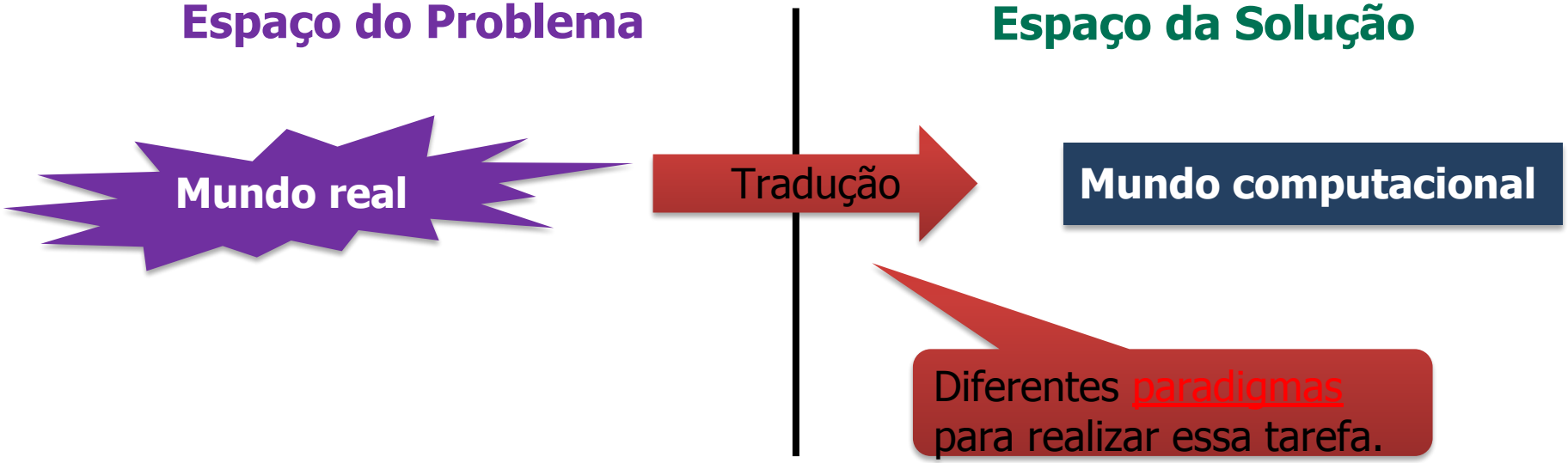
Orientação a Objetos com Java

2ª aula

Prof. Douglas Oliveira
douglas.oliveira@prof.infnet.edu.br

Orientação à Objetos



De forma simplificada, podemos dizer que desenvolver um sistema é criar uma solução computacional para um problema do mundo real.

Orientação à Objetos

Estruturado

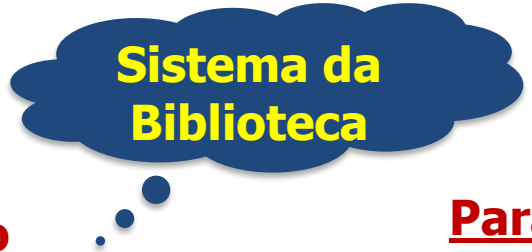
- ✔ Os sistemas são criados a partir da decomposição funcional, ou seja, o problema a ser resolvido é decomposto em problemas menores, que por sua vez podem ser implementados como rotinas ou procedimentos. Além disso, os dados e as funções que manipulam esses dados são tratados separadamente.

Orientado a Objetos

- ✔ Pressupõe que o mundo é composto por objetos, que integram dados e funções. Os sistemas são criados a partir dos objetos que existem no domínio do problema, isto é, os sistemas são modelados como um conjunto de objetos que interagem entre si.

