Bacharelado em Ciências da Computação

BC1501 – Programação Orientada a Objetos

UFABC

Universidade Federal do ABC

Aula 04

Material adaptado do
Prof. André G. R. Balan

Aula Passada

- Herança e Polimorfismo

```
public abstract class Profissional {
      public String nome;
      public String email;
      protected ArrayList<Profissional> contatos = new ArrayList<Profissional>();

      public void adicionaContato(Profissional profissional) {
        if (profissional == this) return;
        if (contatos.contains(profissional)) return;
        else {
              contatos.add(profissional);
              profissional.adicionaContato(this);
        }
      }
      .
      .
      .
}
```

Aula Passada

- Herança e Polimorfismo

```
public abstract class Profissional {
      .
      .
      .
      public void mostraContatos() {
          System.out.println("Contatos do " +  getClass().getSimpleName() + " " + nome);
          int i =1;
          for (Profissional p: contatos) {
              System.out.println("Contato " + i++ + ": ");
              System.out.println("\t " + p.getClass().getSimpleName());
              System.out.println("\t " + p.nome);
              System.out.println("\t " + p.email);
              System.out.println("");
          }
      }
}
```

Aula Passada

- Herança e Polimorfismo

```
public class Professor extends Profissional {

    public Professor(String nome, String email) {

          this.nome = nome;
          this.email = email;
    }
}
```

Aula Passada

- Herança e Polimorfismo

```
public static void main(String[] args) {
        Profissional [] profissionais  = new Profissional[9];

        profissionais[0] = new Político("Pedro", "pedro@gmail.com");
        profissionais[1] = new Político("Maria", "maria@gmail.com");
        profissionais[2] = new Político("Celso", "celso@gmail.com");

        profissionais[3] = new Professor("André", "andré@gmail.com");
        profissionais[4] = new Professor("Silvia", "silvia@gmail.com");
        profissionais[5] = new Professor("Márcio", "marcio@gmail.com");

        profissionais[6] = new Empresário("Luis", "luis@gmail.com");
        profissionais[7] = new Empresário("Ana", "ana@gmail.com");
        profissionais[8] = new Empresário("Rodrigo", "rofrigo@gmail.com");
        .
        .
        .
}
```

Aula Passada

- Herança e Polimorfismo

```
public static void main(String[] args) {
        .
        .
        .
        Random ran = new Random();

        for (int i = 0;  i< profissionais.length ; i++ ) {
            Profissional p = profissionais[i];
            int n = ran.nextInt(5)+1;
            for (int j=0; j<n; j++ )
                p.adicionaContato(profissionais[ran.nextInt(9)]);
        }
        for (Profissional p : profissionais)
             p.mostraContatos();
}
```

Nesta aula

- **Campos e Métodos Estáticos**

Campos Estáticos

- Campos/Atributos estáticos também são conhecidos como **campos de classes.**

- São campos declarados com o modificador *static*

```
public class  A {
    public static double  Teste;
}
```

- A alocação de memória para um campo estático se dá no momentos em que o programa começa a rodar.

- Diferentemente, a alocação de memória para um campo normal só ocorre quando se cria um objeto.

Campos Estáticos

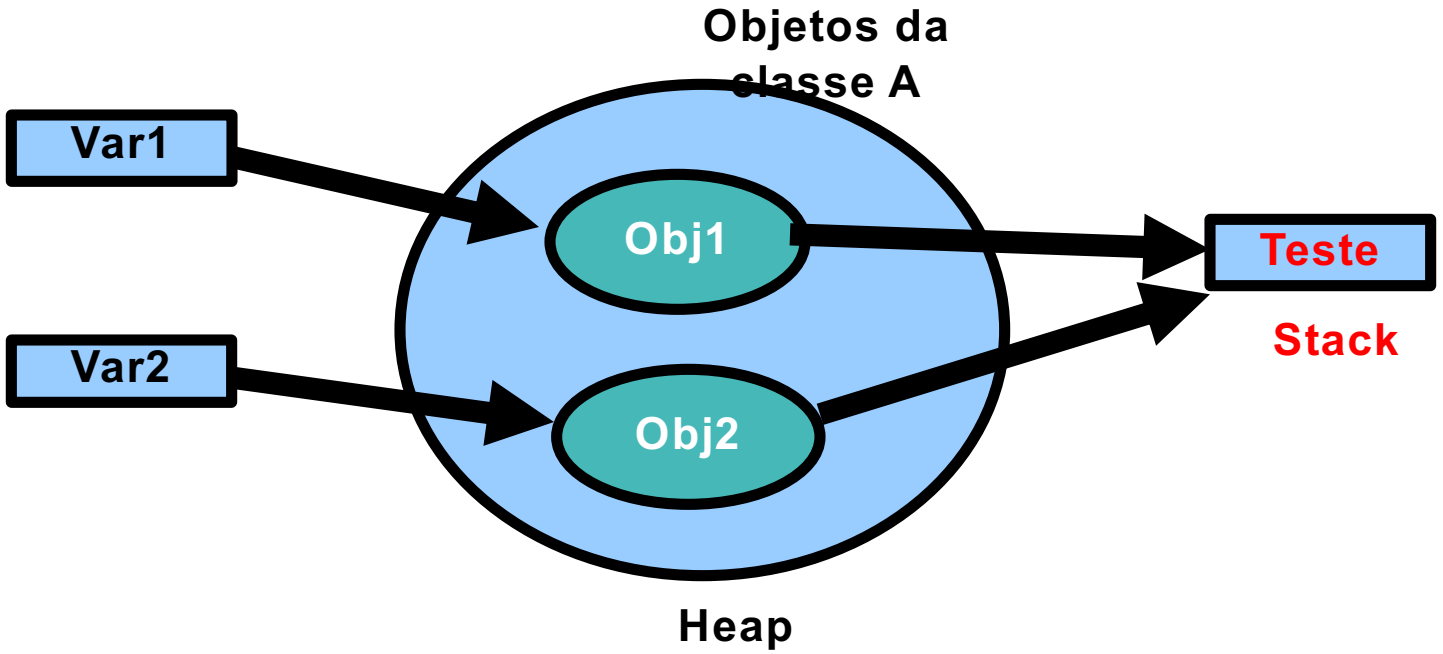
- Deste modo, os *campos estáticos podem ser acessados diretamente usando o nome da classe, sem que seja necessária a criação de uma* instância da classe

```
A.teste = 10.0;
```

