Membros *static* e *abstract*

POO

Prof. Márcio Delamaro

Membro static

- Um membro (atributo ou método) *static* está associado com a classe, e não com os objetos
- Quando instanciamos um objeto da classe, esse membro não está no objeto
- Como o membro está na classe, existe apenas uma instância desse membro
- Exemplo da classe EntradaTeclado

EntradaTeclado

```
public static int leInt() throws IOException,
NumberFormatException  {
     String x = leString();
     return Integer.parseInt(x);
}
```

EntradaTeclado

```
public static int leInt() throws IOException,
NumberFormatException  {
     String x = leString();
     return Integer.parseInt(x);
}

static public void main(String[] args) {
    EntradaTeclado et = new EntradaTeclado();
    int k = et.leInt();
}
```

EntradaTeclado

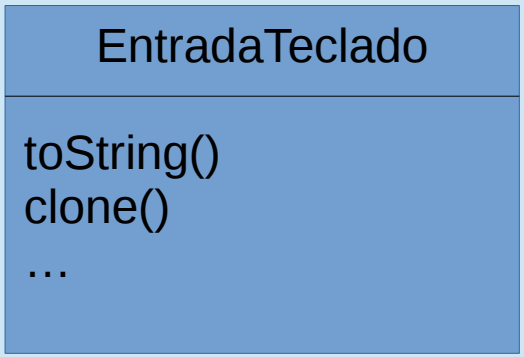
```
public static int leInt() throws IOException,
NumberFormatException  {
     String x = leString();
     return Integer.parseInt(x);
}

static public void main(String[] args) {
    EntradaTeclado et = new EntradaTeclado();
    int k = et.leInt();
}
```

Membro static deve ser acessado de forma estática.

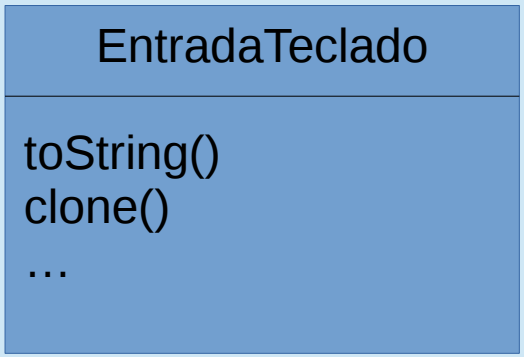
Classe X objeto

et = new EntradaTeclado();



Classe X objeto

et = new EntradaTeclado();



Criação dinâmica.

String s = et.toString();

Classe X objeto

et = new EntradaTeclado();

EntradaTeclado
toString() clone() …

Criação dinâmica.

String s = et.toString();

EntradaTeclado (classe)
leInt() leString() leDouble()

Criação estática.

String s = EntradaTeclado.leString();

Regras

- Não existe um objeto para acessar os membros static
- Por isso devemos usar o nome da classe
- Membro dinâmico pode acessar membro estático
- Membro estático não pode acessar membro dinâmico

Outro exemplo

- Vamos tomar as classes PessoaXXXX
- Nós queremos ter controle de quantos objetos são instanciados
- Sugestões?

Outro exemplo

- Vamos tomar as classes PessoaXXXX
- Nós queremos ter controle de quantos objetos são instanciados
- Vamos criar uma variável estática que é incrementada cada vez que um objeto é instanciado
- Vamos criar um método para recuperar o valor dessa variável

Classe Pessoa

```
public class Pessoa {
   static private int nroPessoas = 0;

   static public int getNroPessoas() {
      return nroPessoas;
   }
```

Que mais?

Classe Pessoa

```
public class Pessoa {
   static private int nroPessoas = 0;

   static public int getNroPessoas() {
      return nroPessoas;
   }

  public Pessoa(double peso, double altura, int idade){
    this.peso = peso;
    this.altura = altura;
    this.idade = idade;
    nroPessoas++;
}
```

Classe Pessoa

```
public class Pessoa {
   static private int nroPessoas = 0;

   static public int getNroPessoas() {
      return nroPessoas;
   }

  public Pessoa(double peso, double altura, int idade){
    this.peso = peso;
    this.altura = altura;
    this.idade = idade;
    nroPessoas++;
}
```

```
PessoaMulher pm = new PessoaMulher(51, 1.68, 35);
PessoaHomem ph = new PessoaHomem(88, 1.8, 17);
System.out.println("Objetos: " + Pessoa.getNroPessoas());
```

Resumindo

- Com isso contamos quantos objetos de qualquer uma das três classes são instanciados
- Criamos um controle “global”
- Podemos fazer o mesmo para as subclasses *PessoaHomem* e *PessoaMulher*

Mas...