Orientação à Objetos

- Paradigma Estruturado x Paradigma Orientado a Objetos



Paradigma Estruturado

Decomposição por funções

- Registrar empréstimos
- Calcular multa
- Administrar livros
- Cadastrar alunos

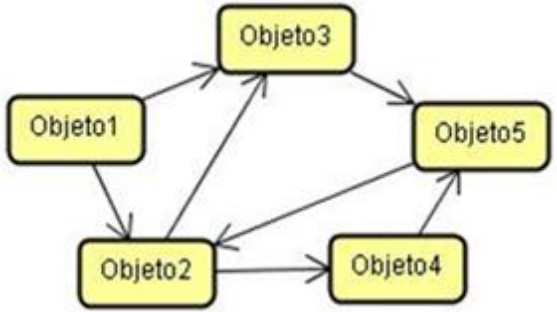
Paradigma OO

Decomposição por objetos do domínio

- Livro
- Periódico
- Aluno
- Empréstimo

Orientação à Objetos

- A Orientação a Objeto é um paradigma de análise, projeto e programação baseado na interação entre diversas unidades de software chamadas objetos.

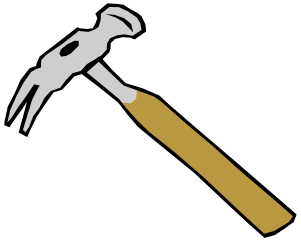


- Mas o que é um objeto ?

Um objeto é a representação computacional de um elemento ou processo do mundo real.

Exemplos de Objetos

martelo *carro* *piloto* *casa*
cliente *disciplina* *aluno* *computador*



Note que os objetos possuem
características e *comportamentos*

Características dos Objetos

- Cada característica ou propriedade é chamada de atributo do objeto.

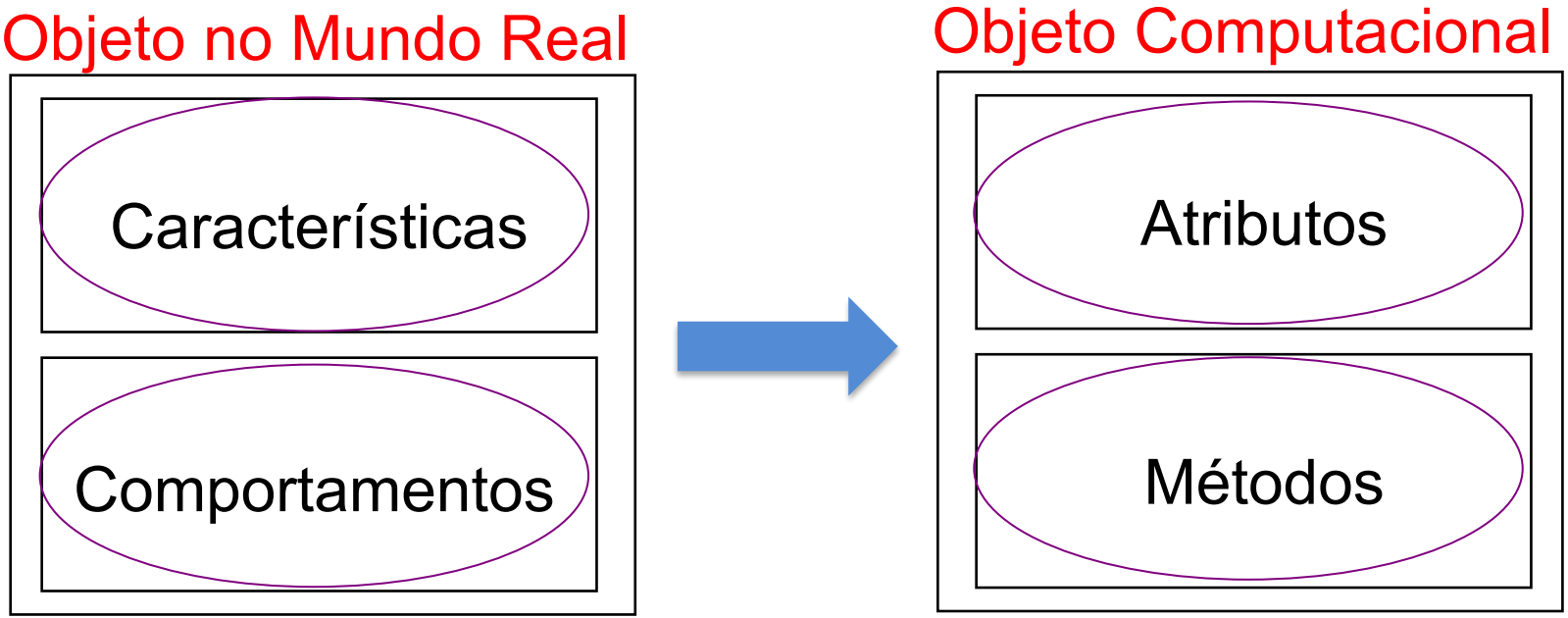
- Exemplo: objeto carro
  - ✔ Cor
  - ✔ Marca
  - ✔ Ano de fabricação
  - ✔ Tipo de combustível
  - ✔ ...

