Aula 13 – Classes em Java

Norton Trevisan Roman

3 de maio de 2013

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes
  - Problema: o preço do m^2 é diferente entre as regiões

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes
  - Problema: o preço do m^2 é diferente entre as regiões
  - Como fazer?

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes
  - Problema: o preço do m^2 é diferente entre as regiões
  - Como fazer?

```
class Projeto {
    ...
    public static void main(String[] args) {
        double valorM2_ant = AreaCasa.valorM2;

        // preço da casa 1 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(15,10) ) );

        // novo valor do m2
        AreaCasa.valorM2 = 1270;

        // preço da casa 2 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(18,8) ) );

        // restauro o valor anterior
        AreaCasa.valorM2 = valorM2_ant;
    }
}
```

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes
  - Problema: o preço do m^2 é diferente entre as regiões
  - Como fazer?
- Trabalhoso e Perigoso!

```
class Projeto {
    ...
    public static void main(String[] args) {
        double valorM2_ant = AreaCasa.valorM2;

        // preço da casa 1 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(15,10) ) );

        // novo valor do m2
        AreaCasa.valorM2 = 1270;

        // preço da casa 2 (sem piscina)
        System.out.println( AreaCasa.valor(
                          AreaCasa.area(18,8) ) );

        // restauro o valor anterior
        AreaCasa.valorM2 = valorM2_ant;
    }
}
```

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes
  - Problema: o preço do m^2 é diferente entre as regiões
  - Como fazer?
- Trabalhoso e Perigoso!
  - À medida em que o programa cresce, podemos esquecer de restaurar o valor original

```
class Projeto {
    ...
    public static void main(String[] args) {
        double valorM2_ant = AreaCasa.valorM2;

        // preço da casa 1 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(15,10) ) );

        // novo valor do m2
        AreaCasa.valorM2 = 1270;

        // preço da casa 2 (sem piscina)
        System.out.println( AreaCasa.valor(
                          AreaCasa.area(18,8) ) );

        // restauro o valor anterior
        AreaCasa.valorM2 = valorM2_ant;
    }
}
```

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes
  - Problema: o preço do m^2 é diferente entre as regiões
  - Como fazer?
- Trabalhoso e Perigoso!
  - À medida em que o programa cresce, podemos esquecer de restaurar o valor original

```
class Projeto {
    ...
    public static void main(String[] args) {
        double valorM2_ant = AreaCasa.valorM2;

        // preço da casa 1 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(15,10) ) );

        // novo valor do m2
        AreaCasa.valorM2 = 1270;

        // preço da casa 2 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(18,8) ) );

        // restauro o valor anterior
        AreaCasa.valorM2 = valorM2_ant;
    }
}
```

- Podemos então mudar variáveis em outras classes?

Classes em Java

- Queremos agora calcular a área de 2 casas em regiões geográficas diferentes
  - Problema: o preço do m^2 é diferente entre as regiões
  - Como fazer?
- Trabalhoso e Perigoso!
  - À medida em que o programa cresce, podemos esquecer de restaurar o valor original

```
class Projeto {
    ...
    public static void main(String[] args) {
        double valorM2_ant = AreaCasa.valorM2;

        // preço da casa 1 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(15,10) ) );

        // novo valor do m2
        AreaCasa.valorM2 = 1270;

        // preço da casa 2 (sem piscina)
        System.out.println( AreaCasa.valor(
                        AreaCasa.area(18,8) ) );

        // restauro o valor anterior
        AreaCasa.valorM2 = valorM2_ant;
    }
}
```

- Podemos então mudar variáveis em outras classes?
  - Nem sempre, mas nesse caso, sim.

Classes em Java

- Como é possível mudarmos o valor em *AreaCasa*?

Classes em Java

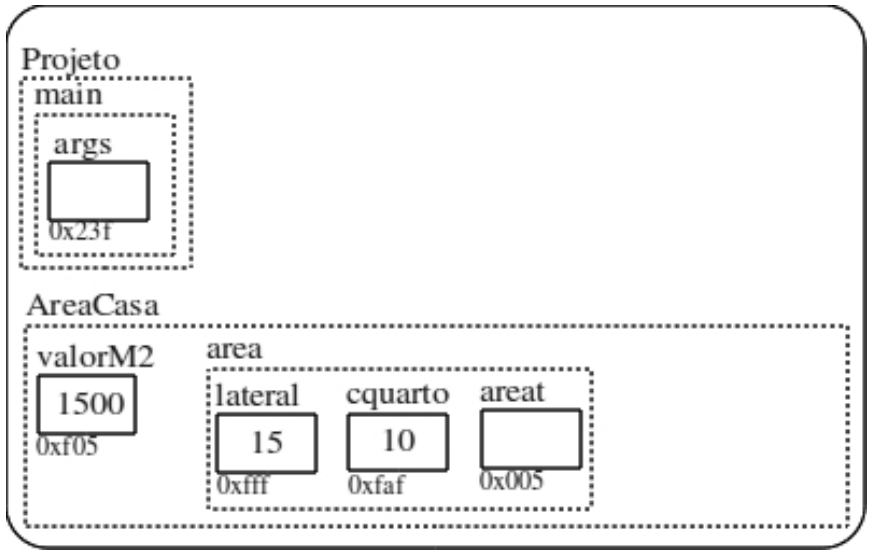
- Como é possível mudarmos o valor em *AreaCasa*?
- Considere o código de *Projeto*:

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

    // preço da casa 1 (sem piscina)
    System.out.println( AreaCasa.valor(
                  AreaCasa.area(15,10) ) );

    AreaCasa.valorM2 = 1270;

    ...
}
```

Classes em Java

- Como é possível mudarmos o valor em *AreaCasa*?
- Considere o código de *Projeto*:
  - Ao chamarmos *AreaCasa* pela primeira vez, reservamos memória para seus atributos e para o método chamado, atualizando os parâmetros

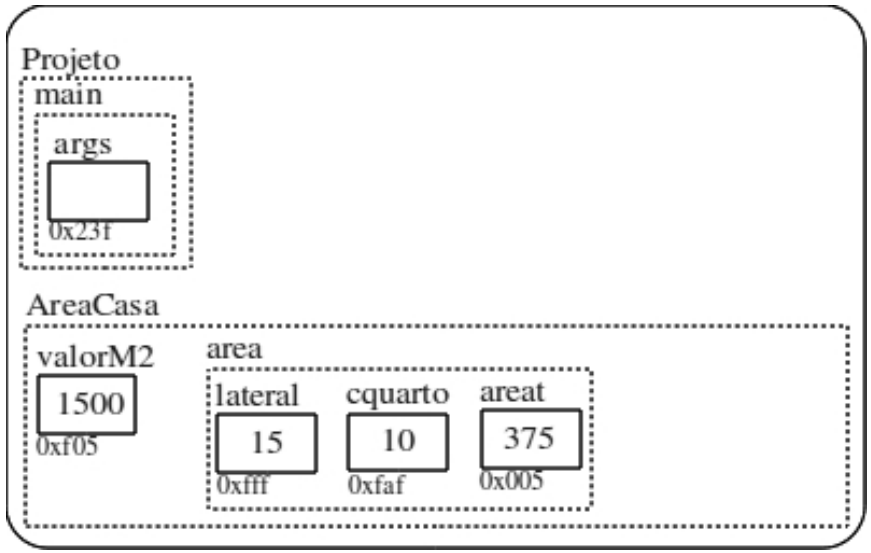
```
public static void main(String[] args) {
    ...

    // preço da casa 1 (sem piscina)
    System.out.println( AreaCasa.valor(
                AreaCasa.area(15,10) ) );

    AreaCasa.valorM2 = 1270;

    ...
}
```



Classes em Java

- Como é possível mudarmos o valor em *AreaCasa*?
- Considere o código de *Projeto*:
  - Ao chamarmos *AreaCasa* pela primeira vez, reservamos memória para seus atributos e para o método chamado, atualizando os parâmetros

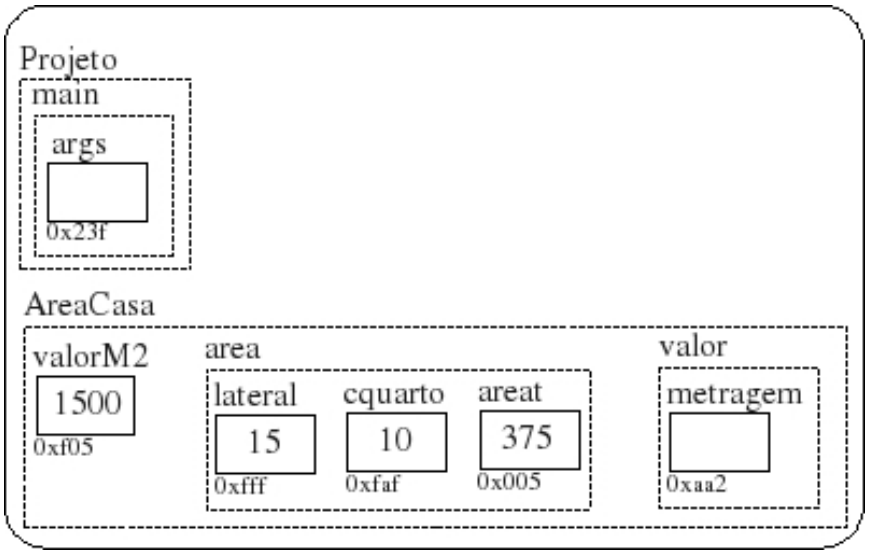
```
public static void main(String[] args) {
    ...

    // preço da casa 1 (sem piscina)
    System.out.println( AreaCasa.valor(
                AreaCasa.area(15,10) ) );

    AreaCasa.valorM2 = 1270;

    ...
}
```



Classes em Java

- Como é possível mudarmos o valor em *AreaCasa*?
- Considere o código de *Projeto*:
  - Ao chamarmos *AreaCasa* pela primeira vez, reservamos memória para seus atributos e para o método chamado, atualizando os parâmetros
  - Ao chamarmos pela segunda vez, alocamos apenas para o método

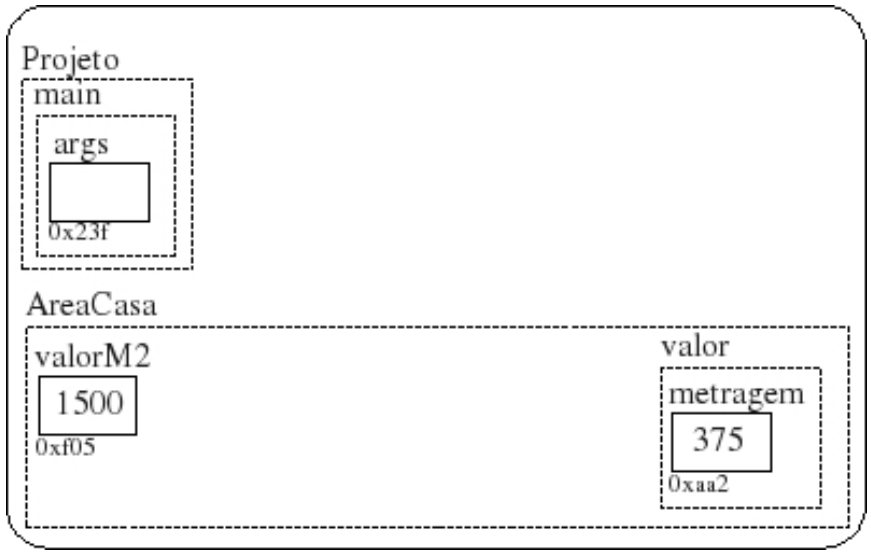
```
public static void main(String[] args) {
    ...

    // preço da casa 1 (sem piscina)
    System.out.println( AreaCasa.valor(
                AreaCasa.area(15,10) ) );

    AreaCasa.valorM2 = 1270;

    ...
}
```



Classes em Java

- Como é possível mudarmos o valor em *AreaCasa*?
- Considere o código de *Projeto*:
  - Ao chamarmos *AreaCasa* pela primeira vez, reservamos memória para seus atributos e para o método chamado, atualizando os parâmetros
  - Ao chamarmos pela segunda vez, alocamos apenas para o método
    - ★ Atualizamos então os parâmetros

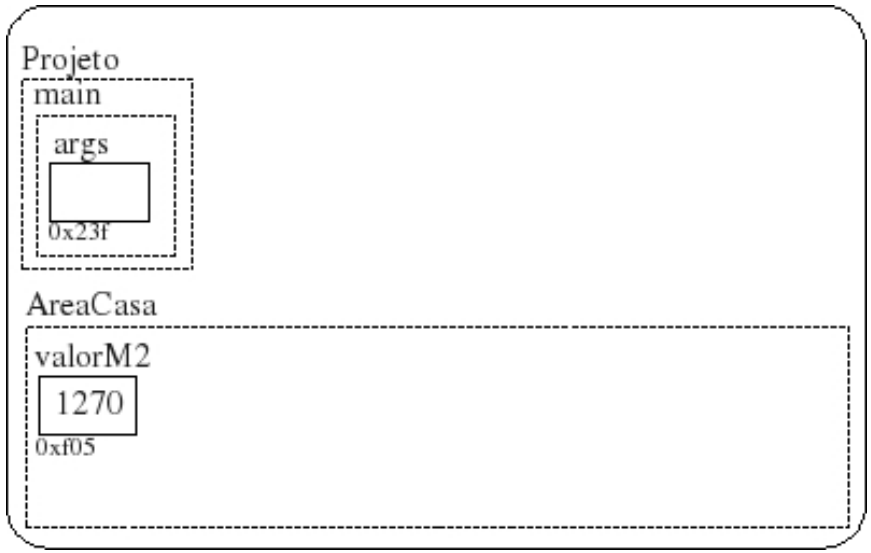
```
public static void main(String[] args) {
    ...