- Mas também é possível acessar os atributos estáticos por meio dos objetos da classe onde o atributo é definido.
  - Ex:
    - `A obj1 = new A();`
    - `obj2.teste = 20.0;`

Campos Estáticos

- Os objetos da classe compartilharão as porções de memória referentes aos campos estáticos



Campos Estáticos

- *Em outras palavras, somente um valor será armazenado em um campo estático, e caso este valor seja* modificado por uma das instâncias desta classe, a modificação será refletida em todas as outras instâncias desta classe.

- Ex:

```
obj1.Teste = 10.0;
obj2.Teste = 100.0;
A obj3 = new A();
System.out.print(obj3.Teste);
```

Qual o valor impresso?

Campos Estáticos

- **Exemplo 1** – Contador do número de objetos de uma classe

```
public class Mesa {
    public static int count = 0;


    public Mesa() {
            count++;
            . . .
    }
}
```

Campos Estáticos

- **Exemplo 1** – Contador do número de objetos de uma classe

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

    Mesa m1 = new Mesa();
    Mesa m2 = new Mesa();
    Mesa m3 = new Mesa();

    System.out.print( Mesa.count);
}
```

Campos Estáticos

- **Exemplo 2** – Criar um identificar único para cada objeto

```
public class Mesa {
    public static int count = 0;
    public int id;


    public Mesa() {
            count++;
            id = count;
            . . .
    }
}
```

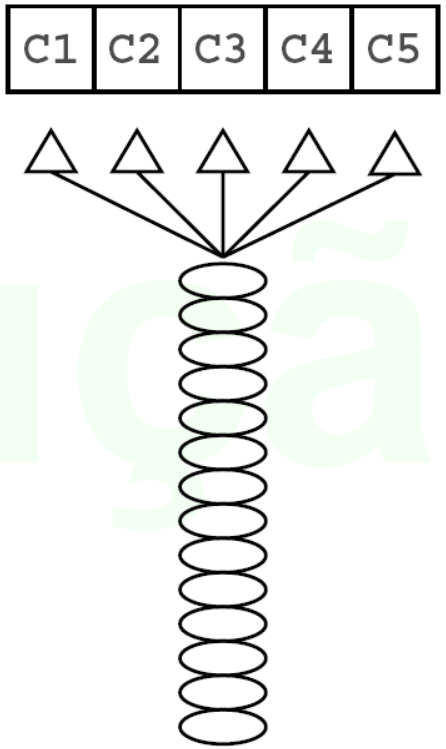
Campos Estáticos

- **Exemplo 2** – Criar um identificar único para cada objeto

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

    Mesa m1 = new Mesa();
    Mesa m2 = new Mesa();
    Mesa m3 = new Mesa();

    System.out.print( m1.id);
    System.out.print( m2.id);
    System.out.print( m3.id);

}
```

Campos Estáticos

- **Exemplo 3** – Criar um gerenciador de filas simples



Campos Estáticos

- **Exemplo 3** – Criar um gerenciador de filas simples

```
public class Atendente {
    public static int proximo = 0;
    public int numero;

    public Atendente(int numero) {
          this.numero = numero;
    }

    public chamaPróximo() {
       System.out.println("Senha " + proximo);
       proximo++;
       System.out.println("Dirija-se ao atendente " + numero);
    }
}
```

