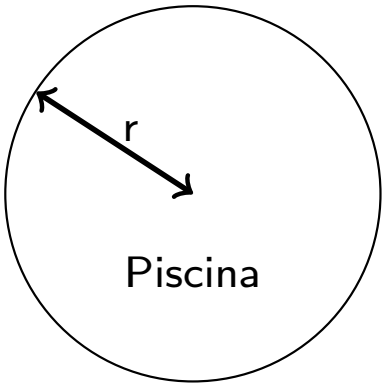
Aula 05 – Subrotinas

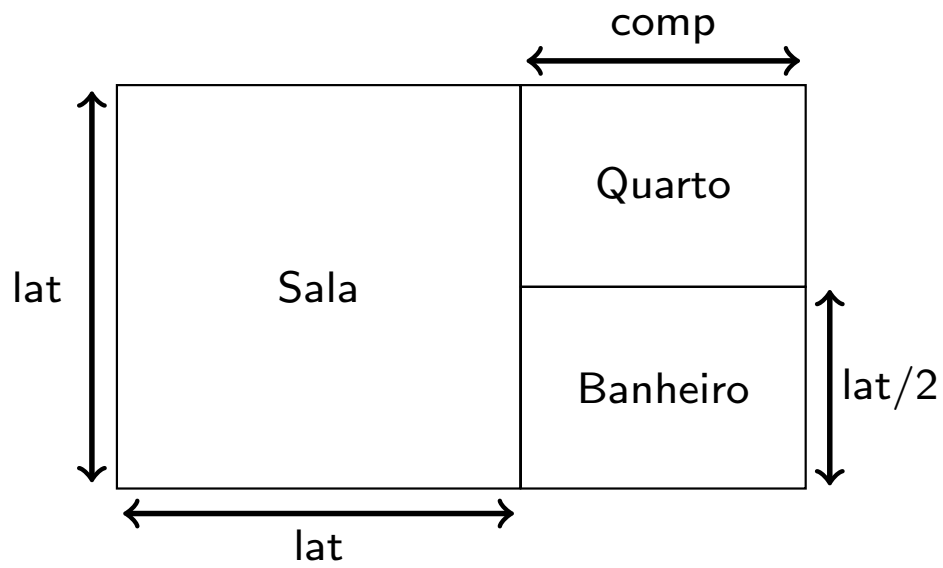
Norton Trevisan Roman

9 de abril de 2018

Constantes

- Suponha que queremos incrementar nossa cabana com uma piscina:





- Queremos então fazer um programa que calcule a área da cabana e da piscina

Constantes

- Como?

Constantes

- Como?
  - Temos o raio da piscina
  - Basta vermos como adicionar o π

Constantes

- Como?
  - Temos o raio da piscina
  - Basta vermos como adicionar o π
- Podemos fazer:

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    areap = pi * raio * raio;
    System.out.println("Área: "+
                        areap);
  }
}
```

Constantes

- Como?
  - Temos o raio da piscina
  - Basta vermos como adicionar o π
- Podemos fazer:
  - E a saída será "Área: 12.56636"

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    areap = pi * raio * raio;
    System.out.println("Área: "+
                          areap);
  }
}
```

Constantes

- E se fizermos:

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    pi = 12;
    areap = pi * raio * raio;
    System.out.println("Área: "+
                          areap);
  }
}
```

Constantes

- E se fizermos:
  - Teremos "`Área:  48.0`"
  - Inadvertidamente mudamos algo que deveria ser constante

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    pi = 12;
    areap = pi * raio * raio;
    System.out.println("Área: "+
                          areap);
  }
}
```

Constantes

- Devemos tornar π constante, fazendo:

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    final double pi = 3.14159;


    areap = pi * raio * raio;
    System.out.println("Área: "+
                          areap);
  }
}
```

Constantes

- Devemos tornar π constante, fazendo:
- E, se tentarmos mudar o valor, teremos

```
$ javac AreaPiscina.java
AreaPiscina.java:11: cannot
assign a value to final
variable pi
        pi = 12;
        ^
1 error
```

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    final double pi = 3.14159;

    pi = 12;
    areap = pi * raio * raio;
    System.out.println("Área: "+
                          areap);
  }
}
```

Constantes

- "final" especifica que o valor não mais poderá ser mudado no programa
- Define uma constante

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    final double pi = 3.14159;

    areap = pi * raio * raio;
    System.out.println("Área: "+
                          areap);
  }
}
```

Constantes

- Alternativamente, podemos usar uma constante já definida no java:
  - `Math.PI`, valendo 3.141592653589793
  - `Math.PI` é double, por isso *areap* também o é

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;

    areap = Math.PI * raio
                          * raio;
    System.out.println("Área: "+
                           areap);
  }
}
```

Constantes

- E como podemos mudar `raio * raio`?

```
class AreaPiscina {
  public static void main(
              String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;

    areap = Math.PI * raio
                        * raio;
    System.out.println("Área: "+
                          areap);
  }
}
```

Constantes

- E como podemos mudar `raio * raio`?
- `Math.pow(a,b)` dá o resultado de a^b
  - O resultado também é *double*

```
class AreaPiscina {
  public static void main(
                String[] args) {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;

    areap = Math.PI *
              Math.pow(raio,2);
    System.out.println("Área: "+
                              areap);
  }
}
```

Métodos

- Math é como se fosse uma biblioteca, que nos fornece o método pow (além da constante PI)
- Não é uma biblioteca, é mais que isso, mas veremos mais adiante

```
class AreaPiscina {
  public static void main(
                String[] args){
    double raio = 2;
    double areap;


    areap = Math.PI *
          Math.pow(raio,2);
    System.out.println(
          "Área: "+ areap);
  }
}
```

Métodos

- Math é como se fosse uma biblioteca, que nos fornece o método pow (além da constante PI)
- Não é uma biblioteca, é mais que isso, mas veremos mais adiante
- Método?

```
class AreaPiscina {
  public static void main(
                String[] args){
    double raio = 2;
    double areap;


    areap = Math.PI *
          Math.pow(raio,2);
    System.out.println(
          "Área: "+ areap);
  }
}
```

Métodos

- Um método é uma implementação de uma subrotina
- Nesse caso, `pow(a,b)` recebe dois valores, a e b, devolvendo o resultado de a^b
  - Os valores a e b fornecidos ao método são chamados argumentos de seus parâmetros

```
class AreaPiscina {
  public static void main(
                String[] args){
    double raio = 2;
    double areap;


    areap = Math.PI *
          Math.pow(raio,2);
    System.out.println(
          "Área: "+ areap);
  }
}
```

Métodos

Vamos então juntar os dois programas que vimos até agora em um só:

```
class AreaCasa {
  public static void main(
            String[] args) {
    float lateral = 11;
    float cquarto = 7;
    float areaq;
    float areas;
    float areat;
    double raio = 2;
    double areap;

    System.out.println(...);
    areas = lateral*lateral;
    System.out.println("..."+areas);
    areaq = cquarto*(lateral/2);
    System.out.println("..."+areaq);
    System.out.println("..."+areaq);
    areat = areas + 2*areaq;
    System.out.println("..."
                          + areat);
    areap = Math.PI *
                  Math.pow(raio,2);
    System.out.println("..."+areap);
  }
}
```

Métodos

E qual a saida?