    // preço da casa 1 (sem piscina)
    System.out.println( AreaCasa.valor(
                AreaCasa.area(15,10) ) );

    AreaCasa.valorM2 = 1270;

    ...
}
```



Classes em Java

- Como é possível mudarmos o valor em *AreaCasa*?
- Considere o código de *Projeto*:
  - Ao chamarmos *AreaCasa* pela primeira vez, reservamos memória para seus atributos e para o método chamado, atualizando os parâmetros
  - Ao chamarmos pela segunda vez, alocamos apenas para o método
    - ★ Atualizamos então os parâmetros
  - Ao atualizarmos o campo *valorM2*, simplesmente colocamos valor naquela região de memória

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

    // preço da casa 1 (sem piscina)
    System.out.println( AreaCasa.valor(
                AreaCasa.area(15,10) ) );

    AreaCasa.valorM2 = 1270;

    ...
}
```



Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?
  - Criando objetos!

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?
  - Criando objetos!
    - ★ Uma entidade que represente uma única casa na memória

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?
  - Criando objetos!
    - ★ Uma entidade que represente uma única casa na memória
  - Classe $\rightarrow$ especificação do código

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?
  - Criando objetos!
    - ★ Uma entidade que represente uma única casa na memória
  - Classe $\rightarrow$ especificação do código
    - ★ Papel dos atores

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?
  - Criando objetos!
    - ★ Uma entidade que represente uma única casa na memória
  - Classe $\rightarrow$ especificação do código
    - ★ Papel dos atores
  - Objeto $\rightarrow$ entidade na memória que usa esse código

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?
  - Criando objetos!
    - ★ Uma entidade que represente uma única casa na memória
  - Classe $\rightarrow$ especificação do código
    - ★ Papel dos atores
  - Objeto $\rightarrow$ entidade na memória que usa esse código
    - ★ Cada ator específico

Objetos em Java

- Deixar o programa assim, além de trabalhoso, é potencialmente perigoso, à medida em que o programa cresce
- Qual a alternativa?
  - Fazer com que o valor do m^2 seja algo intrínseco a cada casa específica.
  - Cada casa teria seu próprio valor de m^2
- Como?
  - Criando objetos!
    - ★ Uma entidade que represente uma única casa na memória
  - Classe $\rightarrow$ especificação do código
    - ★ Papel dos atores
  - Objeto $\rightarrow$ entidade na memória que usa esse código
    - ★ Cada ator específico
    - ★ Diz-se que objetos instanciam as (são instâncias das) classes

Classe × Objeto

- Atributos e Métodos da classe:
- Atributos e Métodos do objeto:

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
- Atributos e Métodos do objeto:

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
- Atributos e Métodos do objeto:
  - São visíveis apenas pelo objeto da classe

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
  - Também são visiveis por outras classes
- Atributos e Métodos do objeto:
  - São visíveis apenas pelo objeto da classe

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
  - Também são visiveis por outras classes
- Atributos e Métodos do objeto:
  - São visíveis apenas pelo objeto da classe
  - Não são visiveis por outras classes (a não ser via um objeto)

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
  - Também são visiveis por outras classes
  - Utilizam a palavra reservada *static*
- Atributos e Métodos do objeto:
  - São visíveis apenas pelo objeto da classe
  - Não são visiveis por outras classes (a não ser via um objeto)

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
  - Também são visiveis por outras classes
  - Utilizam a palavra reservada *static*

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}
```

- Atributos e Métodos do objeto:
  - São visíveis apenas pelo objeto da classe
  - Não são visiveis por outras classes (a não ser via um objeto)

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
  - Também são visiveis por outras classes
  - Utilizam a palavra reservada *static*

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}
```

- Atributos e Métodos do objeto:
  - São visíveis apenas pelo objeto da classe
  - Não são visiveis por outras classes (a não ser via um objeto)
  - Não podem ser *static*

Classe × Objeto

- Atributos e Métodos da classe:
  - São visíveis dentro de toda a classe
  - Também são visiveis por outras classes
  - Utilizam a palavra reservada *static*

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}
```

- Atributos e Métodos do objeto:
  - São visíveis apenas pelo objeto da classe
  - Não são visiveis por outras classes (a não ser via um objeto)
  - Não podem ser *static*

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) {
        ...
    }

    double valor(double area) {
        ...
    }
}
```

Objetos

- Atributos e Métodos da classe:

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}
```

- Atributos e Métodos do objeto:

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) {
        ...
    }

    double valor(double area) {
        ...
    }
}
```

Objetos

- Atributos e Métodos da classe:
  - Acessados com *NomeDaClasse.metodo*

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}
```

- Atributos e Métodos do objeto:

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) {
        ...
    }

    double valor(double area) {
        ...
    }
}
```

Objetos

- Atributos e Métodos da classe:
  - Acessados com *NomeDaClasse.metodo*

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

- Atributos e Métodos do objeto:

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) {
        ...
    }

    double valor(double area) {
        ...
    }
}
```

Objetos

- Atributos e Métodos da classe:
  - Acessados com *NomeDaClasse.metodo*

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

- Atributos e Métodos do objeto:
  - Deve-se criar um objeto

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) {
        ...
    }

    double valor(double area) {
        ...
    }
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                       casa1.area(15,10) ) );
    }
}
```

Objetos

- Atributos e Métodos da classe:
  - Acessados com *NomeDaClasse.metodo*

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto) {
        ...
    }

    static double valor(double area) {
        ...
    }
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

- Atributos e Métodos do objeto:
  - Deve-se criar um objeto
  - Então acessá-lo com *nome_do_objeto.método*

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) {
        ...
    }

    double valor(double area) {
        ...
    }
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                       casa1.area(15,10) ) );
    }
}
```

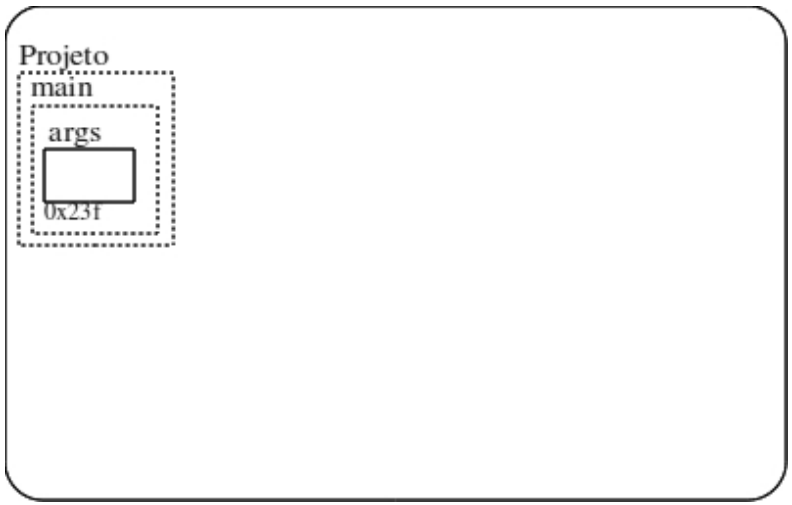
Comportamento na memória

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
    }
}
```

Comportamento na memória

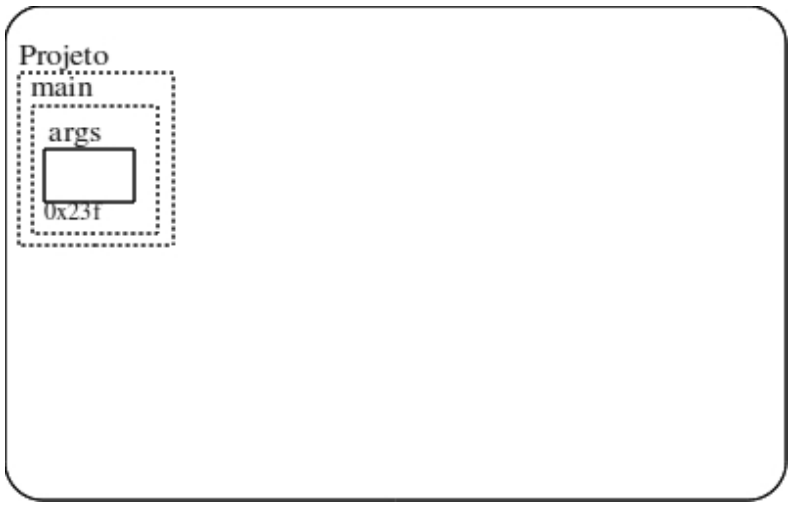
```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```



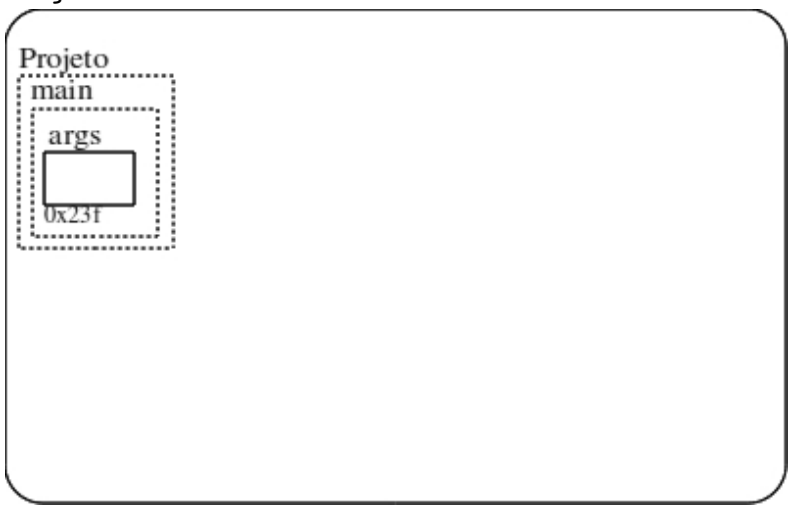

```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
    }
}
```

Comportamento na memória

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

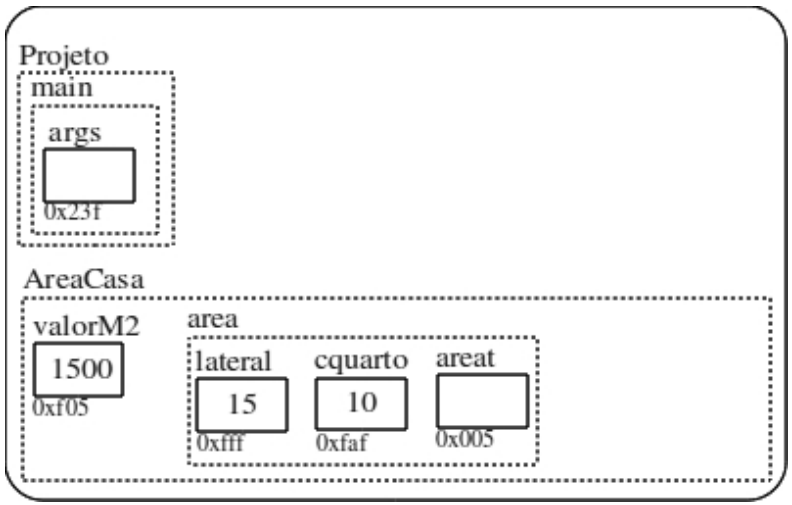



```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
    }
}
```

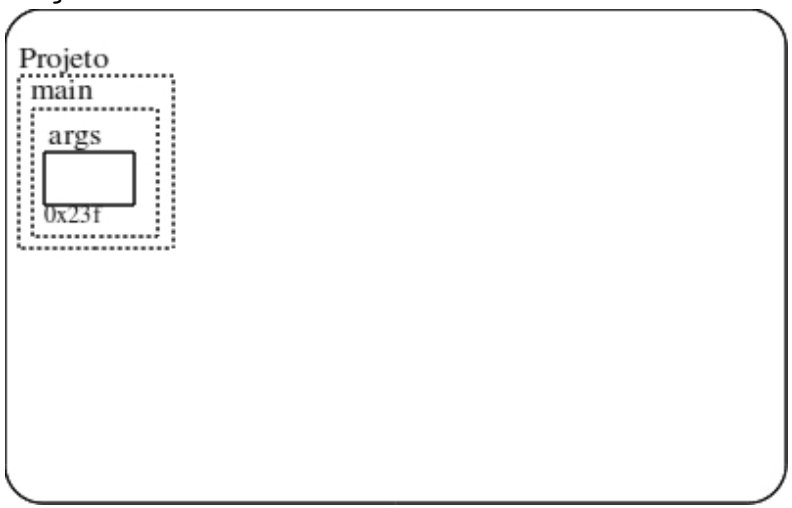


Comportamento na memória

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

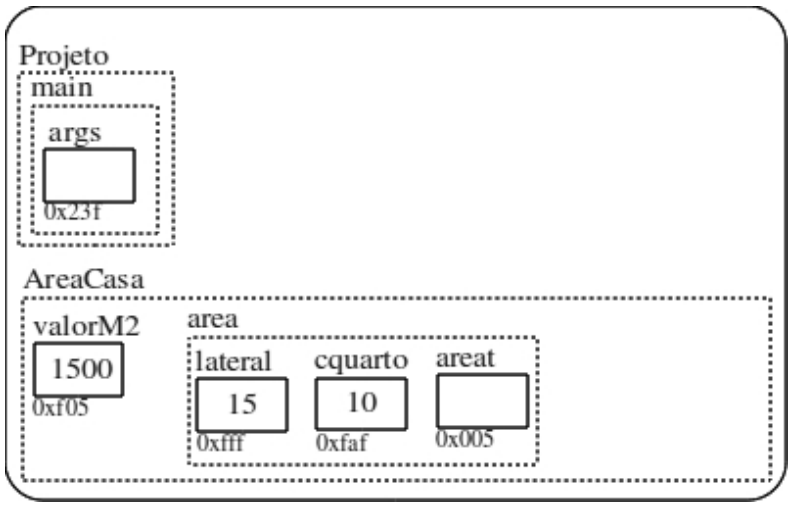



