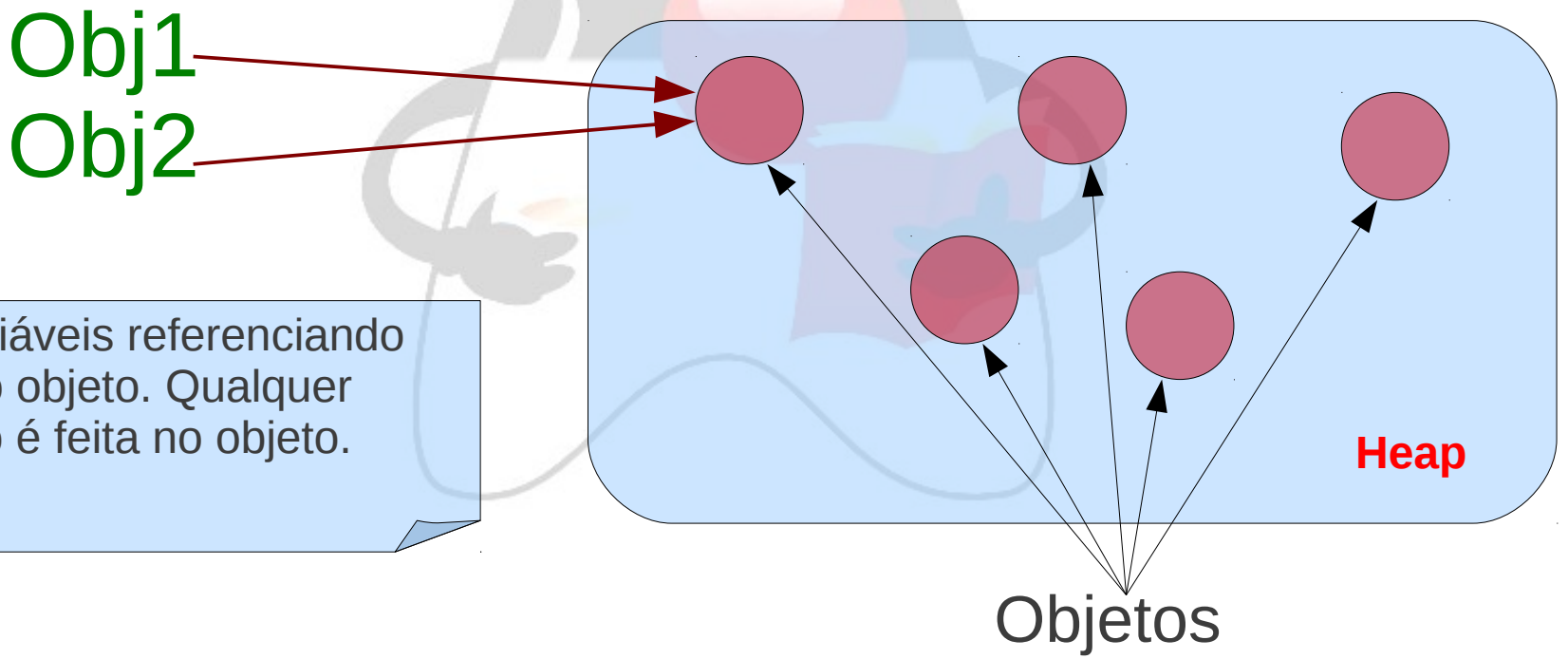


A variável obj1 armazena uma referência para o objeto em si. Seu conteúdo é o "endereço" do objeto.

Referência

Classe obj1, obj2;
obj1 = new Classe();
obj2 = obj1

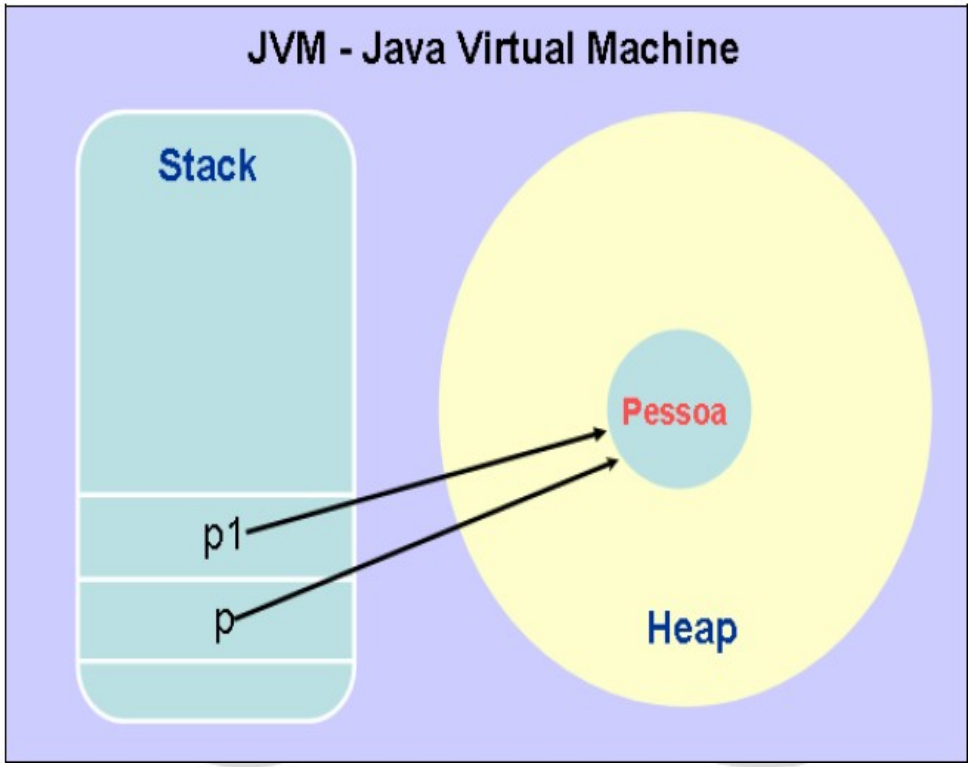
faz obj2 referenciar o mesmo objeto que obj1 referência