- Podemos sempre associar um conjunto de valores aos atributos.
- Por exemplo: a cor do carro pode ser vermelha, azul, verde, etc.

Comportamento dos Objetos

- Cada comportamento é chamado de método do objeto.
- Um comportamento representa uma reação ou resposta de um objeto a uma ação do mundo real.
- Exemplos de comportamentos para o objeto carro:
  - ✔ Acelerar
  - ✔ Frear
  - ✔ Ligar farol
  - ✔ Desligar farol
  - ✔ Estacionar
  - ✔ ...

Mapeamento de Objetos



Principais Características de OO

- Classificação
- Abstração
- Encapsulamento
- Relacionamentos
- Herança
- Polimorfismo

Classificação

- Classificação

  - ✔ Objetos com as mesmas características e comportamentos são agrupados em uma **classe**.

  - ✔ Cada classe define um conjunto infinito de objetos.

Classificação

- Exemplo: imagine os alunos de uma instituição de ensino. Que características você consegue observar nesses alunos?

  - ✔ Nome: João, Maria, André, Ana, Felipe, Cátia, ...
  - ✔ CPF: 823745967-80, 674124745-56, 278452872-78, ...
  - ✔ Data de nascimento: 01/03/1997, 25/06/1995, ...
  - ✔ Altura: 1.85, 1.64, 1.78, 1.58, 1.87, ...
  - ✔ ...

Classificação

- Exemplo: imagine os alunos de uma instituição de ensino. Que comportamentos você consegue observar nesses alunos?

  - ✔ Inscrever em disciplina
  - ✔ Consultar nota
  - ✔ Consultar frequência
  - ✔ ...

Classificação

Objetos Aluno

abstrair em

Classe Aluno

Aluno
Nome CPF Data de nascimento
Inscrever em disciplina Consultar nota Consultar frequência

Classificação

- Podemos dizer que a classe descreve os atributos e métodos de um conjunto de objetos:
  - ✔ A classe é um molde a partir do qual todos os objetos daquela classe são criados.
  - ✔ Cada objeto pertence a uma única classe.
  - ✔ A classe é o bloco básico para a construção de programas OO.

Classificação

- Quando criamos um objeto a partir de uma classe dizemos que temos uma instância da classe.

- Por ser algo concreto, um objeto possui valores associados aos atributos da classe a qual ele pertence.

Classificação

Classe: Aluno

Aluno
Nome CPF Data de nascimento
Inscrever em disciplina Consultar nota Consultar frequência

Instância ou Objeto: A1

Nome: João Oliveira
CPF: 254.387.562-76
Data de nascimento: 10/09/1989

Instância ou Objeto: A2

Nome: Maria Oliveira
CPF: 945.753.519-03
Data de nascimento: 07/15/1997

Classificação

- Criação e destruição de objetos de uma classe:
  - ✔ A classe é responsável pela criação e destruição de seus próprios objetos.
  - ✔ A criação de um objeto é realizada através de um método especial chamado **construtor**.
  - ✔ A destruição de um objeto é realizada através de um método especial chamado **destrutor**.