```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
    }
}
```

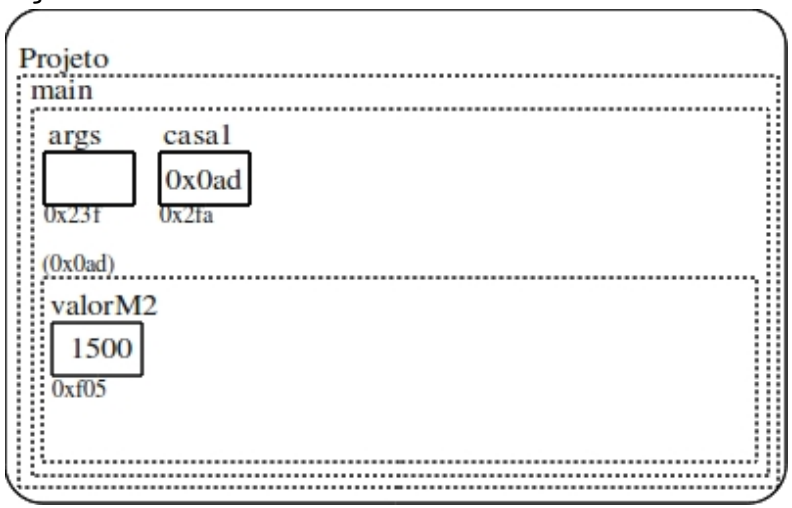


Comportamento na memória

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

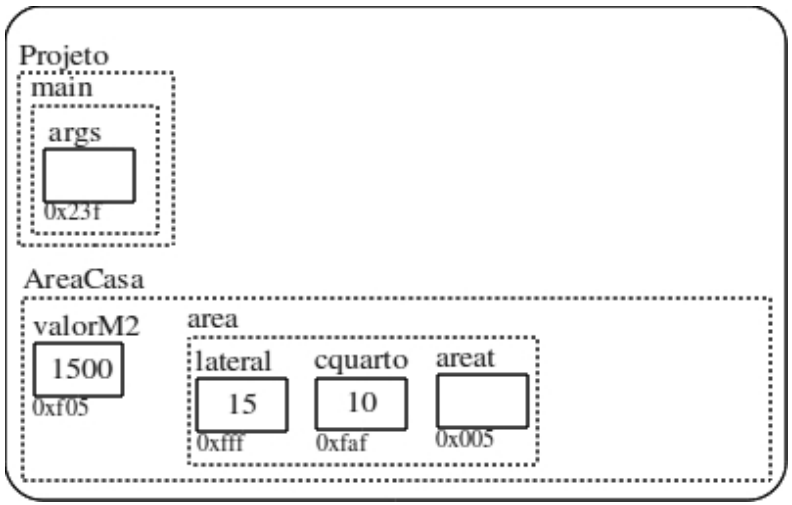



```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
    }
}
```

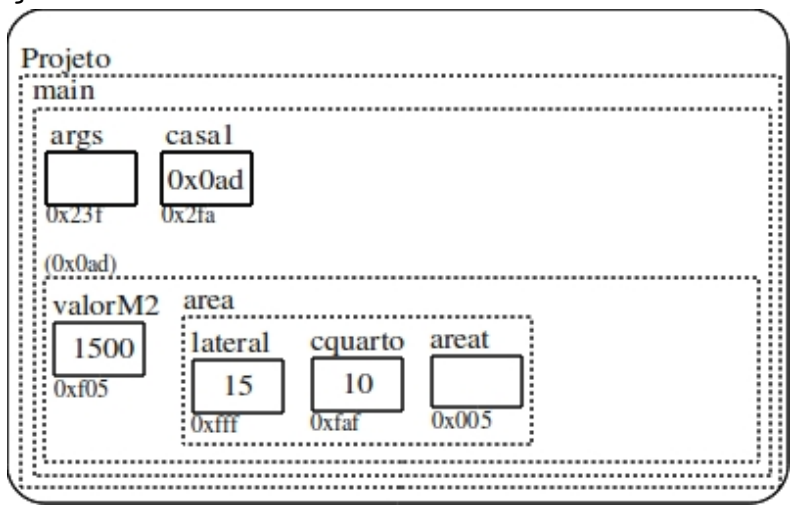


Comportamento na memória

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

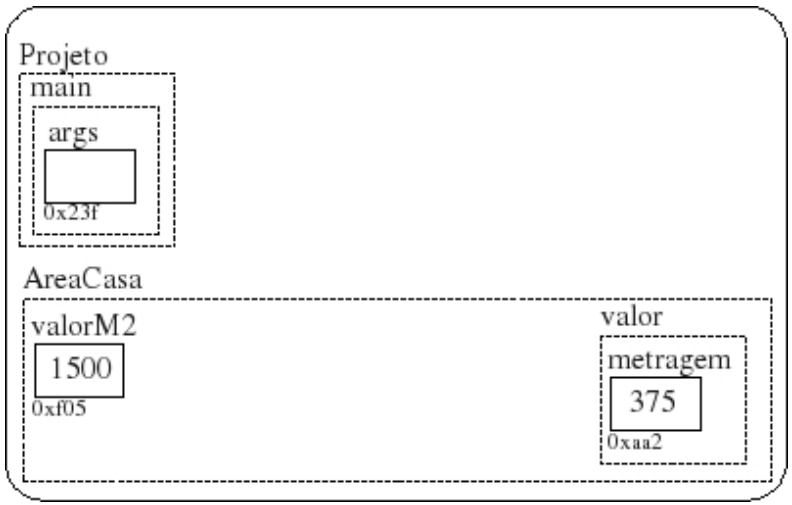



```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                       casa1.area(15,10) ) );
    }
}
```

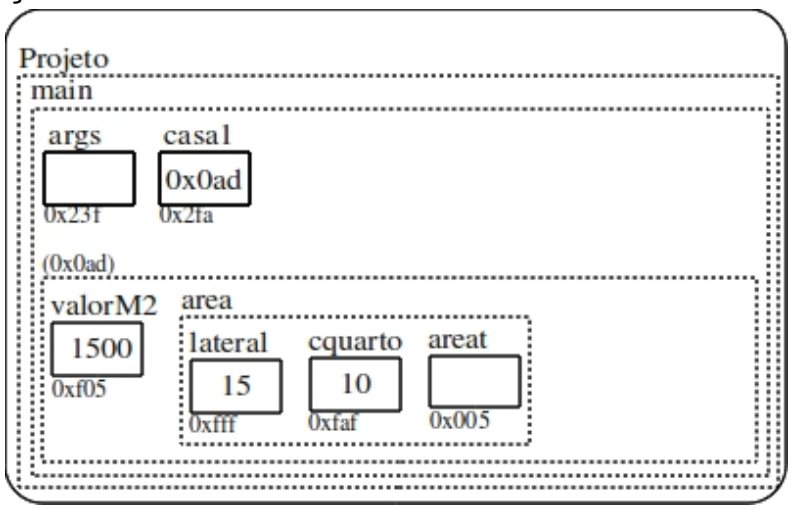


Comportamento na memória

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

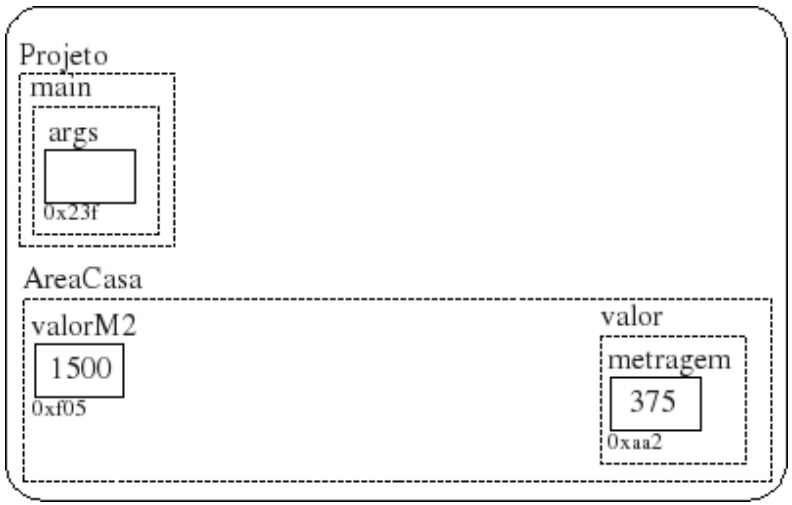



```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
    }
}
```

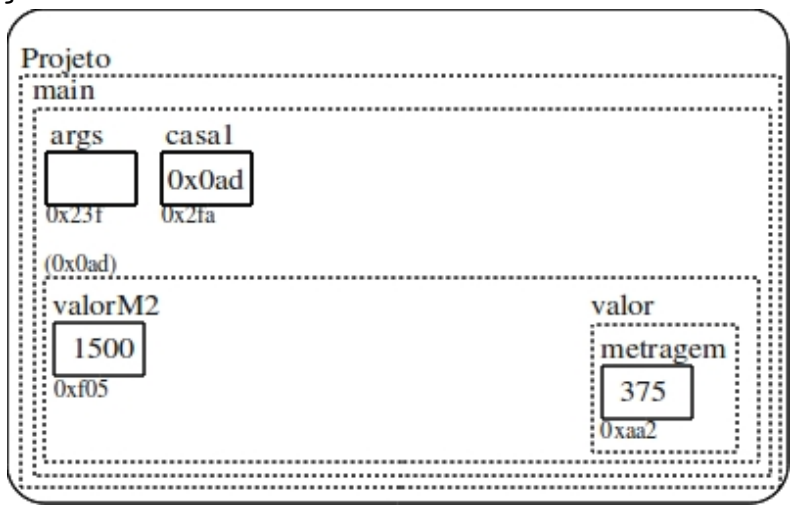


Comportamento na memória

```
class AreaCasa {
    static double valorM2 = 1500;

    static double area(double lateral,
                       double cquarto)...

    static double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                       AreaCasa.area(15,10) ) );
    }
}
```

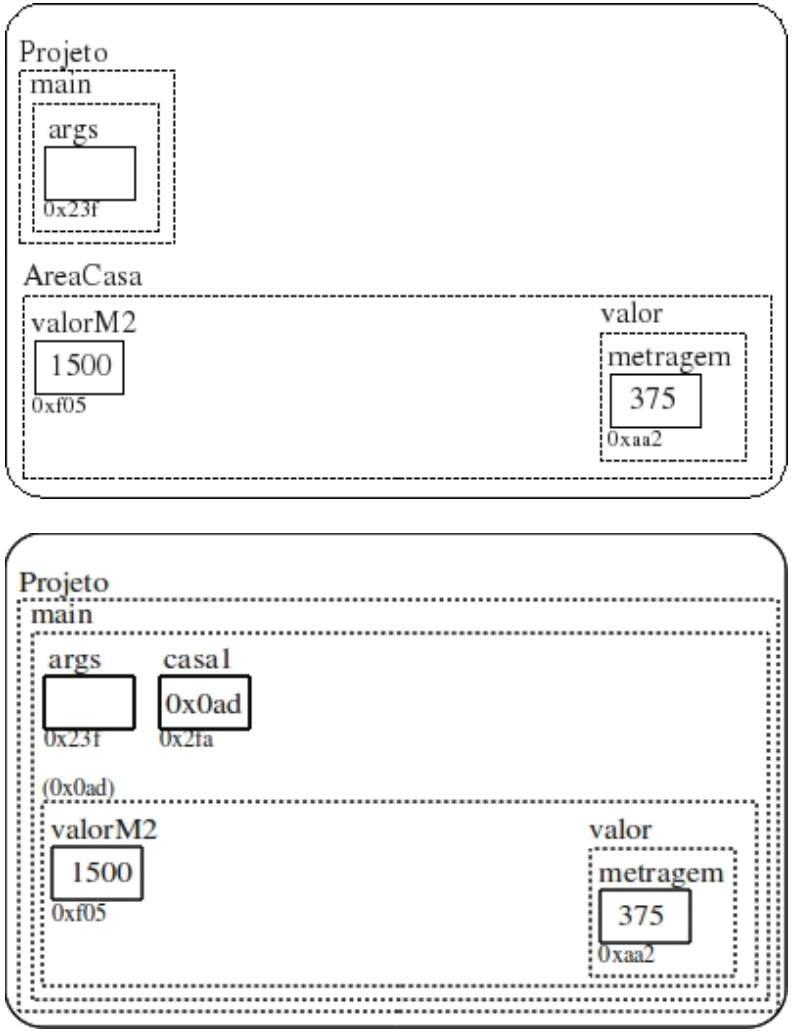



```
class AreaCasa {
    double valorM2 = 1500;

    double area(double lateral,
                       double cquarto) ...

    double valor(double area) ...
}

class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();

