



# Campos e Métodos Estáticos

MAT A55



# Criando Aplicações em Java

- Até o momento criamos classes que mapeiam modelos do mundo real, mas não criamos programas de computador.
- Programas completos executam um algoritmo com passos bem definidos para realização de uma tarefa.
- O primeiro passo para executar um programa é determinar um ponto de entrada, ponto inicial



# Campos e Métodos Estáticos

## ○ Campos estáticos

- Campos compartilhados por todas as instâncias de uma classe

## ○ Métodos estáticos

- Métodos que podem ser executados sem que instâncias da classe sejam criadas



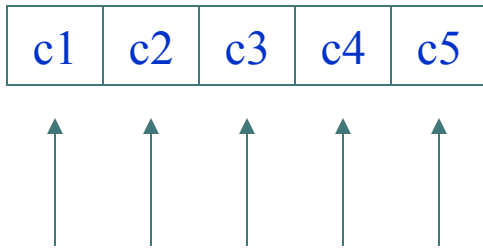
# Campos Estáticos

- Campos de Classe

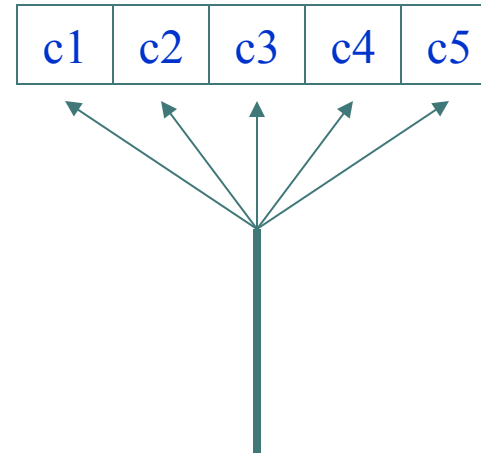
- Somente um valor é armazenado em um campo estático.
- A alteração deste valor por qualquer instância afeta todas as outras instâncias da classe.

# Campos Estáticos

Exemplo:



Banco com múltiplas  
filas



Banco com fila única



# Campos Estáticos

```
class SimuladorDeCaixaDeBanco0
{
    private int númeroDoCliente;
    private int númeroDoCaixa;
```

Exemplo: Caixa de banco

```
SimuladorDeCaixaDeBanco0(int n)
{
    númeroDoCaixa = n;
    númeroDoCliente = 0;
    System.out.println("Caixa "+númeroDoCaixa+" iniciou operação.");
} // fim do construtor
```

```
public void próximoAtendimento()
{
    númeroDoCliente = númeroDoCliente + 1;
    System.out.print("Cliente com a senha número "+númeroDoCliente+", favor ");
    System.out.println("dirigir-se ao caixa número "+númeroDoCaixa+".");
}
} // fim da classe SimuladorDeCaixaDeBanco0
```

SimuladorDeCaixaDeBanco0.java



# Campos Estáticos

## Exemplo: Caixa de banco

```
class DemoSimuladorDeCaixaDeBanco0
{
    public static void main(String[] argumentos)
    {
        SimuladorDeCaixaDeBanco0 c1 = new SimuladorDeCaixaDeBanco0(1);
        ....
        SimuladorDeCaixaDeBanco0 c5 = new SimuladorDeCaixaDeBanco0(5);

        c1.próximoAtendimento();
        c2.próximoAtendimento();
        c2.próximoAtendimento();
        c4.próximoAtendimento();
        c5.próximoAtendimento();
        c3.próximoAtendimento();
        c5.próximoAtendimento();
        c2.próximoAtendimento();

    } // fim do método main
} // fim da classe DemoSimuladorDeCaixaDeBanco0
```

DemoSimuladorDeCaixaDeBanco0.java



# Campos Estáticos

- Simulação No BlueJ



# Campos Estáticos

```
class DemoSimuladorDeCaixaDeBanco0  
{
```

A Classe `SimuladorDeCaixaDeBanco0` tem seus próprios campos.

O campo `númeroDoCliente` é restrito a esta classe. Seus valores estão no escopo de cada instância.

Cada caixa de banco fica independente.

