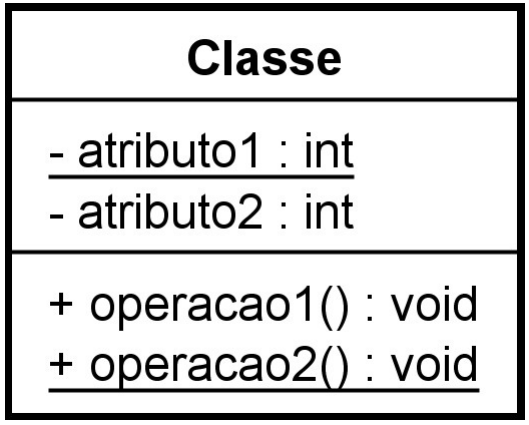
Membros de classe

Membros de classe

- São membros (variáveis ou métodos) que pertencem à classe. Não pertencem a nenhuma instância em particular.
- Membros de classe podem ser utilizados sem que haja uma instância da classe
- No diagrama de classes, os membros de classe são sublinhados.



Variáveis de classe

- Variável de classe:
  - Uma variável que é comum à todas as instâncias
    - Uma variável que é compartilhada entre todas as instâncias
  - Podem ser manipulados sem que haja uma instância
    - Para acessar utilizar a sintaxe: `classe.identificador`
  - Também chamados de “variáveis estáticas” ou “campos de classe”.
- Sintaxe:

```
Modificador static tipo de dado identificador;
```

- Exemplo:

```
private static int atributo1;
```

Métodos de classe

- Métodos de classe:
  - Podem manipular variáveis de classe;
  - Não podem manipular variáveis de instância sem que haja uma instância explicita;
  - Não podem reusar métodos de instância;
  - Não podem utilizar a palavra `this`.
- Sintaxe:

```
Modificador static tipo de dado identificador(parametros);
```

Exemplos de membros estáticos

- `Integer.MAX_VALUE`
- `Math.sqrt()`
- `Math.abs()`
- `Math.max()`
- `JOptionPane.showInputDialog()`
- `JOptionPane.showMessageDialog()`

Sobrecarga de métodos

FURB
UNIVERSIDADE DE BLUMENAU

Sobrecarga de métodos

- A linguagem Java suporta a sobrecarga de métodos, isto é, implementação de vários métodos com mesmo nome.
- Os métodos devem ter assinaturas diferentes
  - Métodos podem ter mesmo nome se a lista de parâmetros for diferente.
  - O compilador não considera o tipo de retorno para diferenciar o método. Por isso, dois métodos com a mesma assinatura mas retornos distintos não podem ser implementados na mesma classe
- Deve ser utilizado com moderação pois pode tornar o código menos legível

Construtores

Construtores

- São similares à métodos, com exceção de que são invocados exclusivamente durante a criação de objetos
- Geralmente utilizados para “inicializar” um objeto
- A declaração de construtor é semelhante à declaração de métodos, porém não possuem tipo de dado de retorno e seu identificador é igual ao da classe
- Não é preciso criar construtor para a classe. Quando não é implementado um construtor, o compilador automaticamente fornece um construtor padrão.
  - Um “construtor padrão” é um construtor sem argumentos.

O operador new

- O operador **new** faz então quatro operações:
  1. Cria o objeto na memória, alocando espaço para armazenar valores para suas variáveis de instância
  2. Inicializa as variáveis de instância
  3. **Executa o construtor que foi utilizado no operador new**
  4. Retorna o endereço de memória criado pelo objeto.

Exceções

FURB
UNIVERSIDADE DE BLUMENAU

Exceções

- Uma exceção é um evento, que ocorre durante a execução do programa, que interrompe o fluxo normal de execução.
- Quando uma operação incorreta é identificada dentro de um método, o método pode criar um objeto de uma classe que caracteriza o erro e notificá-lo ao sistema
  - Este objeto é denominado de “objeto de exceção”
  - O objeto de exceção contém informação sobre o erro
  - Esta operação (criar objeto e notificar o sistema) é conhecido como “lançamento de exceção”
- O efeito de uma exceção lançada é (por enquanto) abortar a execução do programa
  - Não poderá existir comandos, no mesmo método, após a instrução que lança a exceção

Lançamento de exceções em Java

- Sintaxe:

```
throw new RuntimeException("mensagem");
```

- Onde:
  - Mensagem: indica uma mensagem que pode ser apresentada quando a exceção for gerada
- Exemplo:

```
public void setSalario(double novoSalario) {
    if (novoSalario < 0) {
        throw new RuntimeException("Salário incorreto");
    }
    salario = novoSalario;
}
```