        System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
    }
}
```



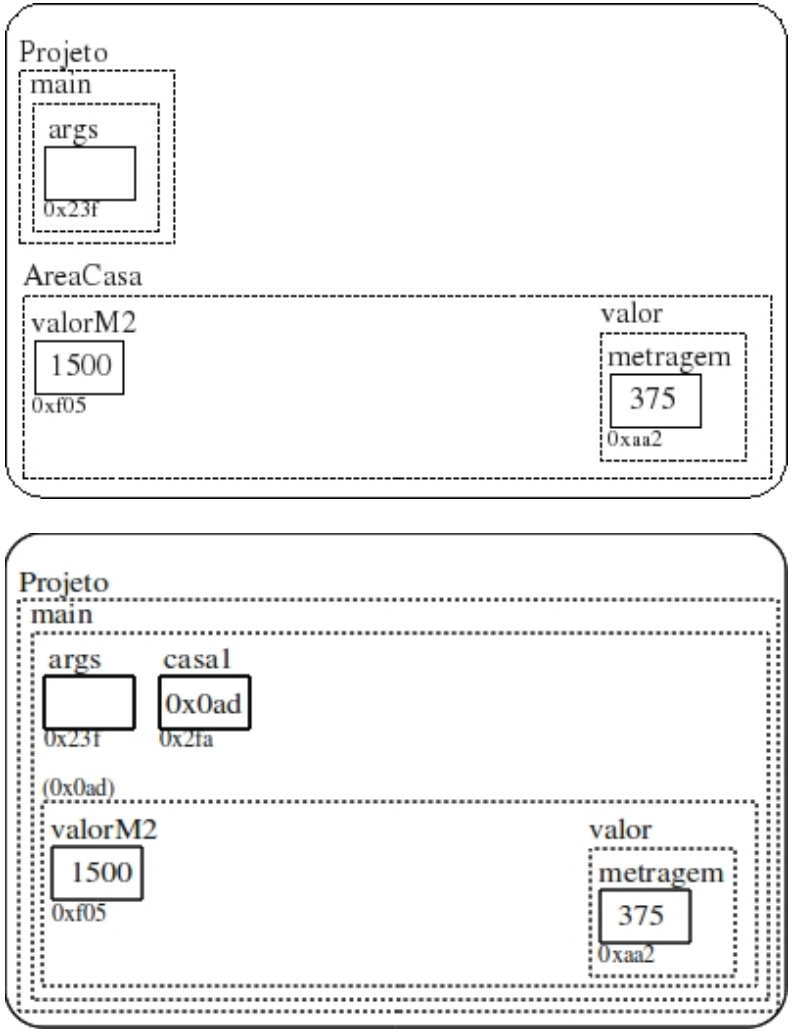
Objetos em Java

- Note que:



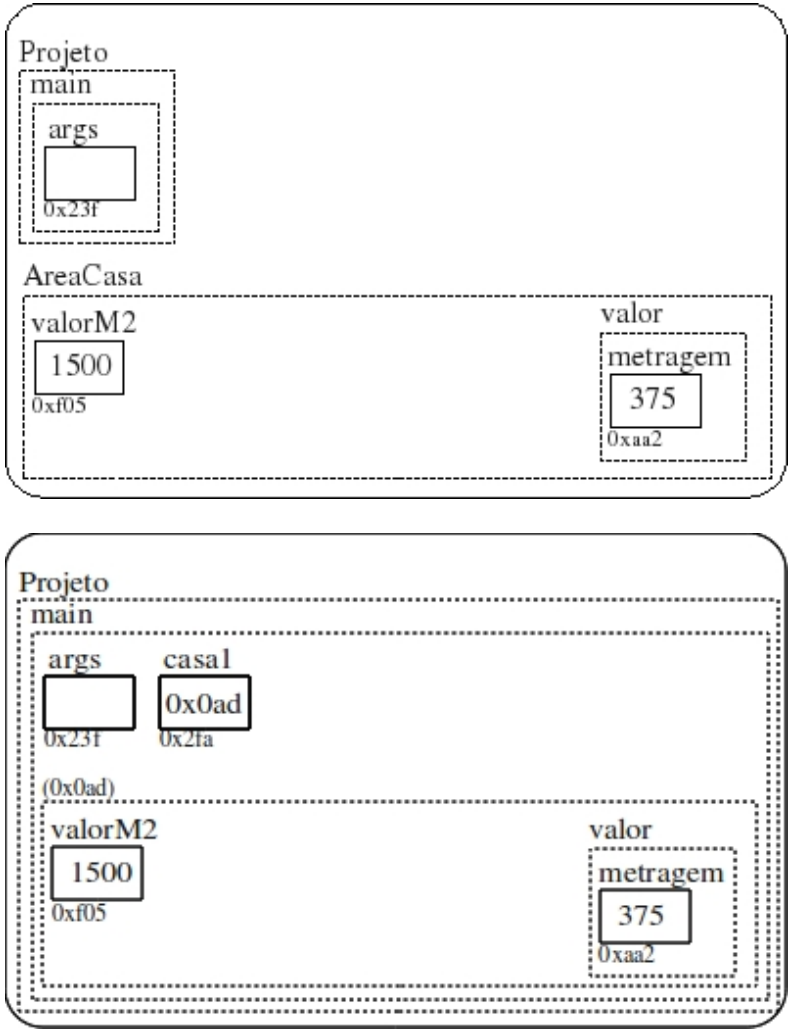
Objetos em Java

- Note que:
  - Enquanto *AreaCasa* é visível por todos, *casa1* só é vista dentro do *main* de *Projeto*



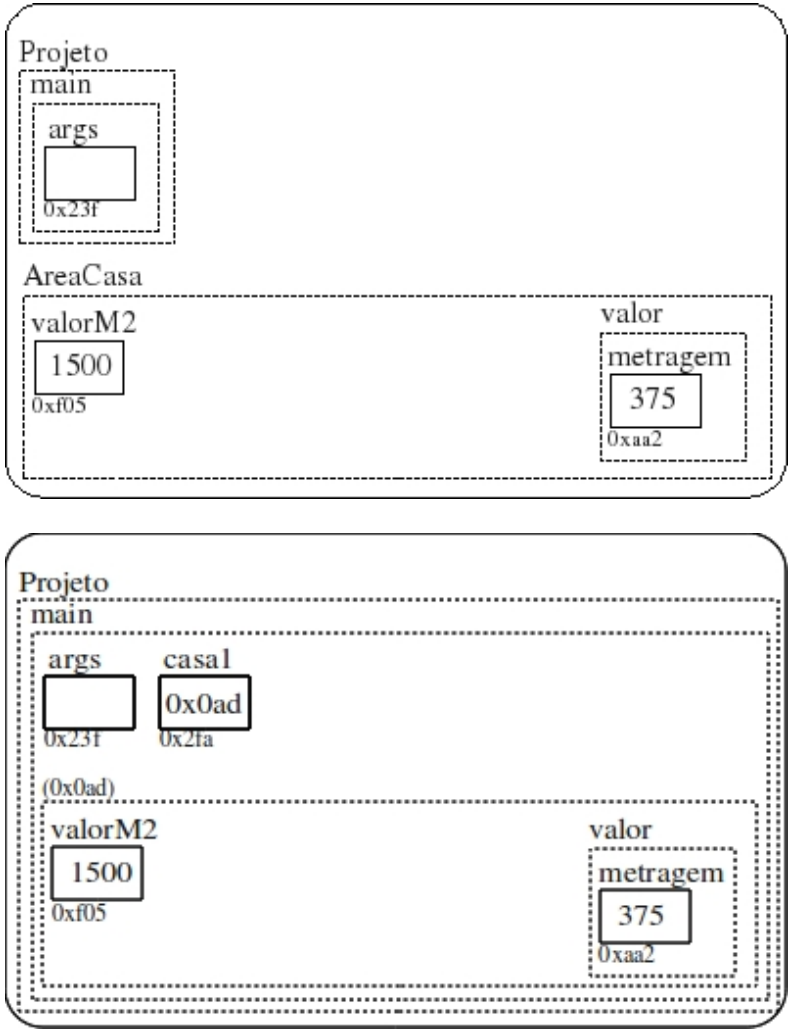
Objetos em Java

- Note que:
  - Enquanto *AreaCasa* é visível por todos, *casa1* só é vista dentro do *main* de *Projeto*
    - ★ Esse é seu escopo



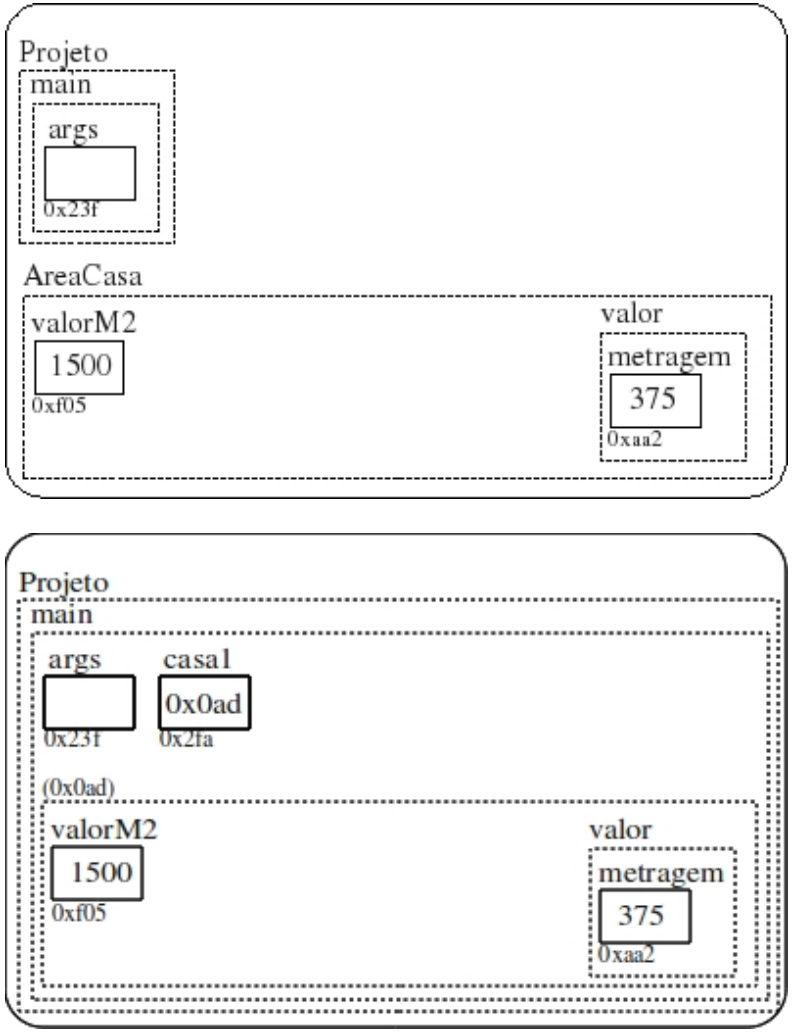
Objetos em Java

- Note que:
  - Enquanto *AreaCasa* é visível por todos, *casa1* só é vista dentro do *main* de *Projeto*
    - ★ Esse é seu escopo
  - *Projeto* nada mais é que mais uma classe dentro do programa (também chamado de processo)



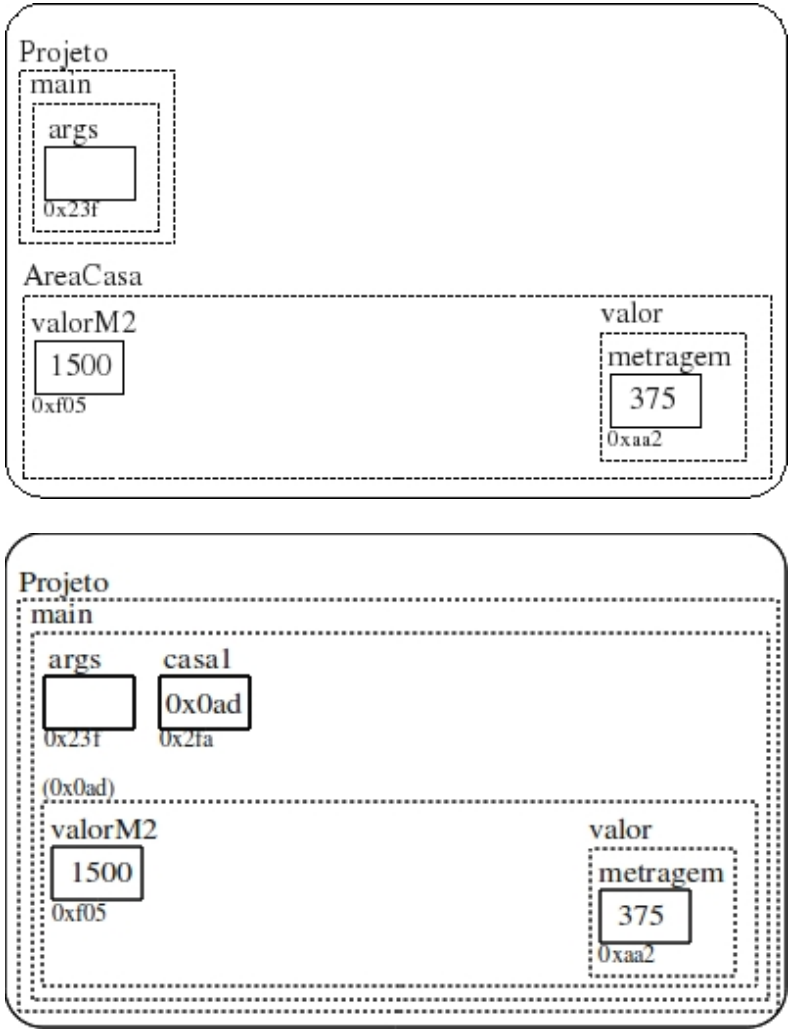
Objetos em Java

- Note que:
  - Enquanto *AreaCasa* é visível por todos, *casa1* só é vista dentro do *main* de *Projeto*
    - ★ Esse é seu escopo
  - *Projeto* nada mais é que mais uma classe dentro do programa (também chamado de processo)
    - ★ Pode ser chamada por outras classes, como *AreaCasa* ou *AreaPiscina*



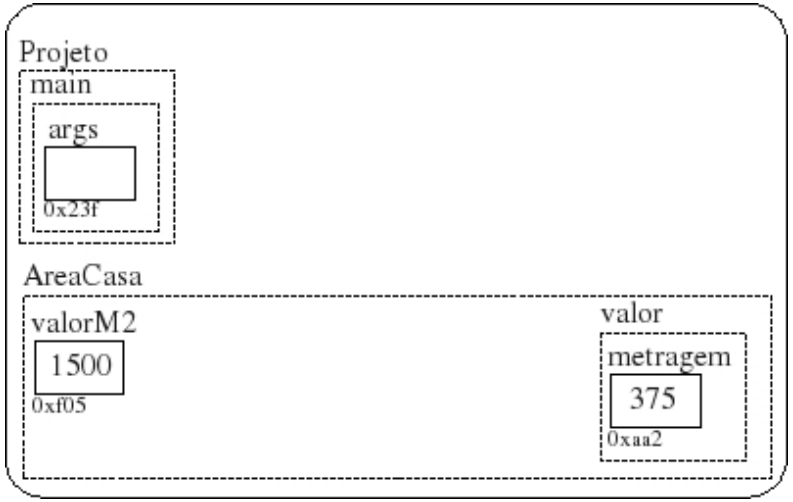
Objetos em Java

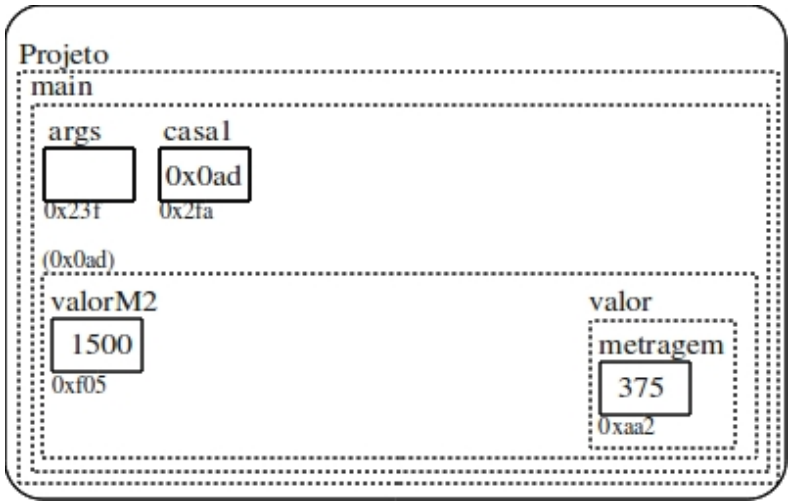
- Note que:
  - Enquanto *AreaCasa* é visível por todos, *casa1* só é vista dentro do *main* de *Projeto*
    - ★ Esse é seu escopo
  - *Projeto* nada mais é que mais uma classe dentro do programa (também chamado de processo)
    - ★ Pode ser chamada por outras classes, como *AreaCasa* ou *AreaPiscina*
- Embora bastante semelhantes, as diferenças ocorrem quando criamos mais de um objeto



Objetos em Java

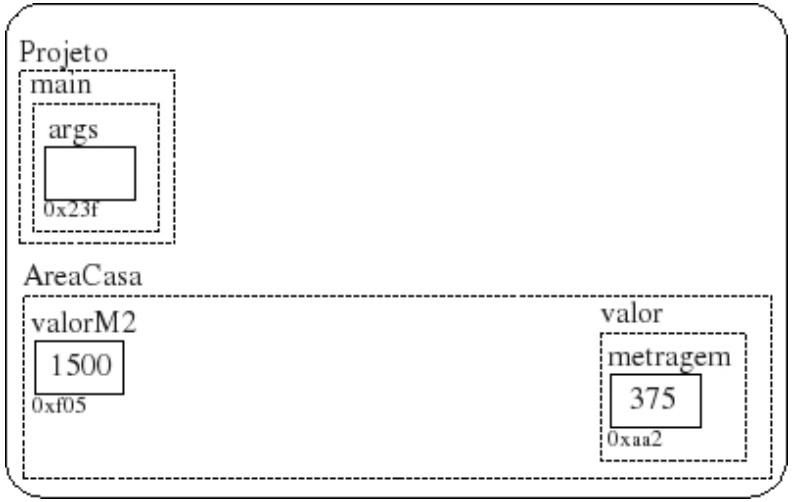
- Note que:
  - Enquanto *AreaCasa* é visível por todos, *casa1* só é vista dentro do *main* de *Projeto*
    - ★ Esse é seu escopo
  - *Projeto* nada mais é que mais uma classe dentro do programa (também chamado de processo)
    - ★ Pode ser chamada por outras classes, como *AreaCasa* ou *AreaPiscina*
- Embora bastante semelhantes, as diferenças ocorrem quando criamos mais de um objeto
  - Podemos sim ter mais de um objeto de uma mesma classe

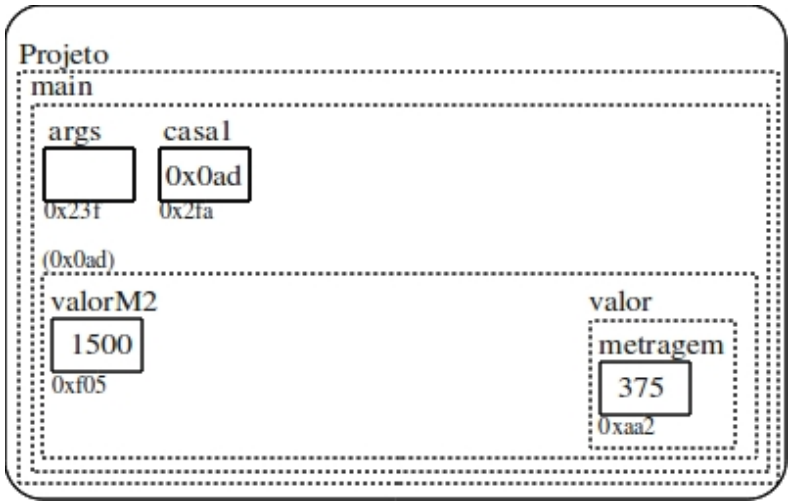




Objetos em Java

- Note que:
  - Enquanto *AreaCasa* é visível por todos, *casa1* só é vista dentro do *main* de *Projeto*
    - ★ Esse é seu escopo
  - *Projeto* nada mais é que mais uma classe dentro do programa (também chamado de processo)
    - ★ Pode ser chamada por outras classes, como *AreaCasa* ou *AreaPiscina*
- Embora bastante semelhantes, as diferenças ocorrem quando criamos mais de um objeto





  - Podemos sim ter mais de um objeto de uma mesma classe
  - Ocuparão porções diferentes da memória

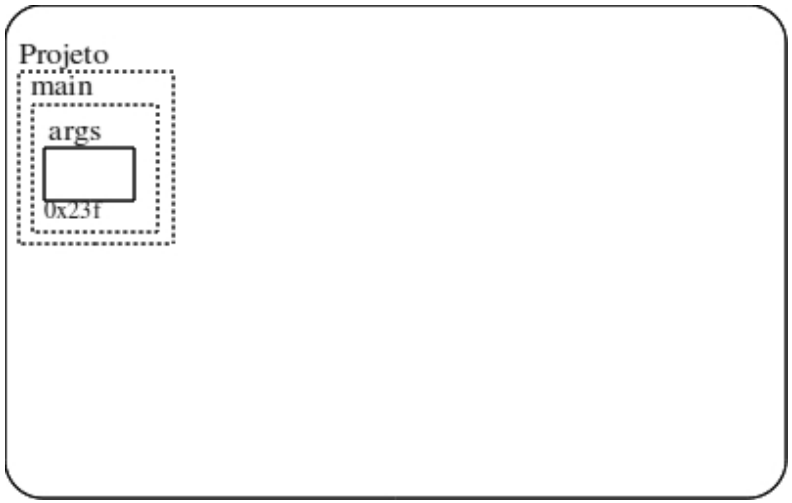
Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                      AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                       AreaCasa.area(18,8) ) );
     }
 }
```

```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                           casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                           casa2.area(18,8) ) );
     }
 }
```