Métodos

E qual a saida?

```
Cálculo da área da casa
A área da sala é 121.0
A área do banheiro é 38.5
A área do quarto é 38.5
A área total é 198.0
A área da piscina é 12.566370614359172
```

Métodos

- Esse programa está ficando confuso:
  - Mistura a casa com a piscina

Métodos

- Esse programa está ficando confuso:
  - Mistura a casa com a piscina
- Que fazer?

Métodos

- Esse programa está ficando confuso:
  - Mistura a casa com a piscina
- Que fazer?
  - Podemos dividi-lo em 2 partes: uma para o cálculo da casa e outra para o cálculo da piscina

Métodos

- Esse programa está ficando confuso:
  - Mistura a casa com a piscina
- Que fazer?
  - Podemos dividi-lo em 2 partes: uma para o cálculo da casa e outra para o cálculo da piscina
- Como?

Métodos

- Esse programa está ficando confuso:
  - Mistura a casa com a piscina
- Que fazer?
  - Podemos dividi-lo em 2 partes: uma para o cálculo da casa e outra para o cálculo da piscina
- Como?
  - Criando nossos próprios métodos

Métodos

```
static void areaCasa() {
  float lateral = 11;
  float cquarto = 7;
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..." + areat);
}
```

```
static double areaPiscina()
{
  double raio = 2;

  return(Math.PI *
         Math.pow(raio,2));
}
```

Métodos

```
static void areaCasa() {
  float lateral = 11;
  float cquarto = 7;
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..." + areat);
}
```

```
static double areaPiscina()
{
  double raio = 2;

  return(Math.PI *
         Math.pow(raio,2));
}
```

Ambos dentro do mesmo programa...

Métodos

- O que significa o `void`?

```
static void areaCasa() {
  float lateral = 11;
  float cquarto = 7;
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos

- O que significa o `void`?
  - Que o método não irá retornar nenhum valor
  - Ele apenas executa a tarefa e termina

```
static void areaCasa() {
  float lateral = 11;
  float cquarto = 7;
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos

- O que significa o `void`?
  - Que o método não irá retornar nenhum valor
  - Ele apenas executa a tarefa e termina
- E o static?

```
static void areaCasa() {
  float lateral = 11;
  float cquarto = 7;
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos

- O que significa o `void`?
  - Que o método não irá retornar nenhum valor
  - Ele apenas executa a tarefa e termina
- E o static?
  - Por hora, apenas aceitemos...

```
static void areaCasa() {
  float lateral = 11;
  float cquarto = 7;
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos

- O que significa o double?

```
static double areaPiscina() {
  double raio = 2;

  return(Math.PI *
     Math.pow(raio,2));
}
```

Métodos

- O que significa o double?
  - Que o método irá retornar um valor do tipo double

```
static double areaPiscina() {
  double raio = 2;

  return(Math.PI *
     Math.pow(raio,2));
}
```

Métodos

- O que significa o double?
  - Que o método irá retornar um valor do tipo double
  - Semelhante ao `pow(a,b)`

```
static double areaPiscina() {
  double raio = 2;

  return(Math.PI *
     Math.pow(raio,2));
}
```

Métodos

- O que significa o double?
  - Que o método irá retornar um valor do tipo double
  - Semelhante ao `pow(a,b)`
- E o `return`?

```
static double areaPiscina() {
  double raio = 2;

  return(Math.PI *
     Math.pow(raio,2));
}
```

Métodos

- O que significa o double?
  - Que o método irá retornar um valor do tipo double
  - Semelhante ao `pow(a,b)`

```
static double areaPiscina() {
  double raio = 2;

  return(Math.PI *
     Math.pow(raio,2));
}
```

- E o `return`?
  - É quando o valor é efetivamente retornado
  - A subrotina para aí
  - Alternativas:
    - `return(Math.PI * Math.pow(raio,2));`
    - `return Math.PI * Math.pow(raio,2);`

Métodos

- E como usamos isso no corpo do programa?

Métodos

- E como usamos isso no corpo do programa?

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();

  areap = areaPiscina();

  System.out.println("A área
         da piscina é "+areap);
}
```

Métodos

- E como usamos isso no corpo do programa?
  - Note que `areaPiscina()` retorna valor, então guardamos esse valor em `areap`

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();

  areap = areaPiscina();

  System.out.println("A área
         da piscina é "+areap);
}
```

Métodos

- E como usamos isso no corpo do programa?
  - Note que `areaPiscina()` retorna valor, então guardamos esse valor em `areap`
  - Já `areaCasa()` não retorna nada, então apenas a executamos

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();

  areap = areaPiscina();

  System.out.println("A área
        da piscina é "+areap);
}
```

Visão Geral do Código

```
class AreaCasa {
  static void areaCasa() {
    float lateral = 11;
    float cquarto = 7;
    float areaq;
    float areas;
    float areat;

    System.out.println("...");
    areas = lateral*lateral;
    System.out.println("..."+areas);
    areaq = cquarto*(lateral/2);
    System.out.println("..."+areaq);
    System.out.println("..."+areaq);
    areat = areas + 2*areaq;
    System.out.println("..."+areat);
  }
```

```
  static double areaPiscina() {
    double raio = 2;

    return(Math.PI *
                Math.pow(raio,2));
  }

  public static void main(String[] args)
  {
    double areap;

    areaCasa();

    areap = areaPiscina();
    System.out.println("..."+areap);
  }
}
```

Métodos

- Qual a utilidade de criarmos nossos próprios métodos?

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();
  areap = areaPiscina();
  System.out.println("A área
       da piscina é "+areap);
}
```

Métodos

- Qual a utilidade de criarmos nossos próprios métodos?
- Clareza: ao olharmos o corpo do programa, vemos claramente o que é feito, sem nos preocuparmos com detalhes
  - O método top-down fica claro

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();
  areap = areaPiscina();
  System.out.println("A área
       da piscina é "+areap);
}
```

Métodos

- Qual a utilidade de criarmos nossos próprios métodos?
- Clareza: ao olharmos o corpo do programa, vemos claramente o que é feito, sem nos preocuparmos com detalhes
  - O método top-down fica claro

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();
  areap = areaPiscina();
  System.out.println("A área
       da piscina é "+areap);
}
```