Os clientes possuem senhas que só valeriam para cada caixa

```
    c5.próximoAtendimento();  
    c2.próximoAtendimento();
```

*afeta as outras  
instâncias !!!!*

```
    } // fim do método main  
} // fim da classe DemoSimuladorDeCaixaDeBanco0
```

DemoSimuladorDeCaixaDeBanco0.java



# Campos Estáticos

- Como fazer para simular caixas em um banco que compartilhassem informações???
- Qual o número da senha do próximo cliente a ser atendido, que deve ser um único valor que pode ser visto e modificado por todos os caixas?



# Campos Estáticos

- Uma Solução:
  - Utilizar um contador de senha no main???



# Campos Estáticos

```
class DemoSimuladorDeCaixaDeBanco01
{
    public static void main(String[] argumentos)
    {
```

Utilizar contador de senha no main:

solução ruim: **A cargo do programador usuário!!! Não assegura o uso correto !!!**

**Não existe controle real da sequencia de atendimento. Qualquer número pode ser passado!!! Inclusive, números errados.....**

```
    } // fim do método main
} // fim da classe DemoSimuladorDeCaixaDeBanco01
```

DemoSimuladorDeCaixaDeBanco01.java



# Campos Estáticos

- Outra solução:
  - Número do Cliente ser declarado como estático

# Campos Estáticos

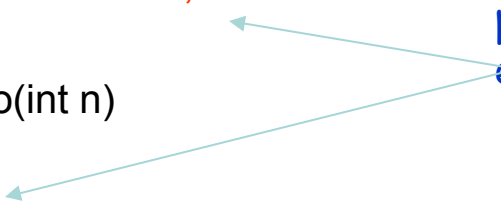
Utilizar campos estáticos:

```
class SimuladorDeCaixaDeBanco
{
    static private int númeroDoCliente = 0;
    private int númeroDoCaixa;

    SimuladorDeCaixaDeBanco(int n)
    {
        númeroDoCaixa = n;
        númeroDoCliente = 0;
        System.out.println("Caixa "+númeroDoCaixa+" iniciou operação.");
    } // fim do construtor

    public void próximoAtendimento()
    {
        númeroDoCliente = númeroDoCliente + 1;
        System.out.print("Cliente com a senha número "+númeroDoCliente+", favor ");
        System.out.println("dirigir-se ao caixa número "+númeroDoCaixa+".");
    }
} // fim da classe SimuladorDeCaixaDeBanco
```

para poder iniciar e atender em qualquer ordem



SimuladorDeCaixaDeBanco.java



# Campos Estáticos

- Simulação no BlueJ



# Campos Estáticos

- Outro uso de campos estáticos:  
declaração de constantes

```
final static public double raizDe2 =  
    1.4142135623730950488;
```





# Criando Aplicações em Java

- Em Java, pode-se criar uma classe com um método especial que será considerado o ponto de entrada de um programa.
- A presença deste método faz com que a classe se torne executável
- O método especial é chamado **main** e deve obrigatoriamente ter:
  - modificadores `public static`;
  - retornar `void`
  - Lista de argumentos formada por um array
- Um ponto central para execução de um programa Java

```
public class Aplicacao{  
    public static void main(String[] s){  
    }  
}
```



# Criando Aplicações em Java

- `public static void main(String[] args)`
  - `public`: é visível para qualquer outra classe
  - `static`: dispensa a criação de objetos
  - `void`: nenhum retorno esperado
  - `main`: convenção para indicar a máquina virtual qual o ponto de entrada da aplicação
  - `String[] args`: parâmetros passados pela aplicação via linha de comando. `args` é o identificador

# Exemplo

```
public class Aplicacao  
{ private Ponto p1,p2;
```

Declaração de objetos da classe Ponto como atributos.

```
public void moverPontos()  
{ p1.mover(4F,2F);  
  p2.mover(-2F,-4F);  
}
```

Utilização do método mover da classe Ponto, através do objeto p1.

```
public void criarPontos()  
{ p1 = new Ponto();  
  p2 = new Ponto();  
}
```

Instância de um objeto da classe Ponto

```
public void rodar()  
{ criarPontos();  
  moverPontos();  
}
```

Método principal, que será executado inicialmente pela máquina virtual Java.

```
public static void main(String[] s)  
{ Aplicacao ap = new Aplicacao();  
  ap.rodar();  
}
```