Memória com mais de um objeto

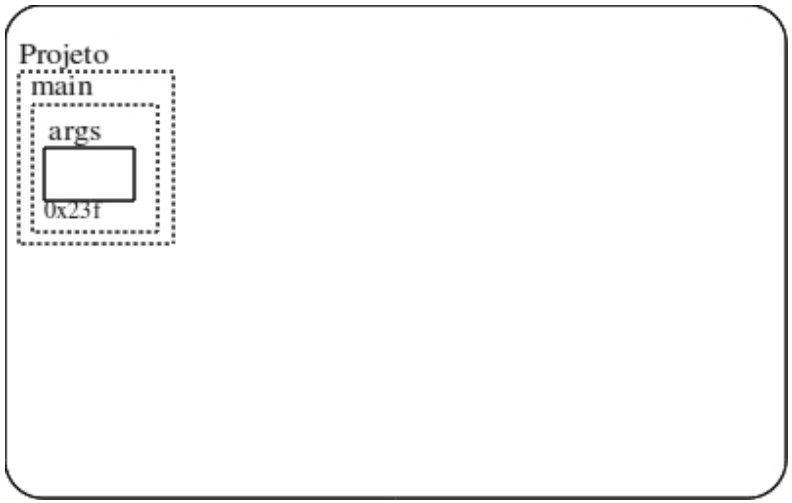
```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                      AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                       AreaCasa.area(18,8) ) );
     }
 }
```



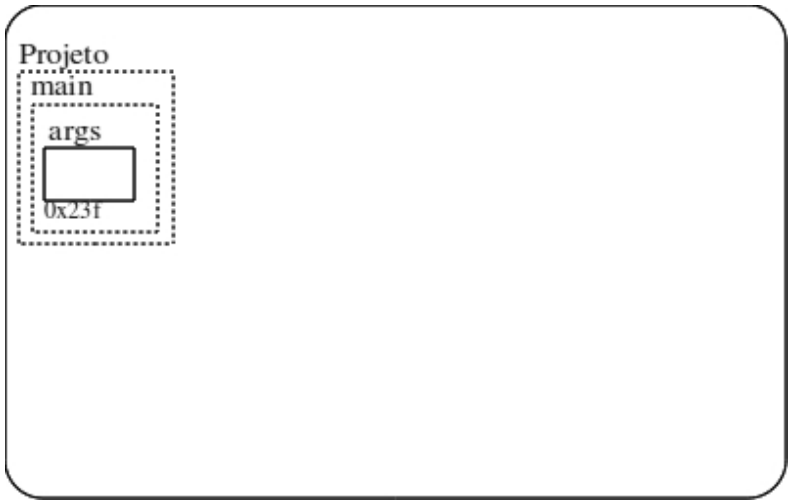

```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                          casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                          casa2.area(18,8) ) );
     }
 }
```

Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                      AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                       AreaCasa.area(18,8) ) );
     }
 }
```

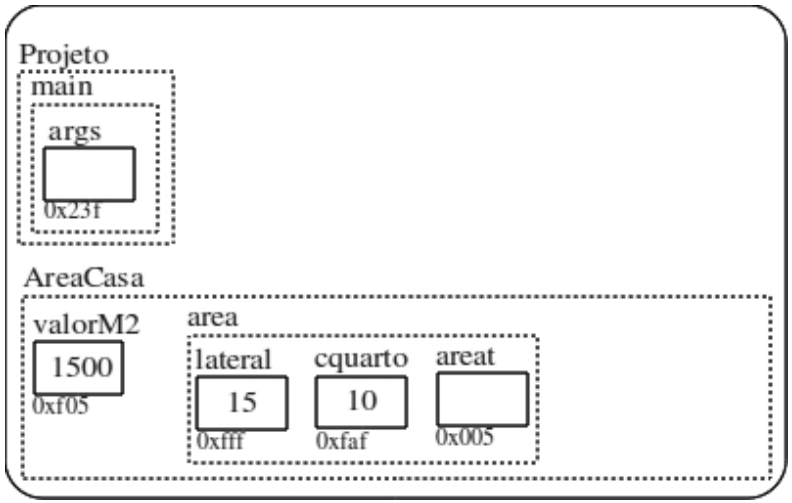



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                          casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                          casa2.area(18,8) ) );
     }
 }
```

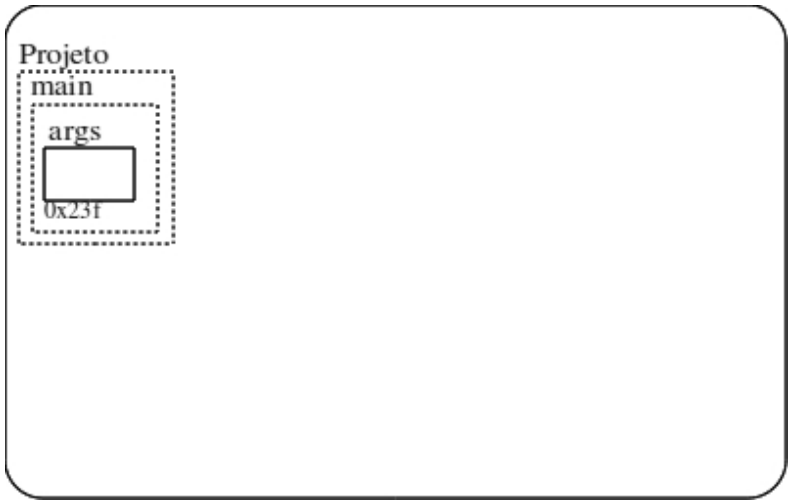


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                     AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                      AreaCasa.area(18,8) ) );
     }
 }
```

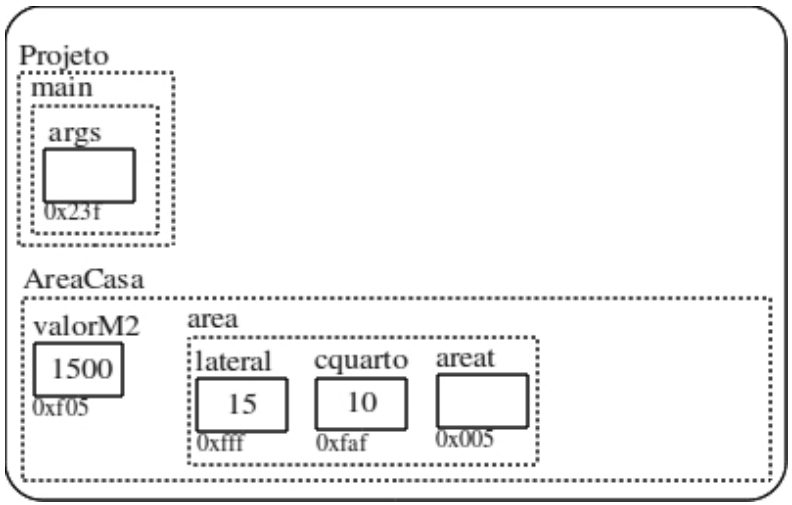



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                          casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                          casa2.area(18,8) ) );
     }
 }
```