Duas variáveis referenciando o mesmo objeto. Qualquer alteração é feita no objeto.

Stack e Heap em funcionamento

```
Pessoa p = new Pessoa ();
p1 = p;
```

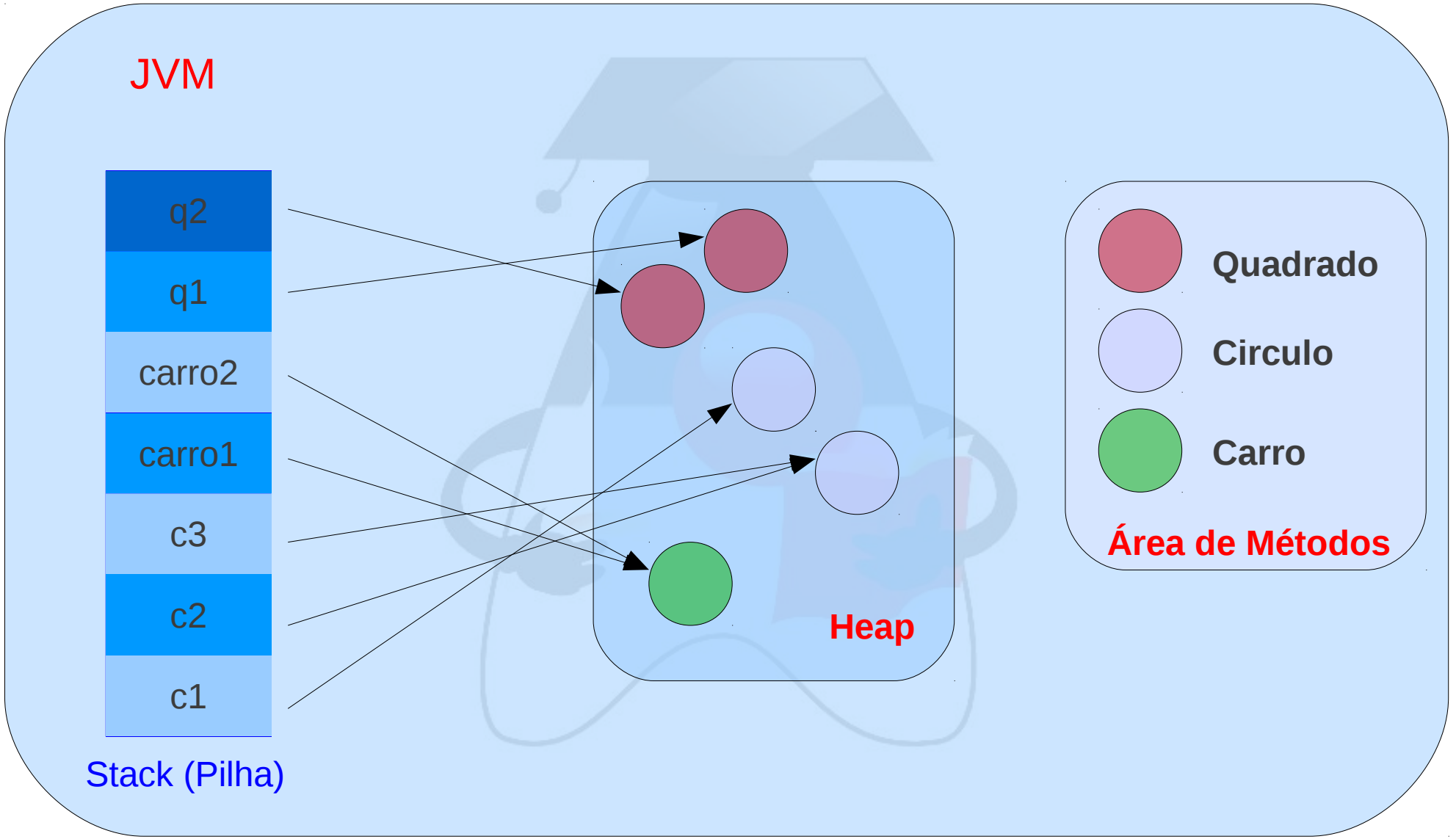


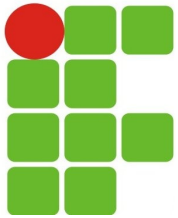
Exercício

- Desenhe a heap e stack para o código a seguir:

```
Circulo c1,c2,c3;
Carro carro1, carro2;
c1 = new Circulo();
Quadrado q1 = new Quadrado();
c2 = c1;
carro1 = new Carro();
Quadrado q2 = q1;
q1 = new Quadrado();
c3 = c2;
c1 = new Circulo();
carro2 = carro1;
```

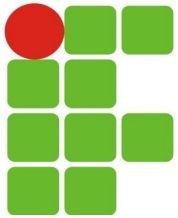
Resposta


JVM
q2
q1
carro2
carro1
c3
c2
c1
Stack (Pilha)
Heap
Quadrado
Circulo
Carro
Área de Métodos



Stack e heap em funcionamento

```
public class Quadrado {
    private int lado;
    public Quadrado (int lado){
        this.setLado(lado);
    }
    public void setLado(int lado) {
        this.lado = lado;
    }
    public int area() {
        return this.lado*this.lado;
    }
}
```



Stack e heap em funcionamento

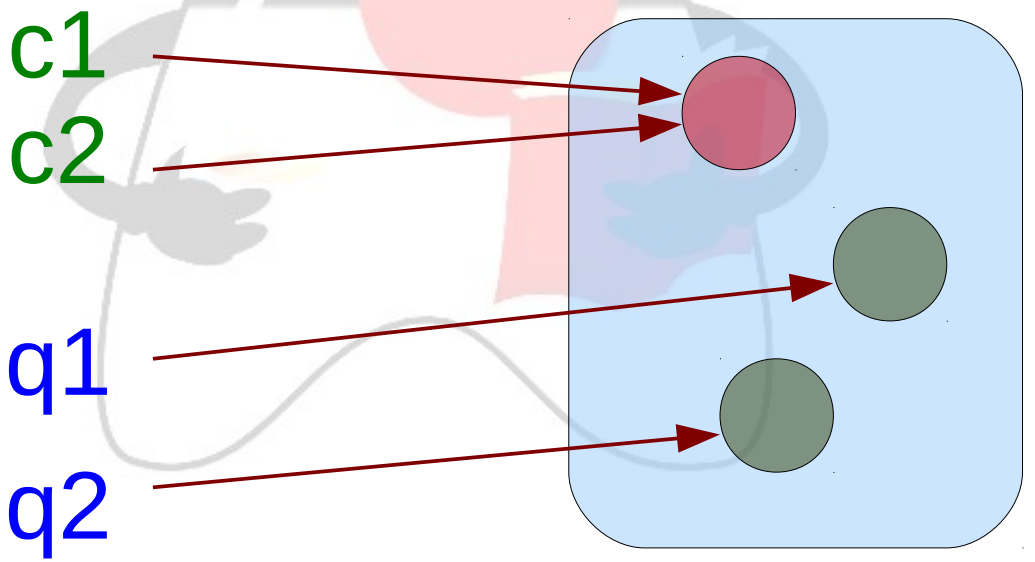
```
public class AplicacaoQuadrado {

    public static void main(String[] args) {

    Quadrado q1 = new Quadrado(2);

    Quadrado q2 = new Quadrado(3);

    System.out.println("Area q1 e: "+q1.area());