Campos Estáticos

- **Exemplo 3** – Criar um gerenciador de filas simples

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

    Atendente a1 = new Atendente(1);
    Atendente a2 = new Atendente(2);

    a1.chamarPróximo();
    a2.chamarPróximo();
    a1.chamarPróximo();
}
```

Campos Estáticos

- *Outra utilização para campos estáticos:*
  - *Armazenar valores constantes*

```
class ConstantesMatematicas   {
   final static public double pi     =  3.14159265358979323846264338327950;
   final static public double raizDe2 = 1.41421356237309504880;
   final static public double raizDe3 = 1.73205080756887729352;
   final static public double raizDe5 = 2.23606797749978969640;
}
```

Campos Estáticos

- O modificador **final** faz com que os valores dos campos, depois de declarados, não possam mais ser modificados
  - o que é desejável para campos que representam **constantes**

- *Exemplo de utilização da classe:*

```
area =  ConstantesMatematicas.pi * raio * raio;
```

Observe que não é preciso criar a instância da classe, embora seja possível

Métodos Estáticos

- Métodos estáticos são aqueles que também são declarados com o modificador static:

- ***Assim como os campos estáticos, os métodos estáticos podem ser acessados a partir do nome da classe, sem que seja necessário criar um objeto daquela classe***

- *Qual a principal aplicação para tais métodos?*

Métodos Estáticos

- Principal aplicação:
  - *Criar bibliotecas de métodos*

```
class Conversao {
        public static double polegadasParaCentímetros(double polegadas) {
            double centímetros = polegadas*2.54;
            return centímetros
        }

        public static double pésParaCentímetros(double pés)  {
            double centímetros = pés*30.48;
            return centímetros;
        }
}
```

Métodos Estáticos

- Principal aplicação:
  - *Criar bibliotecas de métodos*

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

    double cm =  Conversao.polegadasParaCentímetros(3);
    System.out.println(cm);

}
```

Campos Estáticos

- A palavra reservada **final** aplica-se também na declaração de classes. Exemplo

```
public final class Teste  {

}
```

Neste caso, não é mais possível criar sub-classes para a classes **Teste**

A tentativa de criar uma classe herdeira irá gerar um erro de sintaxe (compilação)

Métodos Estáticos

- Exemplos:
  - *Uma bibliotecas de métodos de ordenação*
    - *Quicksort*
    - *MergeSort*
    - *HeapSort*
  - *A classe Math do pacote java.lang*
    - *Constantes : PI*
    - *Métodos estáticos: sin, cos, log, abs, ...*

Métodos Estáticos

- Observação:
  - *Dentro de métodos estáticos não é possível acessar campos e métodos que não sejam estáticos*

```
public class A  {
    int numero;

    public static void teste () {
        numero = 10;  //  Gera um erro de sintaxe
    }
}
```

Erro: ***non-static variable teste cannot be referenced from a static context.***

Porque??

Java Import Static

- Uma declaração **import static** permite que os programadores referenciem membros estáticos (campos e métodos), como se eles fosses declarados na própria classe que os utiliza.

- *Exemplo: importando todos os campos estáticos da classe Math*

```
import static java.lang.Math.* ;
.
.
.
double area = raio * raio * PI;
int teste = abs(-10);
```

Métodos Estáticos

- Exercício 1

Programar e testar os três exemplos envolvendo campos estáticos:

- Contador de objetos da classe
- Gerador de identificação de objetos
- Gerenciador de filas (simulador simples)
  - O programa deve ter um vetor de 10 objetos Atendentes
  - Faça um loop, fazendo com que, em cada iteração, aleatoriamente, um dos atendentes chame o próximo da fila
  - O loop deve parar quando a senha de número 300 for alcançada

Métodos Estáticos

- Exercício 2

Escreva uma classe ConversaoDeTemperatura que contenha métodos estáticos para calcular a conversão entre diferentes escalas de temperatura. Considere as fórmulas de conversão abaixo:

Celsius (*C) para Fahrenheit (F):*	*F = (9.0/5.0)*C + 32*
Celsius (*C) para Kelvin (K):*	*K = C + 273,15*
Fahrenheit (*F) para Celsius (C):*	*C = (5.0/9.0)*(F – 32)*
Kelvin (*K) para Celsius (C):*	*C = K- 273,15*
Fahrenheit (*F) para Kelvin (K):*	*K = (5.0/9.0) (F + 459,67)*
Kelvin (*K) para Fahrenheit (F):*	*F = (9.0/5.0)*K – 459,67*

Faça um programa para testar

Métodos Estáticos

- Exercício 3

- Escreva uma classe que contenha métodos estáticos para retornar o maior de dois, três, e quatro valores, considerando que os argumentos e retorno dos métodos podem ser dos tipos int e double.

- *Dica: os métodos podem ser chamados em cascata: para calcular o maior de três* valores *a, b e c, pode-se calcular o maior valor de a e b, e comparar este resultado com c.*

- Faça um programa para testar