- Portabilidade: se precisarmos, em outro programa, usar a mesma subrotina, ela já está separada

Métodos

- Nossos métodos, contudo, não são gerais:

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();
  areap = areaPiscina();
  System.out.println("A área
       da piscina é "+areap);
}
```

Métodos

- Nossos métodos, contudo, não são gerais:
  - `areaCasa()` funciona apenas para casas da dimensão de nosso projeto

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();
  areap = areaPiscina();
  System.out.println("A área
       da piscina é "+areap);
}
```

Métodos

- Nossos métodos, contudo, não são gerais:
  - `areaCasa()` funciona apenas para casas da dimensão de nosso projeto
  - `areaPiscina()` funciona apenas para piscinas redondas de raio 2

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();
  areap = areaPiscina();
  System.out.println("A área
       da piscina é "+areap);
}
```

Métodos

- Nossos métodos, contudo, não são gerais:
  - `areaCasa()` funciona apenas para casas da dimensão de nosso projeto
  - `areaPiscina()` funciona apenas para piscinas redondas de raio 2

```
public static void main(
                String[] args) {
  double areap;

  areaCasa();
  areap = areaPiscina();
  System.out.println("A área
       da piscina é "+areap);
}
```

- Como poderíamos fazer para tornar esses métodos mais gerais?

Parâmetros

- A ideia é manter o formato da casa e da piscina, mas permitir que seu tamanho varie
- Como fazê-lo?

Parâmetros

- A ideia é manter o formato da casa e da piscina, mas permitir que seu tamanho varie
- Como fazê-lo? Com parâmetros:

```
static double areaPiscina(double raio){
  return(Math.PI * Math.pow(raio,2));
}
```

Parâmetros

- A ideia é manter o formato da casa e da piscina, mas permitir que seu tamanho varie
- Como fazê-lo? Com parâmetros:

```
static double areaPiscina(double raio){
  return(Math.PI * Math.pow(raio,2));
}
```

- O método agora deve receber um valor (argumento) em seu parâmetro
  - Como o `Math.pow`

Parâmetros

- Como chamamos esse método de outras partes do programa?

Parâmetros

- Como chamamos esse método de outras partes do programa?

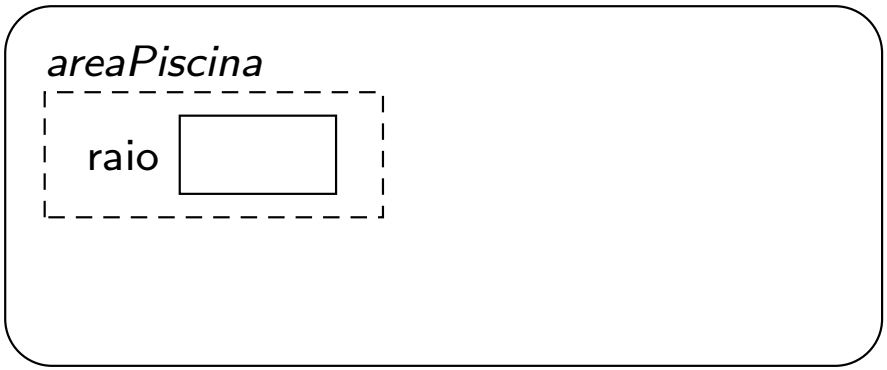
```
public static void main
            (String[] args) {
  double areap;

  areaCasa();