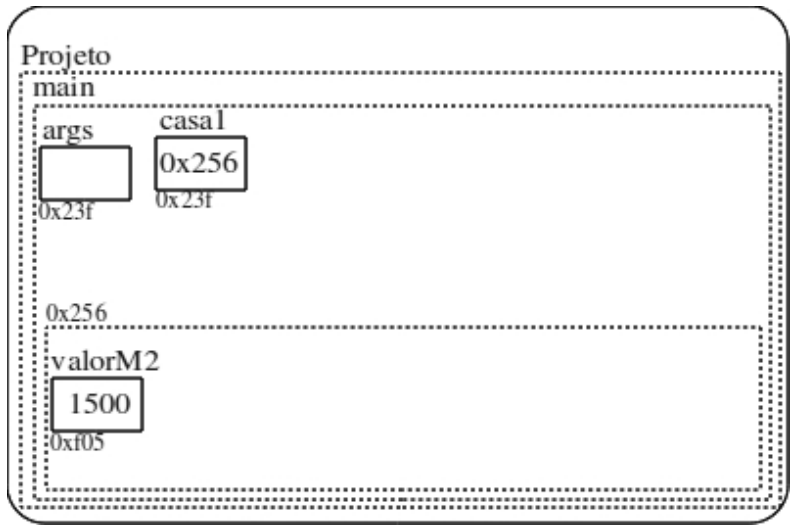


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                         AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                          AreaCasa.area(18,8) ) );
     }
 }
```

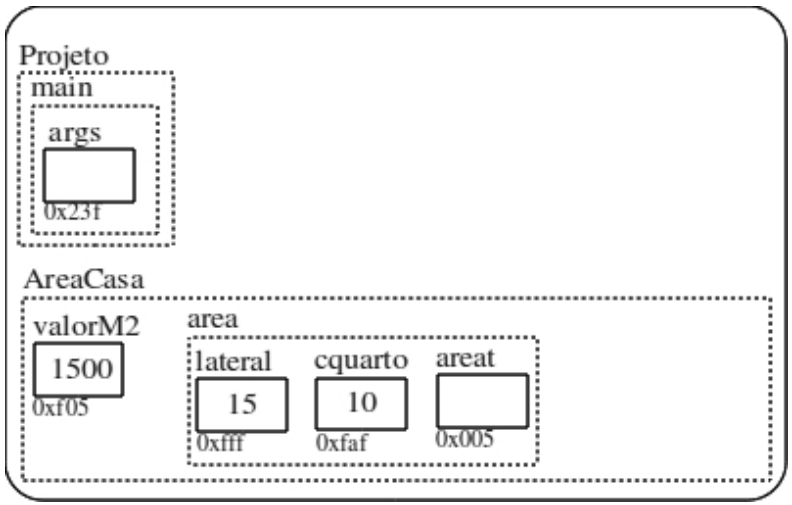



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                          casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                          casa2.area(18,8) ) );
     }
 }
```

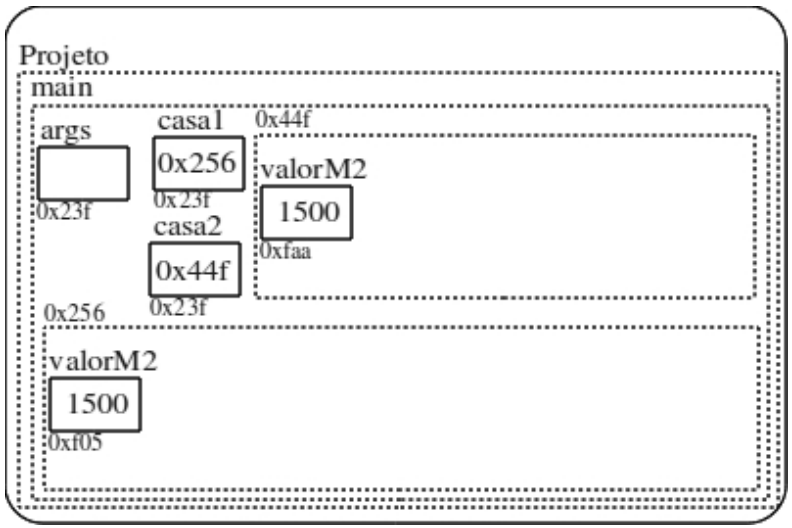


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                         AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                          AreaCasa.area(18,8) ) );
     }
 }
```

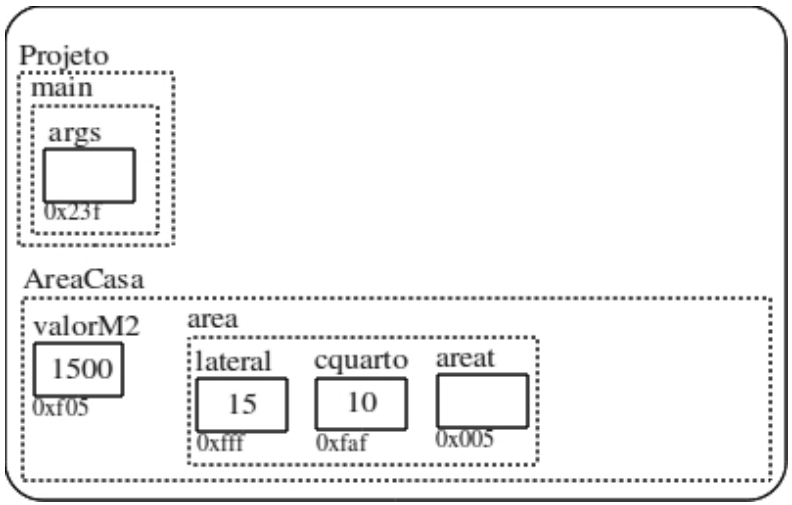



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                          casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                          casa2.area(18,8) ) );
     }
 }
```

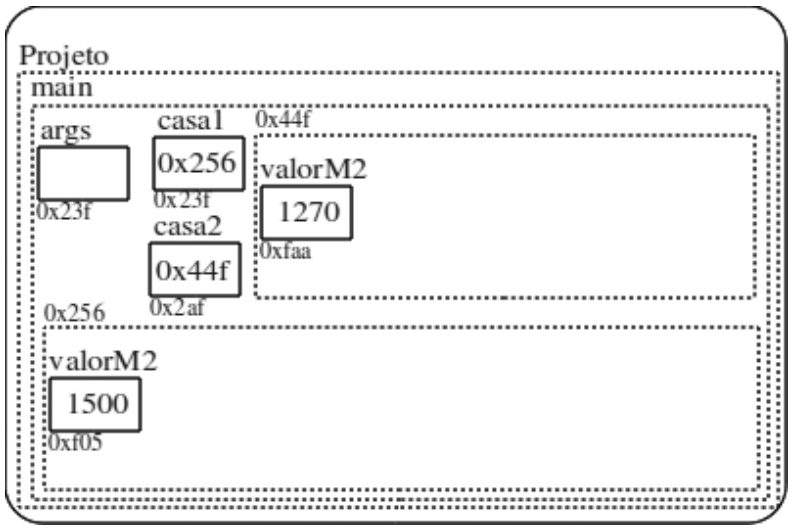


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                     AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                      AreaCasa.area(18,8) ) );
     }
 }
```

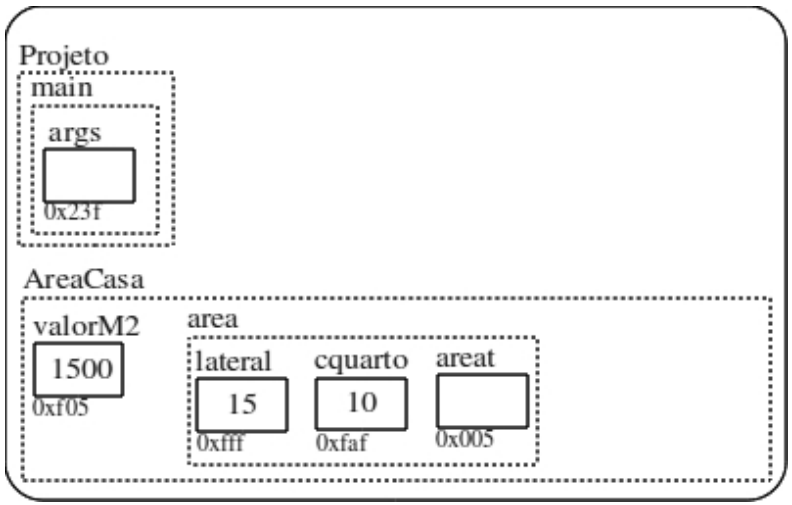



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                         casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                         casa2.area(18,8) ) );
     }
 }
```

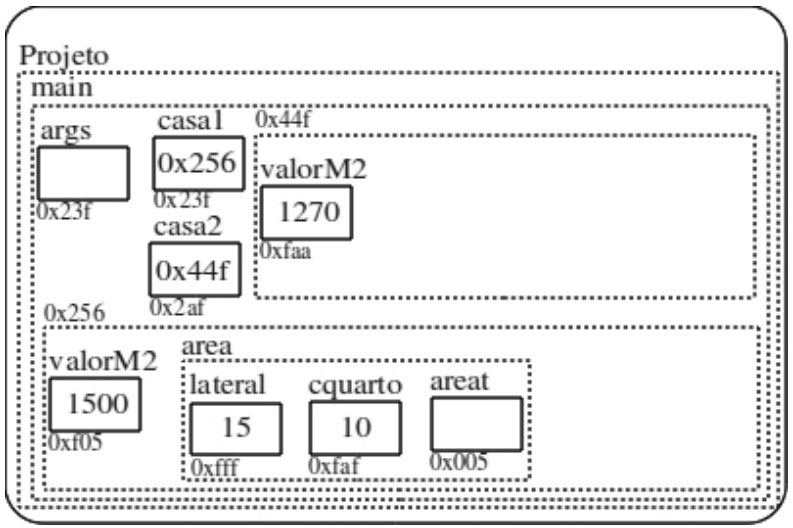


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                         AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                          AreaCasa.area(18,8) ) );
     }
 }
```

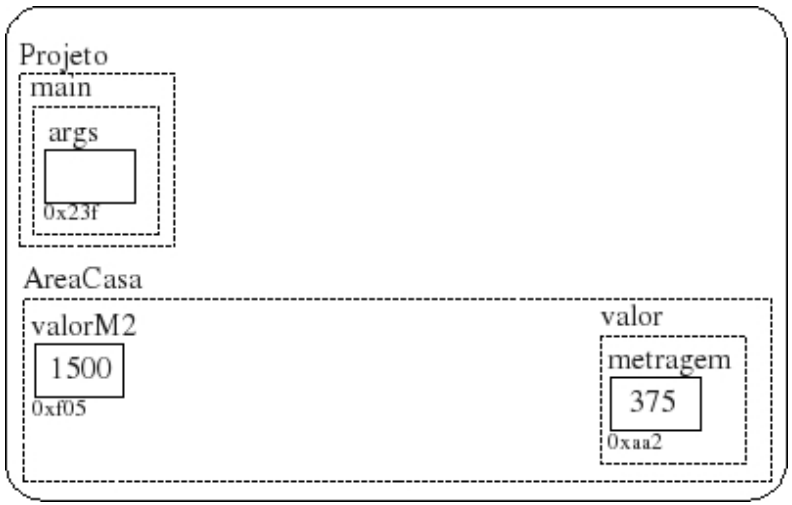



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                         casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                         casa2.area(18,8) ) );
     }
 }
```

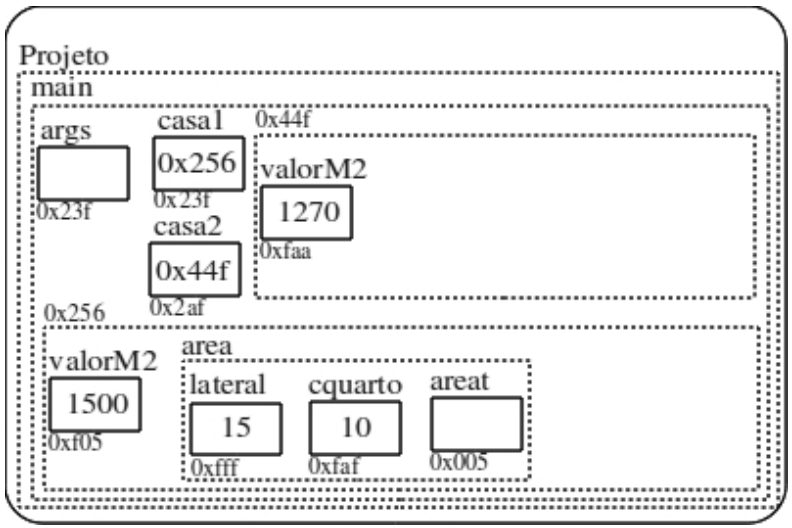


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                     AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                      AreaCasa.area(18,8) ) );
     }
 }
```

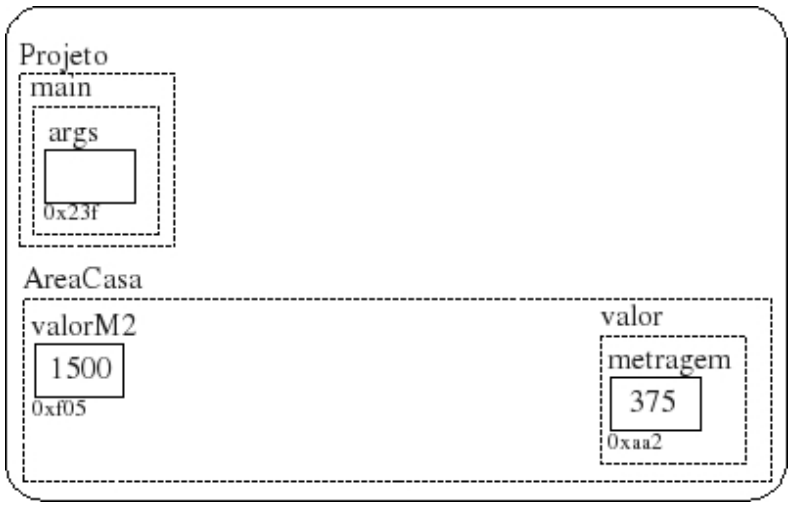



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                          casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                          casa2.area(18,8) ) );
     }
 }
```