- Como são instanciados os objetos dessas 3 classes?

Mas...

- Como são instanciados os objetos dessas 3 classes?
- Elas são instanciadas sempre a partir das subclasses
- Não faz sentido instanciar um objeto da classe *Pessoa* diretamente

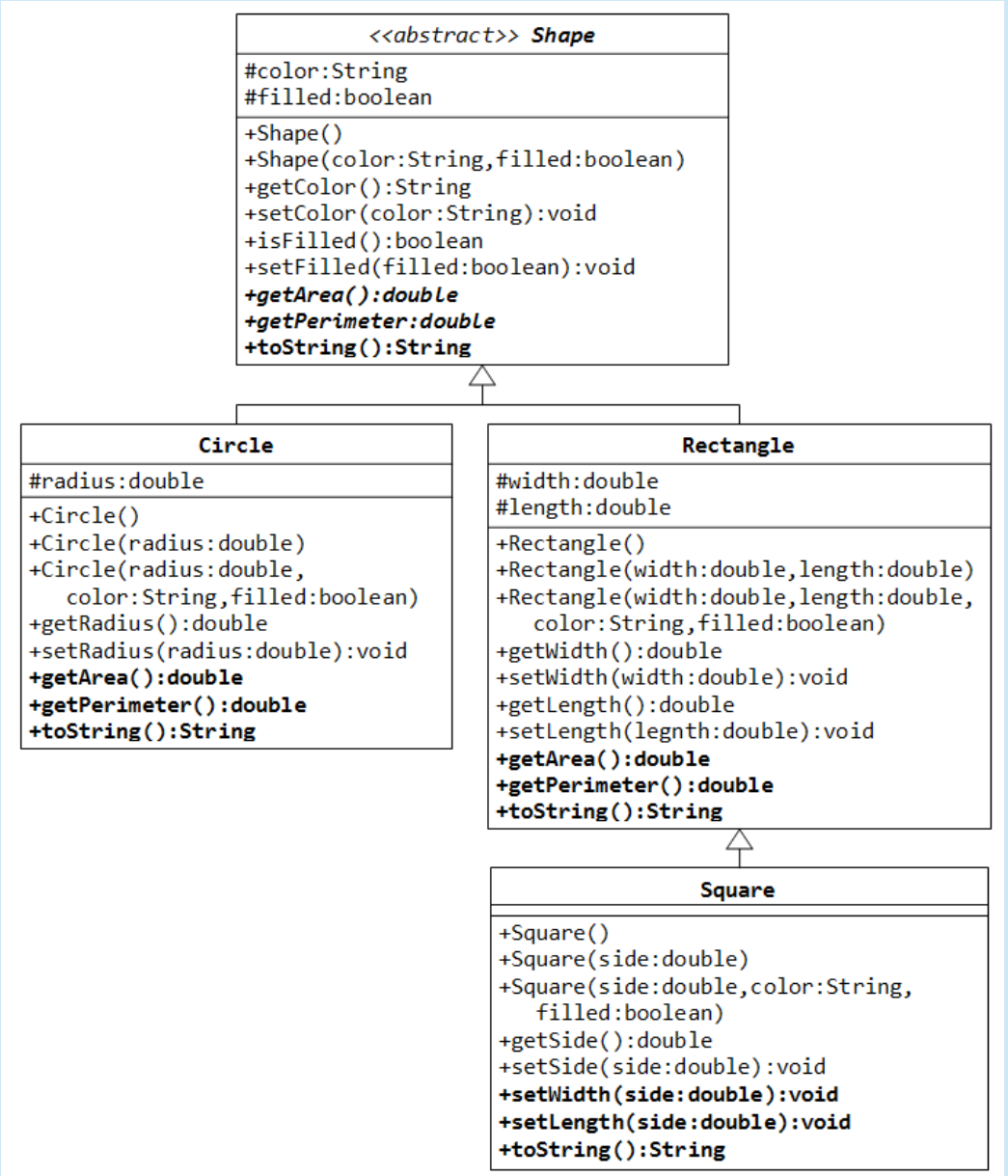
Mas...