  areap = areaPiscina(2);
  System.out.println("A
       área da piscina é "+
                        areap);
}
```

Parâmetros

- E o que acontece ao chamarmos `areaPiscina(2)` de dentro do main?

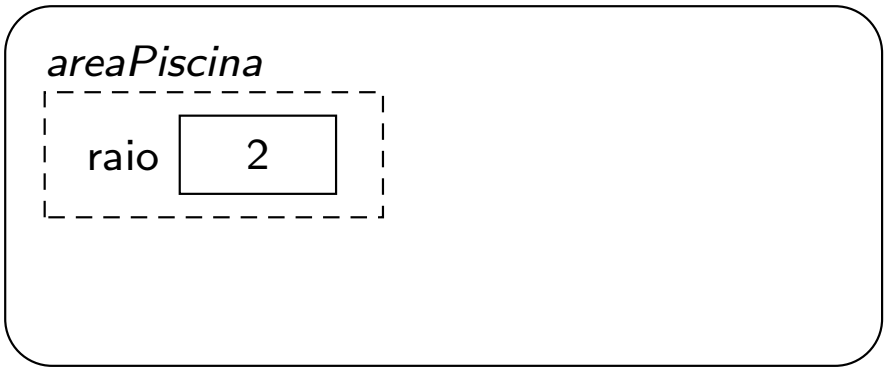
Parâmetros

- E o que acontece ao chamarmos `areaPiscina(2)` de dentro do main?
  - O S.O. irá alocar memória para todas as variáveis e parâmetros declarados dentro do método



Parâmetros

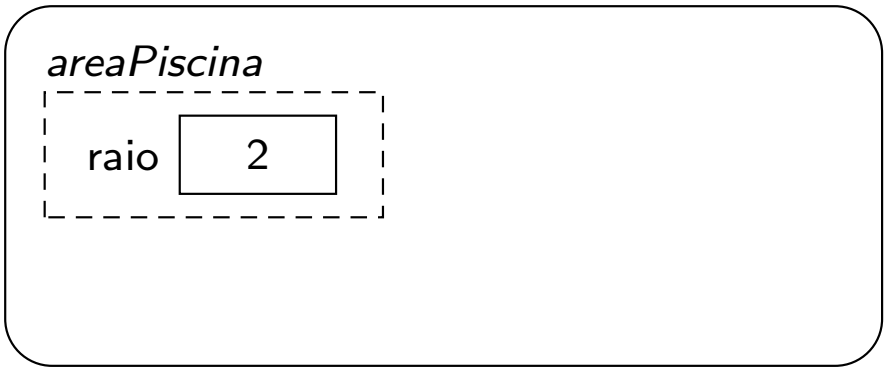
- E o que acontece ao chamarmos `areaPiscina(2)` de dentro do main?
  - O S.O. irá alocar memória para todas as variáveis e parâmetros declarados dentro do método



  - Colocando o valor passado como parâmetro lá

Parâmetros

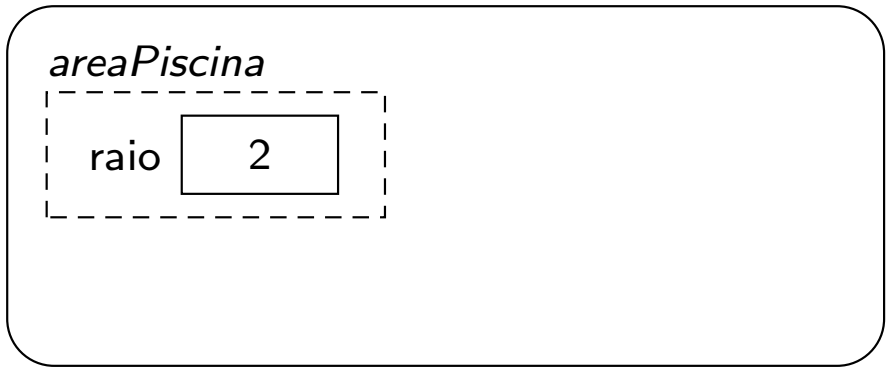
- E o que acontece ao chamarmos `areaPiscina(2)` de dentro do main?
  - O S.O. irá alocar memória para todas as variáveis e parâmetros declarados dentro do método



  - Colocando o valor passado como parâmetro lá
- Ao ato de passar um valor externo para dentro de um procedimento, via parâmetro, chamamos de passagem por valor

Parâmetros

- E o que acontece ao chamarmos `areaPiscina(2)` de dentro do main?
  - O S.O. irá alocar memória para todas as variáveis e parâmetros declarados dentro do método
  - Colocando o valor passado como parâmetro lá

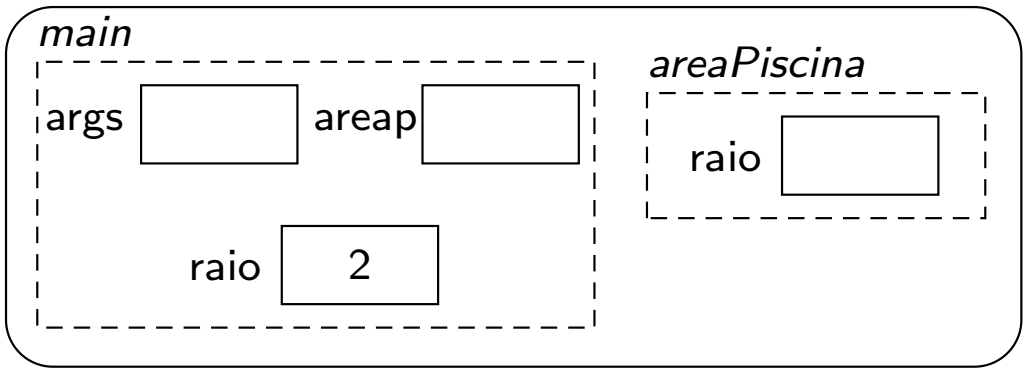


- Ao ato de passar um valor externo para dentro de um procedimento, via parâmetro, chamamos de passagem por valor
  - Nesse caso, o valor externo é copiado para a região de memória correspondente ao parâmetro

Parâmetros

- O que acontece se tivermos algo assim?

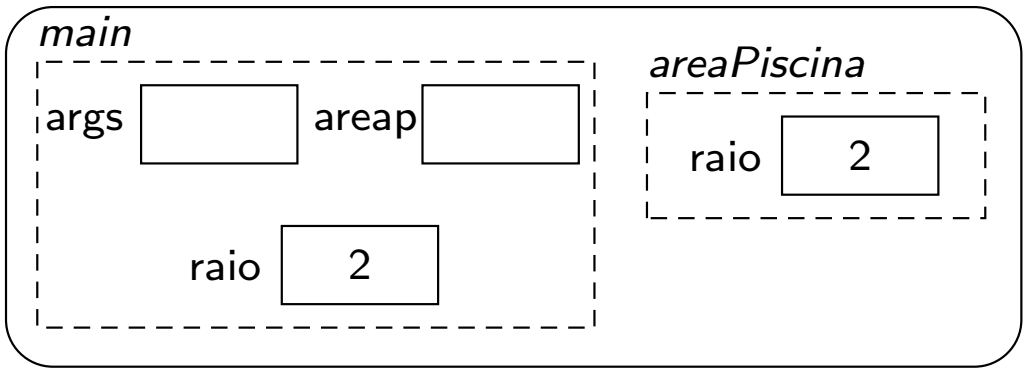
```
public static void main(String[]
                                args) {
    double areap;
    double raio = 2;

    areaCasa();

    areap = areaPiscina(raio);
    System.out.println("A área da
            piscina é "+areap);
}
```



Parâmetros

- O que acontece se tivermos algo assim?
  - O valor de `raio`, em `main`, é copiado para dentro da variável `raio` em `areaPiscina`

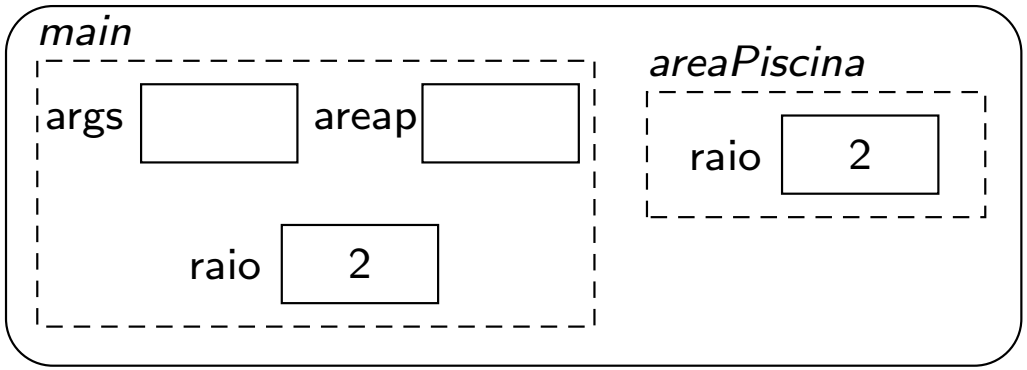
```
public static void main(String[]
                                args) {
    double areap;
    double raio = 2;

    areaCasa();

    areap = areaPiscina(raio);
    System.out.println("A área da
              piscina é "+areap);
}
```



Parâmetros

- O que acontece se tivermos algo assim?
  - O valor de `raio`, em `main`, é copiado para dentro da variável `raio` em `areaPiscina`
  - São duas regiões de memória diferentes

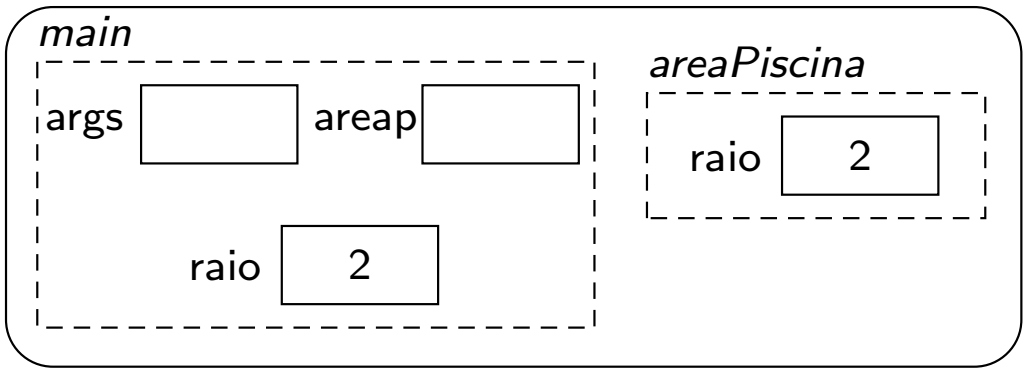
```
public static void main(String[]
                                args) {
    double areap;
    double raio = 2;