Criação de um objeto da própria classe



```
public class Ponto
```

```
{  private float x;  
    private float y;
```

Declaração de variáveis do tipo primitivo float como atributos.

```
    public void mover(float novoX, float novoY)
```

```
{    x = novoX;  
        y = novoY;  
    }
```

```
}
```



# Criando Aplicações em Java

## Observações sobre o método `main`:

- A classe que contém o método `main` pode conter outros métodos, que podem ser chamados pelo `main`.
  - Evitar repetição de código
  - Efetuar tarefas particulares

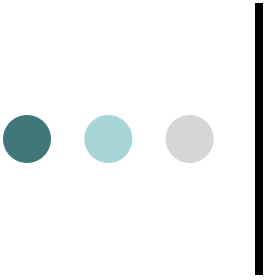
*Métodos chamados a partir do main devem ser obrigatoriamente estáticos!!!*

*Campos da classe acessados pelo main devem ser obrigatoriamente estáticos!!!*



# Métodos estáticos

- Por que o método `main` deve ser declarado como estático?
  - Execução de um programa via linha de comando
    - `Java NomeDaClasse arg1.... argN`
    - Ao executar este comando a JVM invoca o método `main` da classe
      - Neste momento nenhum objeto foi criado
  - Declarar o `main` com `static` permite que a JVM invoque o `main` sem criar uma instância desta classe



- Exemplo
  - Locadora



# Métodos Estáticos

- Funcionam sempre da mesma forma, mesmo quando aplicados à instâncias diferentes.
- Podem ser chamados sem a necessidade de criação de instâncias da classe a qual pertencem.





# Métodos Estáticos

## ○ Uso

- Em classes que tenham o método main
  - Servem como sub rotinas
- Implementação de rotinas que sejam independentes dos dados armazenados na classe
  - Métodos que só necessitam dos dados passados como argumentos
  - Resultado seja independente de qual instância da classe a que pertencem seja usada na sua chamada.



# Métodos Estáticos

- Uso (continuação)
  - Bibliotecas de métodos
    - Classes que só contém métodos estáticos
    - Agrupamento por função



# Métodos Estáticos

- Relembrando....
- Se um método for chamado diretamente do main
  - Deve ser obrigatoriamente declarado como estático
- Se o main for acessar campos declarados na sua classe mas fora do main
  - Campos devem ser declarados com estáticos.



# Métodos Estáticos

- Exemplo

- Uma biblioteca de valores de constantes matemáticas



```
class ConstantesMatematicas // declaração da classe
```

```
{
```

```
/**
```

```
 * Declaração dos campos da classe
```

```
 */
```

```
 // A raiz quadrada de 2
```

```
 final static public double raizDe2 = 1.4142135623730950488;
```

```
 // A raiz quadrada de 3
```

```
 final static public double raizDe3 = 1.7320508075688772935;
```

```
 // A raiz quadrada de 5
```

```
 final static public double raizDe5 = 2.2360679774997896964;
```

```
 // A raiz quadrada de 6: podemos usar as constantes já definidas
```

```
 final static public double raizDe6 = raizDe2*raizDe3;
```

```
 } // fim da classe ConstantesMatematicas
```

```
// Valores obtidos no livro Manual de Fórmulas e Tabelas Matemáticas,  
Murray R.
```

```
// Spiegel, Coleção Schaum, Editora McGraw-Hill
```

```
class DemoConstantesMatematicas // declara a classe
```

```
{
```

```
public static void main(String[] argumentos)
```

```
{
```

```
// Criamos duas instâncias da classe ConstantesMatematicas. Como os campos desta  
// classe são estáticos, os valores são idênticos independentemente das instâncias.
```

```
ConstantesMatematicas const1 = new ConstantesMatematicas();
```

```
ConstantesMatematicas const2 = new ConstantesMatematicas();
```

```
ConstantesMatematicas const3 = null;
```

```
// Verificamos a igualdade...
```

```
System.out.println(const1.raizDe2 == const2.raizDe2); // imprime true
```

```
System.out.println(const1.raizDe3 == const2.raizDe3); // imprime true
```

```
System.out.println(const1.raizDe5 == const2.raizDe5); // imprime true
```

```
System.out.println(const1.raizDe6 == const2.raizDe6); // imprime true
```

```
// É muito mais prático acessar os campos diretamente a partir da classe:
```

```
double raizDe10 = ConstantesMatematicas.raizDe2 * ConstantesMatematicas.raizDe5;
```

```
System.out.println("A raiz quadrada de 10 é "+raizDe10);
```

```
// Acesso a variável mesmo sem que a classe fosse instanciada
```

```
System.out.println("A raiz de dez é "+const3.raizDe2);
```

```
} // fim do método main
```

```
} // fim da classe DemoConstantesMatematicas
```