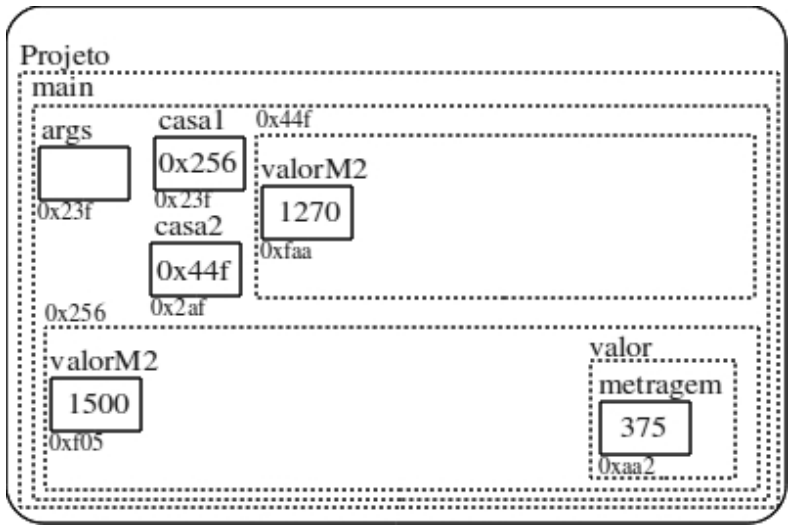


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                     AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                      AreaCasa.area(18,8) ) );
     }
 }
```

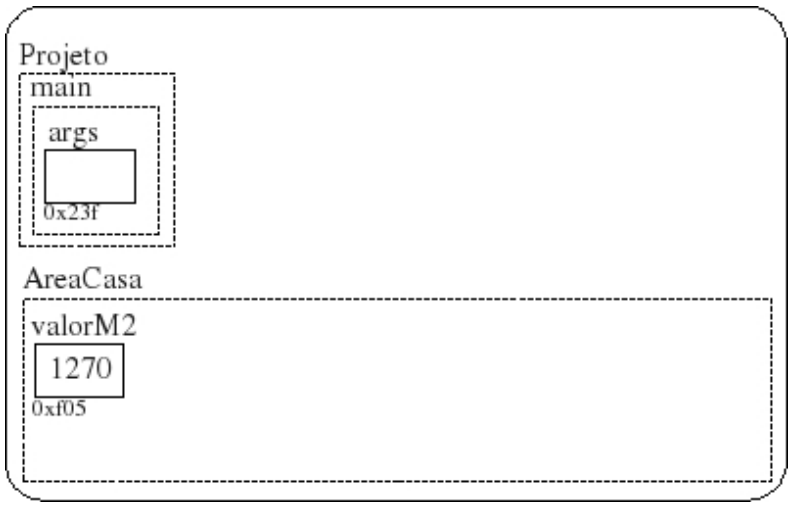



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                         casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                         casa2.area(18,8) ) );
     }
 }
```

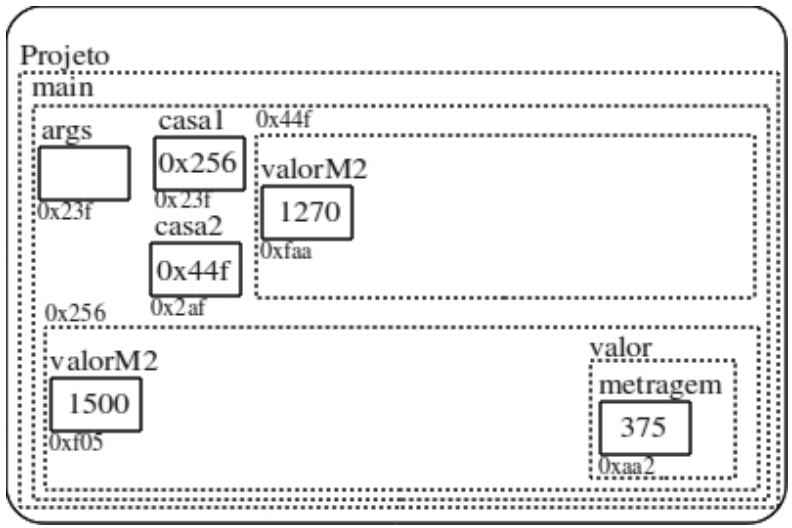


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                     AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                      AreaCasa.area(18,8) ) );
     }
 }
```

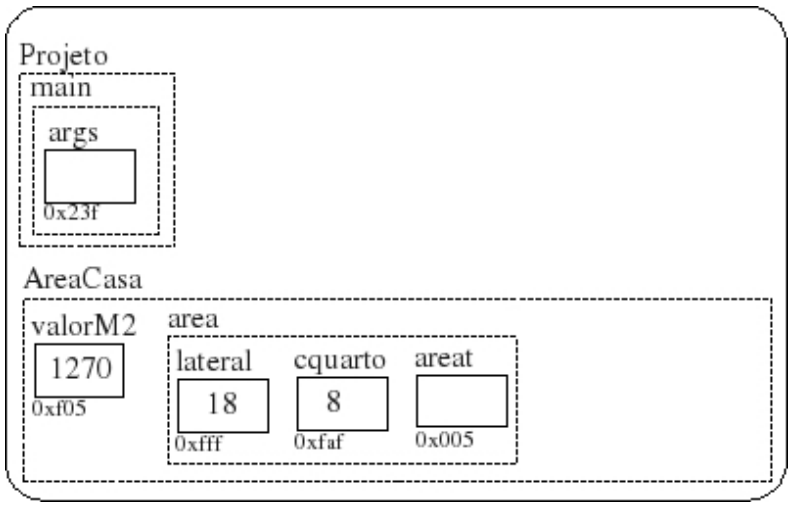



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                          casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                          casa2.area(18,8) ) );
     }
 }
```

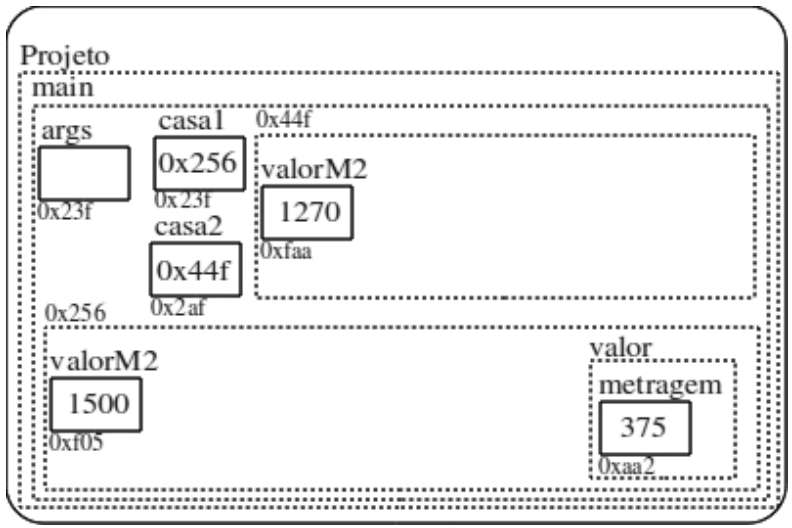


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                     AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                     AreaCasa.area(18,8) ) );
     }
 }
```

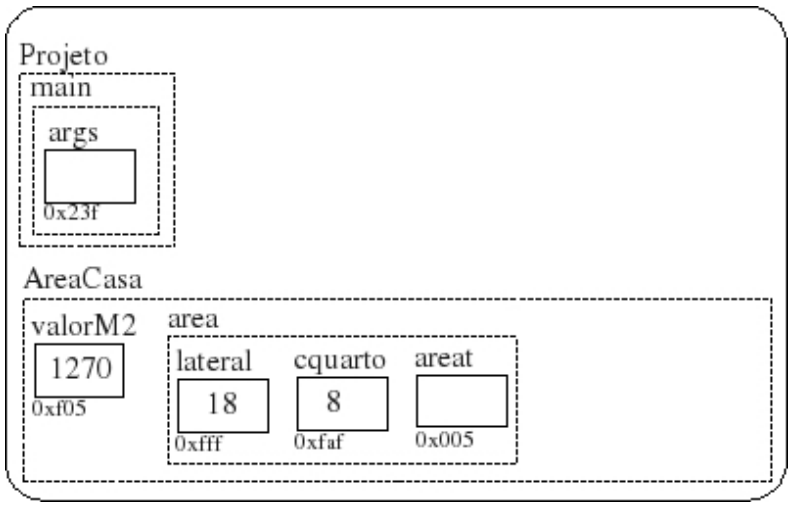



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                     casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                     casa2.area(18,8) ) );
     }
 }
```

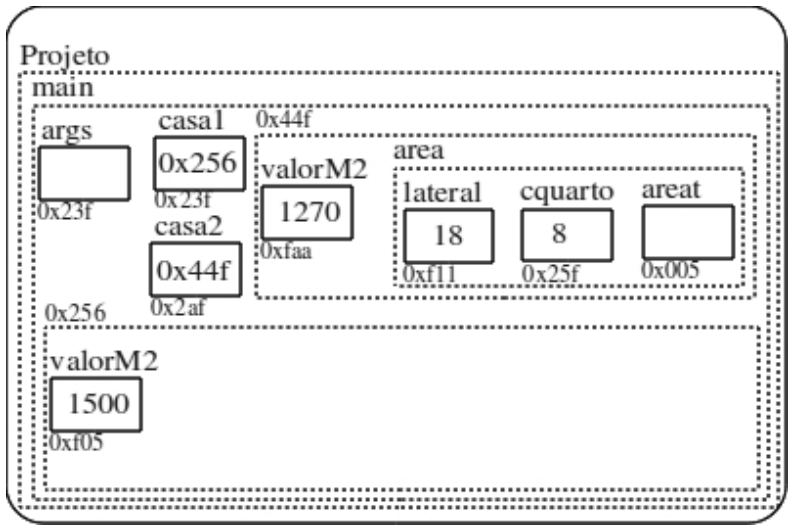


Memória com mais de um objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         System.out.println( AreaCasa.valor(
                     AreaCasa.area(15,10) ) );

         AreaCasa.valorM2 = 1270;

         System.out.println( AreaCasa.valor(
                     AreaCasa.area(18,8) ) );
     }
 }
```

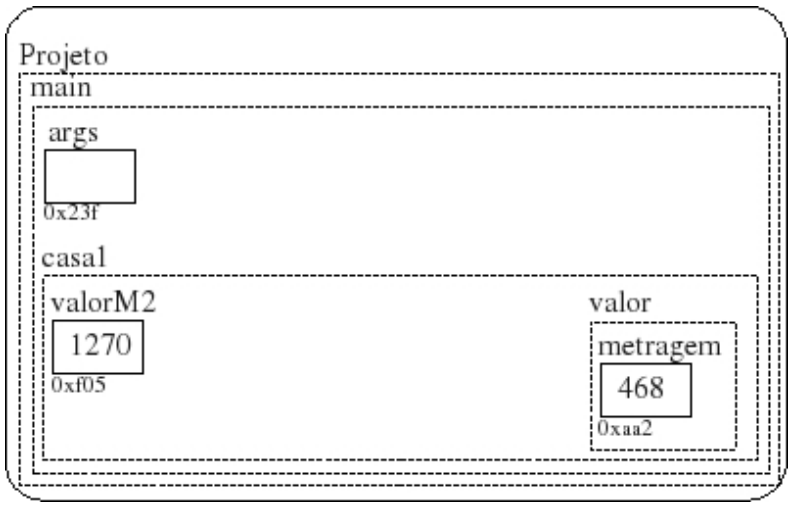



```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                     casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                     casa2.area(18,8) ) );
     }
 }
```

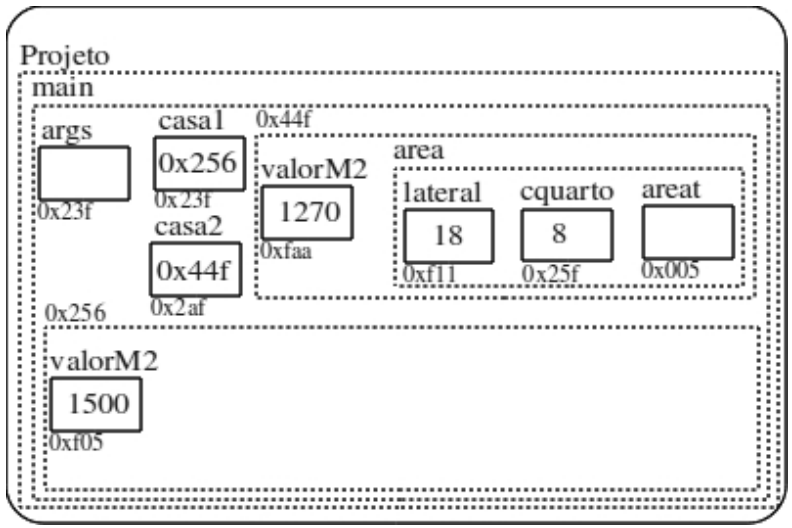


Memória com mais de um objeto