    areaCasa();

    areap = areaPiscina(raio);
    System.out.println("A área da
              piscina é "+areap);
}
```



Parâmetros

- O que acontece se tivermos algo assim?
  - O valor de `raio`, em `main`, é copiado para dentro da variável `raio` em `areaPiscina`
  - São duas regiões de memória diferentes
  - Sim... main é um método também

```
public static void main(String[]
                                args) {
    double areap;
    double raio = 2;

    areaCasa();

    areap = areaPiscina(raio);
    System.out.println("A área da
              piscina é "+areap);
}
```



Parâmetros

Incluindo parâmetros em *areaCasa()*:

```
class AreaCasa {
  static void areaCasa(float
            lateral,float cquarto){
    float areaq;
    float areas;
    float areat;

    System.out.println("...");
    areas = lateral*lateral;
    System.out.println("..."+areas);
    areaq = cquarto*(lateral/2);
    System.out.println("..."+areaq);
    System.out.println("..."+areaq);
    areat = areas + 2*areaq;
    System.out.println("..."+areat);
  }
  static double areaPiscina(double
                              raio){
    return Math.PI*
                Math.pow(raio,2);
  }

  public static void main(String[]
                               args){
    double areap;

    areaCasa(11,7);
    areap = areaPiscina(2);
    System.out.println("..."+
                              areap);
  }
}
```

Métodos e Memória

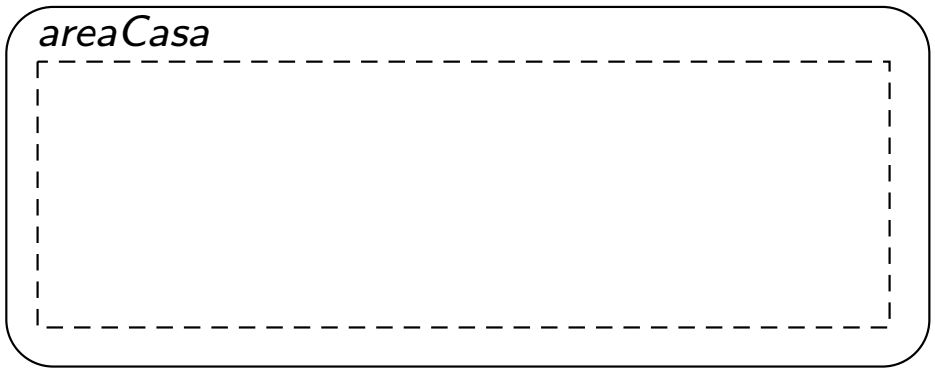
- Como fica o método areaCasa na memória?

```
static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos e Memória

- Como fica o método `areaCasa` na memória?
- Ao ser chamado (ou invocado) em `main`, será separada uma região na memória para esse método

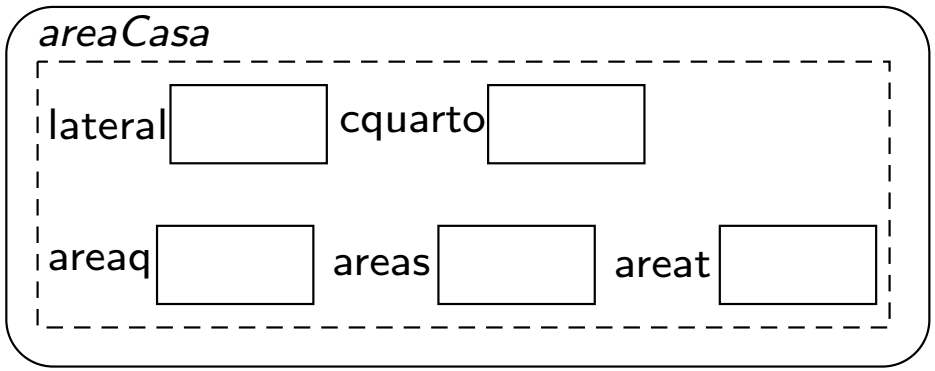
```
static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```



Métodos e Memória

- Como fica o método `areaCasa` na memória?
- Ao ser chamado (ou invocado) em `main`, será separada uma região na memória para esse método
- Essa região conterá todas suas variáveis internas (locais), e todos seus parâmetros

```
static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

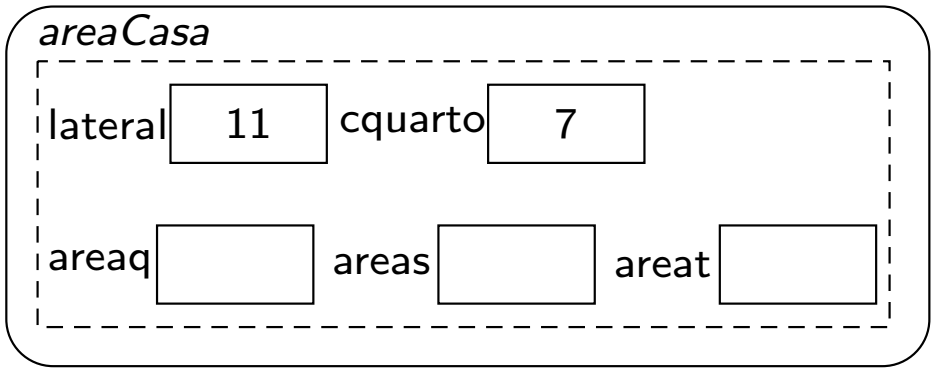


Métodos e Memória

```
public static void main(
                String[] args) {
  ...

  areaCasa(11,7);

  ...
}
```

```
static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

- Os valores de entrada são então copiados para dentro dos parâmetros



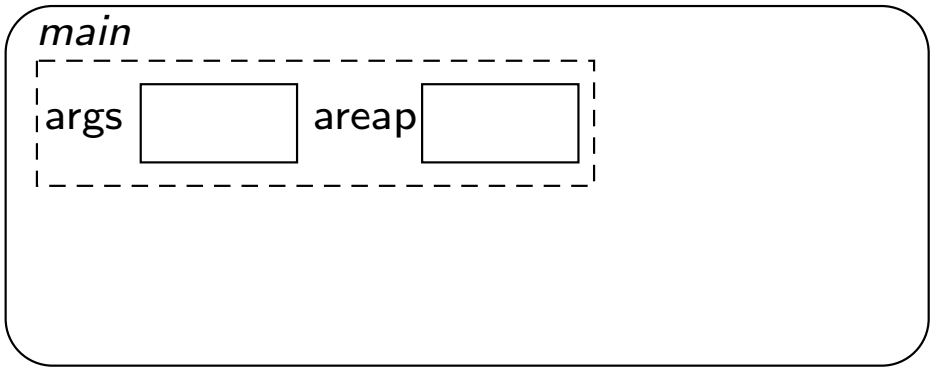
Métodos e Memória

- Considerando o programa como um todo, como agirá na memória?