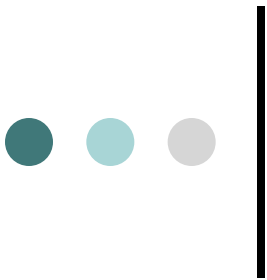
# Métodos Estáticos

- Outro Exemplo
  - Biblioteca de funções para conversão de Unidades de comprimento



ConversaoDeUnidadesDeComprimento





// Criamos uma instância da classe  
ConversaoDeUnidadesDeComprimento. Como a classe  
// não contém campos e os métodos são estáticos,  
não existe real diferença entre  
// chamar os métodos de uma ou outra instância da  
classe.

```
class DemoConversaoDeUnidadesDeComprimento
{
    public static void main(String[] argumentos)
    {
        ConversaoDeUnidadesDeComprimento conv = new ConversaoDeUnidadesDeComprimento();

        System.out.println("vinte pés: "+conv.pésParaCentímetros(20)+" centímetros");
        System.out.println("cinco polegadas;"+conv.polegadasParaCentímetros(5)+" centímetros");

        System.out.println("vinte pés :"+
            ConversaoDeUnidadesDeComprimento.pésParaCentímetros(20)+
            " centímetros");
        System.out.println("cinco polegadas:"+
            ConversaoDeUnidadesDeComprimento.polegadasParaCentímetros(5)+
            " centímetros");
    } // fim do método main
} // fim da classe DemoConversaoDeUnidadesDeComprimento
```

DemoConversaoDeUnidadesDeComprimento



# Métodos Estáticos

- O método `main` pode ser chamado por outros métodos: uma aplicação inteira pode ser parte componente de outra aplicação.



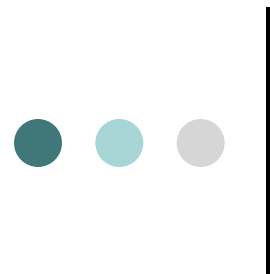
# Chamadas de Métodos

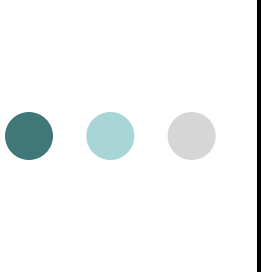
- Próprio nome do método para chamar um método da mesma classe.
- Variável que contém referência a um objeto, seguido de ponto(.) e o nome do método
- Nome de classe, seguido de ponto e o nome do método estático



# Exercício

- Classe para calcular o preço de um terreno baseado em sua área e localização, usando métodos estáticos.
  - Se a localização for 1 (periferia) o preço do m2 é de 22,00
    - 2->27,0
    - 3->29,50
    - 4->31,50
    - 5->34,30

- 
- Escreva uma classe em Java que simule uma calculadora bem simples. Essa classe deve ter como atributos duas variáveis double e um char. Deve possuir um construtor que recebe como parâmetro dois números e um caracter, correspondente a uma das operações básicas (+, -, \*, /).
  - Deve ter um método para calcular a operação desejada e um para imprimir o resultado. O programa deve considerar divisões por zero como sendo erros, e imprimir uma mensagem adequada.

- 
- Durante o período de matrícula de uma faculdade, é disponibilizado um atendente para cada curso de maneira que os alunos efetuam sua matrícula junto ao atendente de seu curso. A matrícula do aluno é efetuada indicando o seu nome e o seu curso. Sempre que um aluno é matriculado é gerado um número de matrícula para ele. Este número é único independente do curso em que ele estude e deve ser apresentado a ele no ato da matrícula.
  - Para o cenário acima, modelar uma classe para atendente e uma classe para aluno, com os métodos necessários para efetuar a matrícula.