- Como são instanciados os objetos dessas 3 classes?
- Elas são instanciadas sempre a partir das subclasses
- Não faz sentido instanciar um objeto da classe *Pessoa* diretamente
- Como solucionar isso?

Classe abstract

- Uma classe abstract não pode ser instanciada
- Ela serve apenas como classe base
- A partir dela, espera-se que outras classes sejam herdeiras

Figuras geométricas



Declaração abstract

```
/**
 * Descreve atributos de uma pessoa e permite
 * calcular algumas medidas como IMC
 * @author delamaro
 */
public abstract class Pessoa {
```

Instanciação

```
static public void main(String args[]) {
      PessoaMulher pm = new PessoaMulher(51, 1.68, 35);
      PessoaHomem ph = new PessoaHomem(88, 1.8, 17);
      Pessoa p = new Pessoa(88, 1.8, 17);
      System.out.println("Objetos: " +
                         Pessoa.getNroPessoas());

   }
```

Instanciação

```
static public void main(String args[]) {
      PessoaMulher pm = new PessoaMulher(51, 1.68, 35);
      PessoaHomem ph = new PessoaHomem(88, 1.8, 17);
      Pessoa p = new Pessoa(88, 1.8, 17);
      System.out.println("Objetos: " +
                         Pessoa.getNroPessoas());

   }
```

Cannot instantiate the type Pessoa

Podemos fazer ainda melhor

- Usando uma classe abstrata, podemos obrigar as classes filhas a implementar alguns métodos
- Esses métodos não são implementados na classe base
- Mas têm que ser implementados nas classes filhas
- São métodos declarados abstract

Método abstract

- É o caso, por exemplo de *classificaObesidade*
- *A declaração abaixo, obriga as classes PessoaHome e Pessoa Mulher a implementar o método*

```
public abstract class Pessoa {

    public abstrat String classificaObesidade();

}
```

Método abstract

- É o caso, por exemplo de *classificaObesidade*
- *A declaração abaixo, obriga as classes PessoaHome e Pessoa Mulher a implementar o método*

```
public abstract class Pessoa {

    public abstrat String classificaObesidade();

}
```

The type PessoaHomem must implement the inherited abstract method Pessoa.classificaObesidade()

Voltando ao polimorfismo

- Uma grande vantagem de declarar métodos abstract é em relação ao polimorfismo dos objetos
- A classe abstrata passa a “ter” os membros declarados como abstratos
- Dessa forma, um objeto declarado estaticamente como Pessoa, agora tem um método *classificaObesidade()*

Polimorfismo

```
PessoaMulher pm = new PessoaMulher(51, 1.68, 35);
PessoaHomem ph = new PessoaHomem(88, 1.8, 17);
Pessoa p = pm; p.classificaObesidade();
p = ph; p.classificaObesidade();
```

Polimorfismo

```
PessoaMulher pm = new PessoaMulher(51, 1.68, 35);
PessoaHomem ph = new PessoaHomem(88, 1.8, 17);
Pessoa p = pm; p.classificaObesidade();
p = ph; p.classificaObesidade();
```

Como p é declarado como Pessoa, não faria sentido chamar classificaObesidade() se essa classe nao tivesse esse método.

Embora não tenha uma implementação, tem um “protótipo” que garante que o objeto que está na variável p vai ter uma implementação do método.

Exercício

- Altere a classe ContaBancaria, incluindo um atributo estático que controle o número da próxima conta a ser criada. Desta forma não é necessário fornecer o número da conta na criação do objeto. Esse número deve ser obtido desse atributo.
- Transforme o método atualiza em um método abstract da classe ContaBancaria. A implementação na classe ContaEspecial, simplesmente não atualiza o saldo

Exercício

- Crie duas subclasses PoupancaOuro e PoupançaSimples. A poupança ouro paga rendimento mensal de 1.5 * taxa
- Transforme a classe ContaPoupanca em abstrata, já que ela não poderá ser instanciada diretamente