```
public static void main(String[] args){
  double areap;

  areaCasa(11,7);
  areap = areaPiscina(2);
  System.out.println("A área da
                piscina é "+areap);
}
```

Métodos e Memória

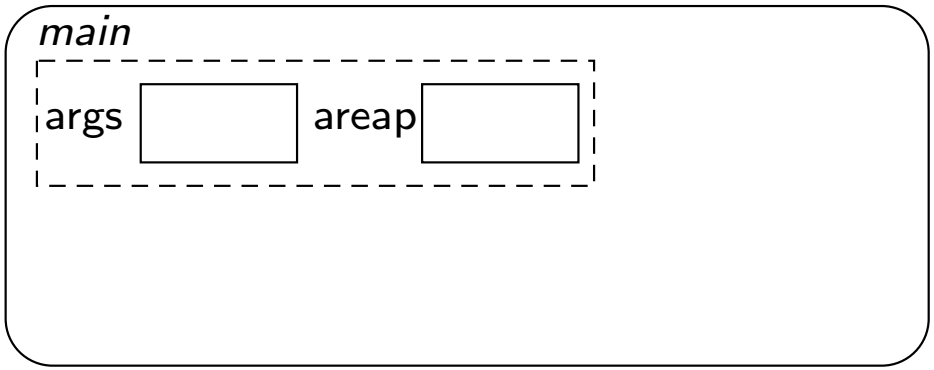
- Considerando o programa como um todo, como agirá na memória?
  - Ao iniciar *main*, será alocado espaço para suas variáveis e parâmetros




```
public static void main(String[] args){
  double areap;

  areaCasa(11,7);
  areap = areaPiscina(2);
  System.out.println("A área da
                piscina é "+areap);
}
```

Métodos e Memória

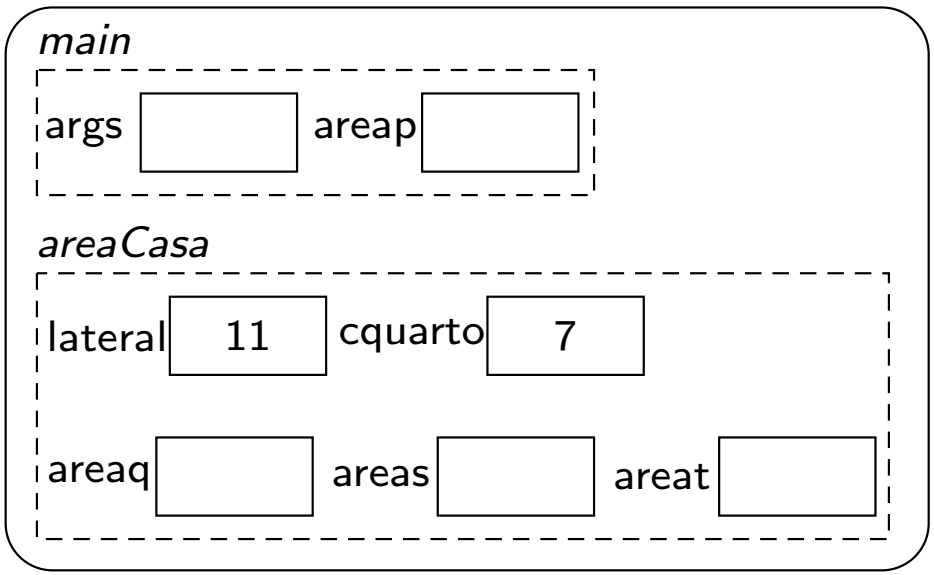
- Considerando o programa como um todo, como agirá na memória?
  - Ao iniciar *main*, será alocado espaço para suas variáveis e parâmetros
  - Então *areaCasa(11,7)* é executada, e o mesmo processo ocorre




```
public static void main(String[] args){
  double areap;

  areaCasa(11,7);
  areap = areaPiscina(2);
  System.out.println("A área da
                piscina é "+areap);
}
```

Métodos e Memória

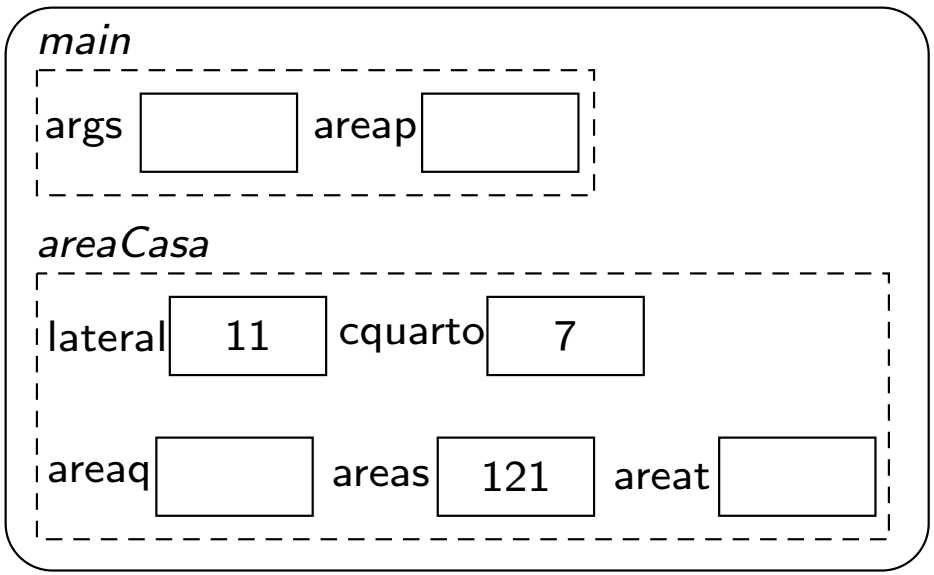
- Aloca-se espaço, copiando-se os valores aos parâmetros:




```
public static void main(String[] args){
  ...
  areaCasa(11,7);
  ...
}

static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos e Memória

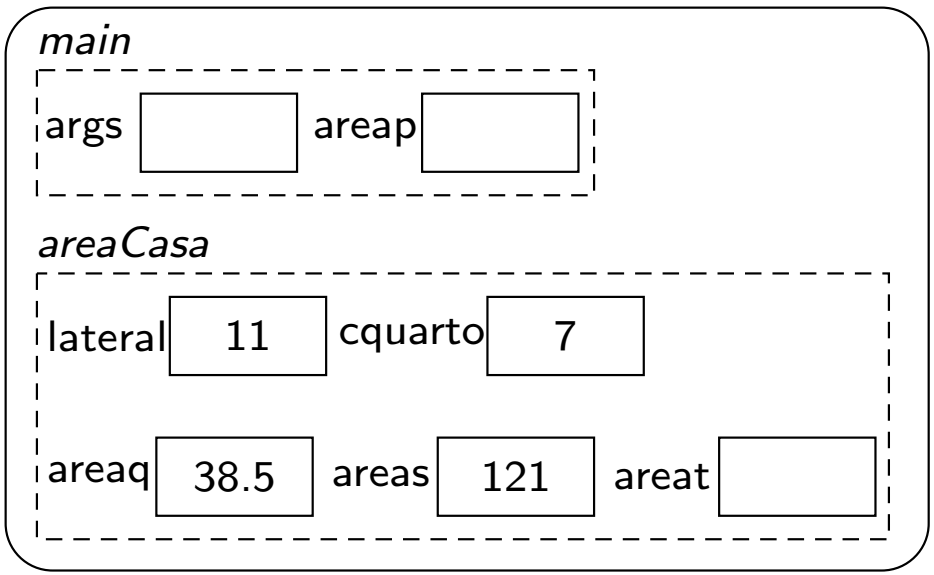
- A cada atribuição, a memória correspondente é atualizada




```
public static void main(String[] args){
  ...
  areaCasa(11,7);
  ...
}

static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos e Memória

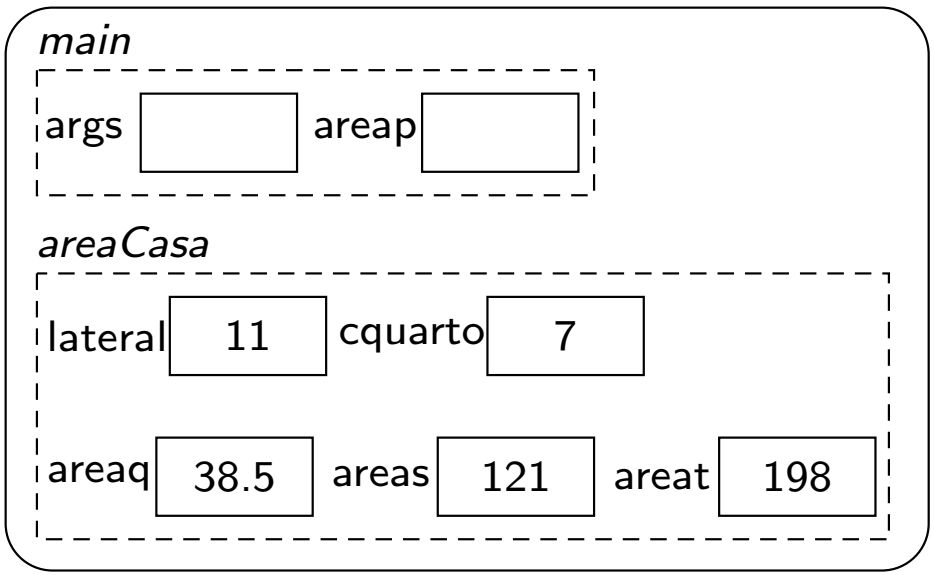
- A cada atribuição, a memória correspondente é atualizada




```
public static void main(String[] args){
  ...
  areaCasa(11,7);
  ...
}

static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos e Memória

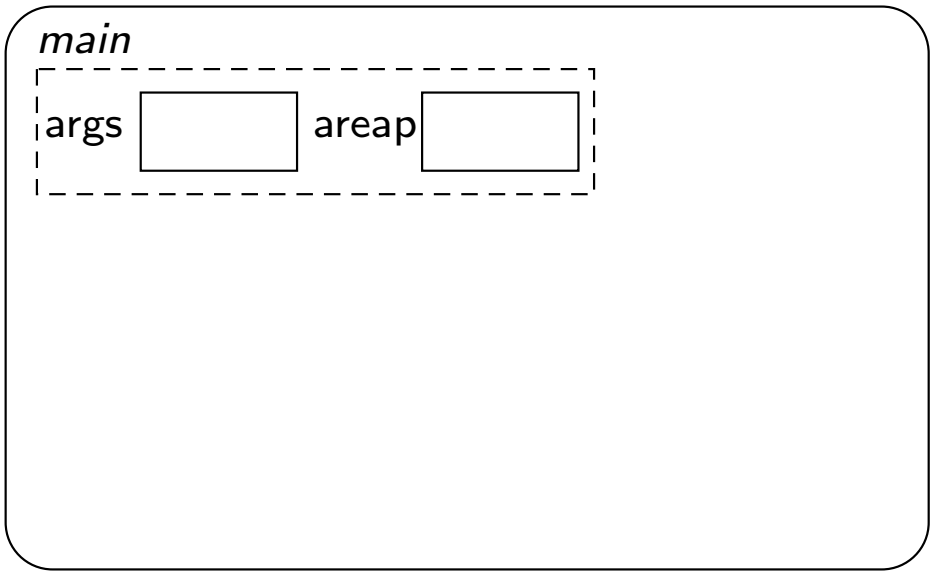
- A cada atribuição, a memória correspondente é atualizada




```
public static void main(String[] args){
  ...
  areaCasa(11,7);
  ...
}

static void areaCasa(float lateral,
                     float cquarto) {
  float areaq;
  float areas;
  float areat;

  System.out.println("...");
  areas = lateral*lateral;
  System.out.println("..."+areas);
  areaq = cquarto*(lateral/2);
  System.out.println("..."+areaq);
  System.out.println("..."+areaq);
  areat = areas + 2*areaq;
  System.out.println("..."+areat);
}
```