    System.out.println("Area q2 e: "+q2.area());

    q1 = q2;

    q2.setLado(4);

    System.out.println("Area q1 e: "+q1.area());

    }
}
```

Igualdade

- Entre variáveis:
  - Compara o valor das variáveis;
  - O valor de uma variável para um objeto é o endereço do objeto:
    - O operador **==** compara se as duas variáveis referenciam o mesmo objeto:
      - obj1 == obj2;
- Entre objetos:
  - Método equals verifica se dois objetos possuem o mesmo estado (são iguais).

Igualdade

- c1 == c2
  - true
    - Referenciam o mesmo obj
- q1 == q2
  - false
    - Mesmo que os objetos sejam iguais



Métodos

- Usamos o operador “.” (ponto):
  - Sintaxe:
    - objeto.método();
  - Executa método que está contido no objeto;
- Objeto deve existir:
  - A variável deve referenciar objeto válido;
    - Se referenciar null ocorre erro;
- Exemplos:

```
obj1.nomeMetodo();
obj1.nomeMetodo(arg1, arg2);
(new NomeClasse()).nomeMetodo();
obj1.nomeAtributo;
```

Destruindo Objetos

- Garbage Collector (gc):
  - Objetos não referenciados são automaticamente apagados:
    - Quando isso é feito depende do algoritmo de coleta de lixo usado;
- System.gc();
  - Força a coleta de lixo.

Dúvidas