```
class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                    AreaCasa.area(15,10) ) );

        AreaCasa.valorM2 = 1270;

        System.out.println( AreaCasa.valor(
                    AreaCasa.area(18,8) ) );
    }
}
```

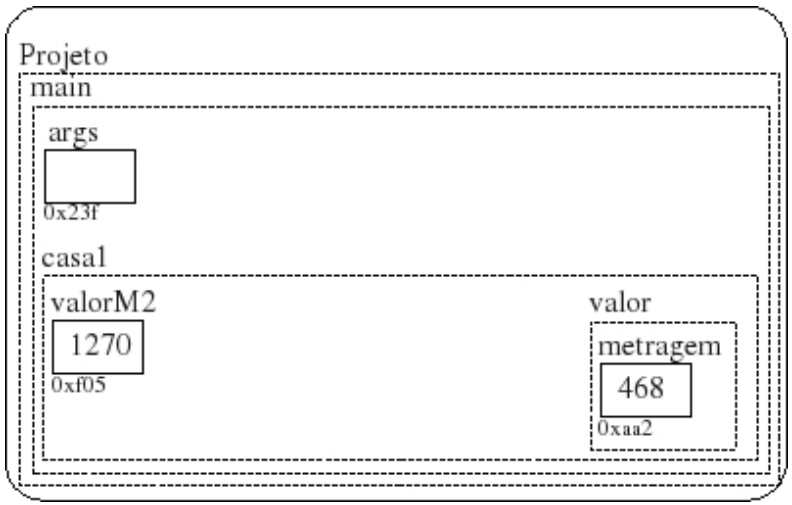



```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa();

        casa2.valorM2 = 1270;

        System.out.println( casa1.valor(
                    casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                    casa2.area(18,8) ) );
    }
}
```

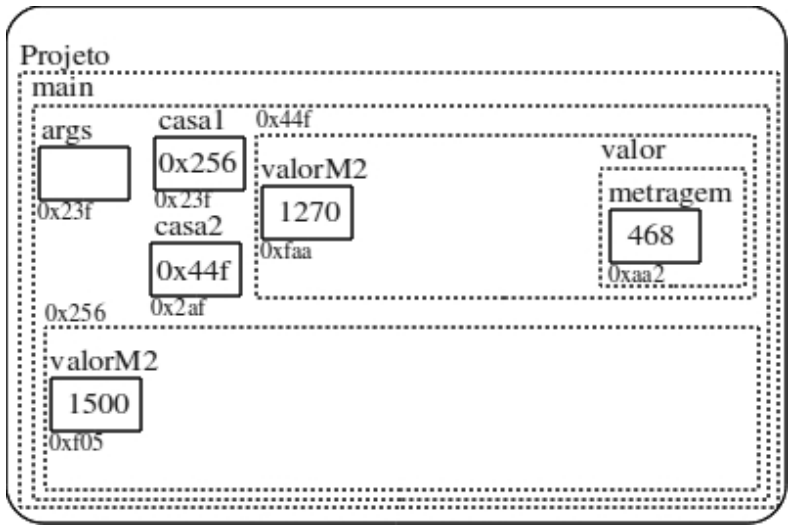


Memória com mais de um objeto

```
class Projeto {
    ...
    public static void main(String[] args) {
        System.out.println( AreaCasa.valor(
                    AreaCasa.area(15,10) ) );

        AreaCasa.valorM2 = 1270;

        System.out.println( AreaCasa.valor(
                    AreaCasa.area(18,8) ) );
    }
}
```




```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa();

        casa2.valorM2 = 1270;

        System.out.println( casa1.valor(
                    casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                    casa2.area(18,8) ) );
    }
}
```



Construtores

- Revisitando a classe *Projeto*:

```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                         casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                         casa2.area(18,8) ) );
     }
 }
```

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                         casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                         casa2.area(18,8) ) );
     }
 }
```

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto

```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                             casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
     }
 }
```

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto
  - Como?

```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                             casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
     }
}
```

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto
  - Como?
    - ★ Construtores

```
class Projeto {
     ...
     public static void main(String[] args) {
         AreaCasa casa1 = new AreaCasa();
         AreaCasa casa2 = new AreaCasa();

         casa2.valorM2 = 1270;

         System.out.println( casa1.valor(
                             casa1.area(15,10) ) );
         System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
     }
}
```

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto
  - Como?
    - ★ Construtores

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa();

        casa2.valorM2 = 1270;

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                            casa2.area(18,8) ) );
    }
}
```

- Construtores são métodos chamados quando da criação do objeto

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto
  - Como?
    - ★ Construtores

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa();

        casa2.valorM2 = 1270;

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                            casa2.area(18,8) ) );
    }
}
```

- Construtores são métodos chamados quando da criação do objeto
  - Servem para inicialização de atributos

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto
  - Como?
    - ★ Construtores

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa();

        casa2.valorM2 = 1270;

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                            casa2.area(18,8) ) );
    }
}
```

- Construtores são métodos chamados quando da criação do objeto
  - Servem para inicialização de atributos
  - Ou execução de algum método, antes de qualquer outra coisa

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto
  - Como?
    - ★ Construtores

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa();

        casa2.valorM2 = 1270;

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                            casa2.area(18,8) ) );
    }
}
```

- Construtores são métodos chamados quando da criação do objeto
  - Servem para inicialização de atributos
  - Ou execução de algum método, antes de qualquer outra coisa
- Toda classe tem seu contrutor

Construtores

- Revisitando a classe *Projeto*:
  - Tivemos que mudar o valor do m^2 após criarmos o objeto
  - Seria interessante fornecermos esse valor ao criarmos o objeto
  - Como?
    - ★ Construtores

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa();

        casa2.valorM2 = 1270;

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                            casa2.area(18,8) ) );
    }
}
```

- Construtores são métodos chamados quando da criação do objeto
  - Servem para inicialização de atributos
  - Ou execução de algum método, antes de qualquer outra coisa
- Toda classe tem seu contrutor
  - Se não for explicitamente declarado, o Java fornecerá um padrão

Construtores

- Construtores são definidos com o mesmo nome da classe

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Construtores são definidos com o mesmo nome da classe
  - Com qualquer número de atributos

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Construtores são definidos com o mesmo nome da classe
  - Com qualquer número de atributos
  - Executando qualquer código dentro dele

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Construtores são definidos com o mesmo nome da classe
  - Com qualquer número de atributos
  - Executando qualquer código dentro dele
- Permitem que se crie o objeto com o atributo atualizado

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa(1500);
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                        casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                         casa2.area(18,8) ) );
    }
}
```

Construtores

- Construtores são definidos com o mesmo nome da classe
  - Com qualquer número de atributos
  - Executando qualquer código dentro dele
- Permitem que se crie o objeto com o atributo atualizado
  - Reduzem a possibilidade de erros

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa(1500);
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

Construtores

- Construtores são definidos com o mesmo nome da classe
  - Com qualquer número de atributos
  - Executando qualquer código dentro dele
- Permitem que se crie o objeto com o atributo atualizado
  - Reduzem a possibilidade de erros
  - Deixam mais legível o código

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa(1500);
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                        casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                         casa2.area(18,8) ) );
    }
}
```

Construtores

- Por que tivemos que pôr o 1500?

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa(1500);
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;


    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Por que tivemos que pôr o 1500?
- E se fizéssemos:

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;



    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Por que tivemos que pôr o 1500?
- E se fizéssemos:

```
Projeto.java:24: cannot find symbol
symbol  : constructor AreaCasa()
location: class AreaCasa
        AreaCasa casa1 = new AreaCasa();
                             ^
1 error
```

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;



    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Por que tivemos que pôr o 1500?
- E se fizéssemos:

```
Projeto.java:24: cannot find symbol
symbol  : constructor AreaCasa()
location: class AreaCasa
        AreaCasa casa1 = new AreaCasa();
                         ^
1 error
```

- Quando é definido um construtor, não é mais possível usar o construtor padrão

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;


    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Por que tivemos que pôr o 1500?
- E se fizéssemos:

```
Projeto.java:24: cannot find symbol
symbol  : constructor AreaCasa()
location: class AreaCasa
        AreaCasa casa1 = new AreaCasa();
                         ^
1 error
```

- Quando é definido um construtor, não é mais possível usar o construtor padrão
  - O construtor sem parâmetros

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;


    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Por que tivemos que pôr o 1500?
- E se fizéssemos:

```
Projeto.java:24: cannot find symbol
symbol  : constructor AreaCasa()
location: class AreaCasa
        AreaCasa casa1 = new AreaCasa();
                         ^
1 error
```

- Quando é definido um construtor, não é mais possível usar o construtor padrão
  - O construtor sem parâmetros
- A menos que implementemos um:

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa() {}

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

Construtores

- Por que tivemos que pôr o 1500?
- E se fizéssemos:

```
Projeto.java:24: cannot find symbol
symbol  : constructor AreaCasa()
location: class AreaCasa
        AreaCasa casa1 = new AreaCasa();
                         ^
1 error
```

- Quando é definido um construtor, não é mais possível usar o construtor padrão
  - O construtor sem parâmetros
- A menos que implementemos um:
  - Uma classe pode ter mais de um construtor

```
class Projeto {
    ...
    public static void main(String[] args) {
        AreaCasa casa1 = new AreaCasa();
        AreaCasa casa2 = new AreaCasa(1270);

        System.out.println( casa1.valor(
                            casa1.area(15,10) ) );
        System.out.println( casa2.valor(
                             casa2.area(18,8) ) );
    }
}
```

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

    AreaCasa() {}

    AreaCasa(double val) {
        valorM2 = val;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo
- Que fazer?

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo
- Que fazer?

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo
- Que fazer?
- *this* é uma referência ao próprio objeto

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo
- Que fazer?

- *this* é uma referência ao próprio objeto
  - Também usada para substituir algum outro construtor

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo
- Que fazer?

- *this* é uma referência ao próprio objeto
  - Também usada para substituir algum outro construtor

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo
- Que fazer?

- *this* é uma referência ao próprio objeto
  - Também usada para substituir algum outro construtor
- Útil em ambigüidades: quando há dois elementos com mesmo nome

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

This

- E se mudarmos o nome do parâmetro, o que acontece?
  - O compilador modifica o parâmetro, não o atributo
- Que fazer?

- *this* é uma referência ao próprio objeto
  - Também usada para substituir algum outro construtor
- Útil em ambigüidades: quando há dois elementos com mesmo nome
  - Do contrário pode ser omitida

```
class AreaCasa {
    /* valor do metro quadrado da casa */
    double valorM2 = 1500;

   AreaCasa() {}

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

Múltiplos Construtores

- Quando há mais de um construtor na classe:

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

Múltiplos Construtores

- Quando há mais de um construtor na classe:
  - Têm o mesmo nome – diferem apenas nos parâmetros

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

Múltiplos Construtores

- Quando há mais de um construtor na classe:
  - Têm o mesmo nome – diferem apenas nos parâmetros
  - O compilador escolhe o construtor correto conforme a assinatura

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

Múltiplos Construtores

- Quando há mais de um construtor na classe:
  - Têm o mesmo nome – diferem apenas nos parâmetros
  - O compilador escolhe o construtor correto conforme a assinatura
    - ★ A assinatura de um método ou construtor corresponde a seu nome, tipo dos parâmetros e ordem dos parâmetros

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

Múltiplos Construtores

- Quando há mais de um construtor na classe:
  - Têm o mesmo nome – diferem apenas nos parâmetros
  - O compilador escolhe o construtor correto conforme a assinatura
    - ★ A assinatura de um método ou construtor corresponde a seu nome, tipo dos parâmetros e ordem dos parâmetros

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

  - O mesmo procedimento é seguido com métodos de mesmo nome, porém com parâmetros diferentes

Múltiplos Construtores

- Quando há mais de um construtor na classe:
  - Têm o mesmo nome – diferem apenas nos parâmetros
  - O compilador escolhe o construtor correto conforme a assinatura
    - ★ A assinatura de um método ou construtor corresponde a seu nome, tipo dos parâmetros e ordem dos parâmetros

```
class AreaCasa {
    double valorM2;

    AreaCasa() {
        this(1500.0);
    }

    AreaCasa(double valorM2) {
        this.valorM2 = valorM2;
    }
    ...
}
```

  - O mesmo procedimento é seguido com métodos de mesmo nome, porém com parâmetros diferentes
    - ★ Polimorfismo de nome

Classes e Objetos

- Como ficaria *AreaPiscina*, nessa nova forma?

Classes e Objetos

- Como ficaria *AreaPiscina*, nessa nova forma?

```
class AreaPiscina {
    static final int ALVENARIA = 0;
    static final int VINIL = 1;
    static final int FIBRA = 2;
    static final int PLASTICO = 3;

    double[] precos;

    static char[][] nomes = {{'A','l','v','e','n',
                              'a','r','i','a'},
                             {'V','i','n','i','l'},
                             {'F','i','b','r','a'},
                             {'P','l','á','s','t',
                               'i','c','o'}};

    AreaPiscina() {
        double[] aux = {1500, 1100, 750, 500};
        this.precos = aux;
    }

    AreaPiscina(double[] precos) {
        this.precos = precos;
    }

    double area(double raio) ...

    double valor(double area, int material)
    ...

    void carregaVal(double[][] m) ...

    double[][] calculaFinal(double[][] val,
                            double[][] desc) ...
}
```