Métodos e Memória

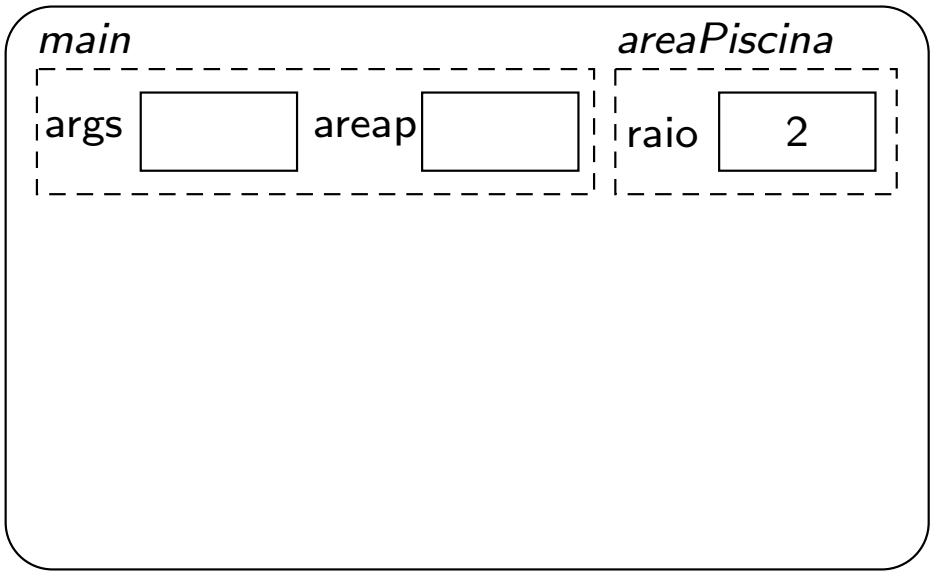
- Ao terminar areaCasa, sua memória é limpa, e areaPiscina é rodada:




```
static double areaPiscina(double raio)
{
    return Math.PI * Math.pow(raio,2);
}

public static void main(String[] args)
{
    double areap;

    areaCasa(11,7);

    areap = areaPiscina(2);
    System.out.println("A área da
                    piscina é "+areap);
}
```

Métodos e Memória

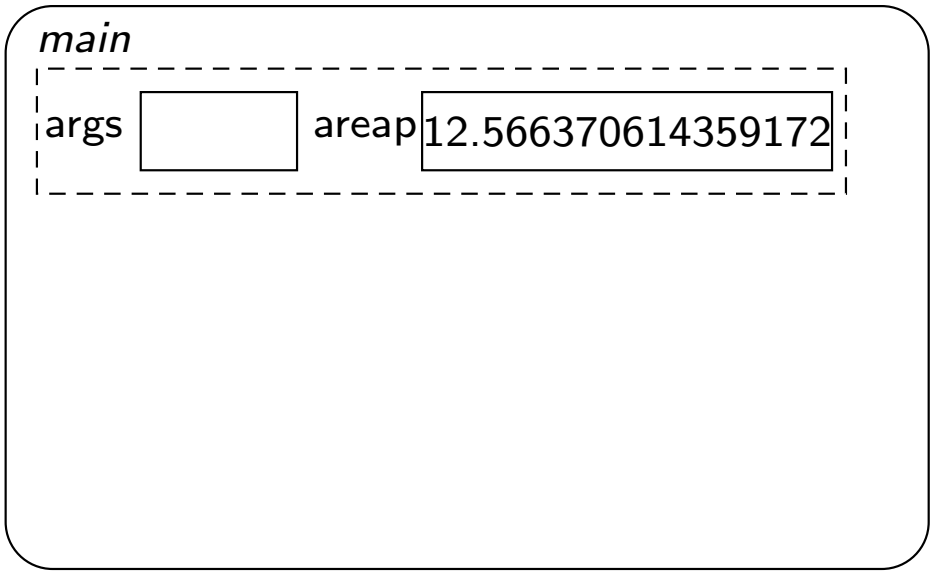
- Ao terminar areaCasa, sua memória é limpa, e areaPiscina é rodada:




```
static double areaPiscina(double raio)
{
    return Math.PI * Math.pow(raio,2);
}

public static void main(String[] args)
{
    double areap;

    areaCasa(11,7);

    areap = areaPiscina(2);
    System.out.println("A área da
                    piscina é "+areap);
}
```

Métodos e Memória

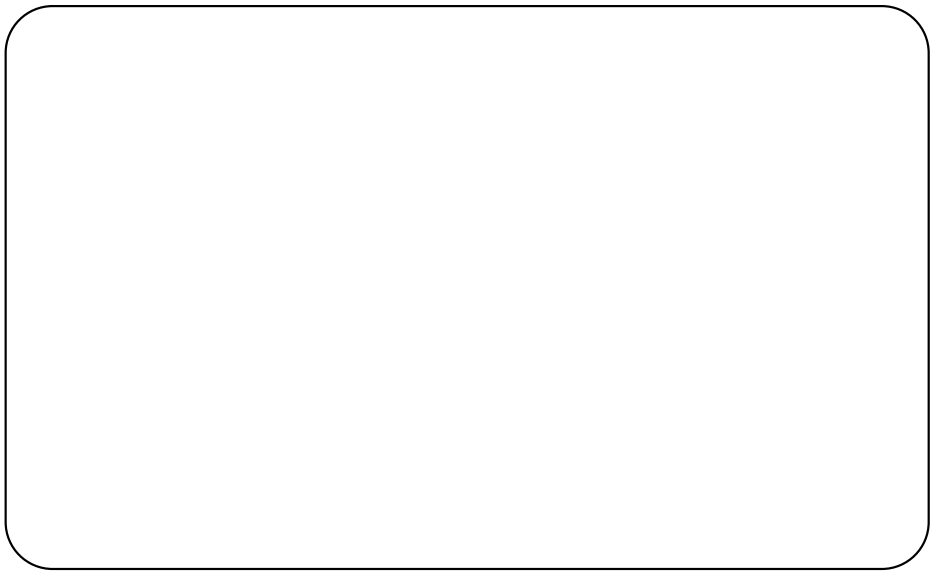
- Ao terminar `areaPiscina`, sua memória é limpa, e o resultado é armazenado em `areap`:




```
public static void main(
                        String[] args){
    double areap;

    areaCasa(11,7);

    areap = areaPiscina(2);
    System.out.println("A área
            da piscina é "+areap);
}
```

Métodos e Memória

- Ao terminar `areaPiscina`, sua memória é limpa, e o resultado é armazenado em `areap`:



```
public static void main(
                        String[] args){
    double areap;

    areaCasa(11,7);

    areap = areaPiscina(2);
    System.out.println("A área
            da piscina é "+areap);
}
```

Finalmente, quando *main* terminar, também será removida

Métodos e Memória

- Repare que toda vez que um método termina ele libera a memória que ocupava
- Então, qual a utilidade de criarmos nossos próprios métodos além de clareza e portabilidade?
  - Melhor uso da memória: as variáveis relevantes ao sub-problema (sub-rotina) ocupam a memória apenas durante a solução desse sub-problema

Videoaula

https://www.youtube.com/watch?v=tZXF4Ar_w58&t=1s
https://www.youtube.com/watch?v=rK3LpUSijGM&t=3s
e
https://www.youtube.com/watch?v=fOintAunVVg&t=11s