Principais Características de OO

- Classificação ✔
- Abstração
- Encapsulamento
- Relacionamentos
- Herança
- Polimorfismo

Abstração

- Abstração

  - ✔ Procura focar as características e comportamentos essenciais de um objeto, de acordo com a perspectiva de quem observa esse objeto e do contexto onde ele será usado.

Abstração

- Objeto: Aluno
- Quais dessas características são relevantes para um Sistema de Controle Acadêmico?
  - ✔ Nome: João, Maria, André, Ana, Felipe, Cátia, ...
  - ✔ CPF: 823745967-80, 674124745-56, 278452872-78, ...
  - ✔ Data de nascimento: 01/03/1997, 25/06/1995, ...
  - ✔ Matrícula: 2016145234, 2015264532, 2015285631, ...
  - ✔ Sexo: feminino, masculino, ...
  - ✔ Altura: 1.85, 1.64, 1.78, 1.58, 1.87, ...
  - ✔ Cor dos olhos: preto, castanho, azul, verde, ...

Abstração

- Objeto: Aluno
- Quais dessas características são relevantes para um Sistema de Controle Acadêmico?
  - ✔ Nome: João, Maria, André, Ana, Felipe, Cátia, ...
  - ✔ CPF: 823745967-80, 674124745-56, 278452872-78, ...
  - ✔ Data de nascimento: 01/03/1997, 25/06/1995, ...
  - ✔ Matrícula: 2016145234, 2015264532, 2015285631, ...
  - ✔ Sexo: feminino, masculino, ...
  - ~~✔ Altura: 1.85, 1.64, 1.78, 1.58, 1.87, ...~~
  - ~~✔ Cor dos olhos: preto, castanho, azul, verde, ...~~

Para definir que características são relevantes é preciso definir o **contexto** onde o objeto será usado.

Abstração

- Objeto: Carro
- Quais dessas características são relevantes para um Sistema de Controle de Fabricação de Carros?
  - ✔ Número do chassi
  - ✔ Placa
  - ✔ Kilometragem
  - ✔ Lote de fabricação
  - ✔ Data da fabricação
  - ✔ Modelo
  - ✔ Cor
  - ✔ Tipo de combustível

Abstração

- Objeto: Carro
- Quais dessas características são relevantes para um Sistema de Controle de Fábricação de Carros?
  - ✔ Número do chassi
  - ✔ ~~Placa~~
  - ✔ ~~Kilometragem~~
  - ✔ Lote de fabricação
  - ✔ Data da fabricação
  - ✔ Modelo
  - ✔ Cor
  - ✔ Tipo de combustível

Abstração

- Objeto: Carro
- Quais dessas características são relevantes para um Sistema de uma Loja de Venda de Carros Usados?
  - ✔ Número do chassi
  - ✔ Placa
  - ✔ Kilometragem
  - ✔ Lote de fabricação
  - ✔ Data da fabricação
  - ✔ Modelo
  - ✔ Cor
  - ✔ Tipo de combustível

Abstração

- Objeto: Carro
- Quais dessas características são relevantes para um Sistema de uma Loja de Venda de Carros Usados?
  - ✔ Número do chassi
  - ✔ Placa
  - ✔ Kilometragem
  - ✔ ~~Lote de fabricação~~
  - ✔ ~~Data da fabricação~~
  - ✔ Modelo
  - ✔ Cor
  - ✔ Tipo de combustível

Classes

- As classes são os elementos básicos para construção de programas OO.
- Assim, para construir um programa OO devemos, inicialmente, identificar as classes que fazem parte do problema que estamos tentando resolver.
- Algumas classes são identificadas de forma bastante natural, enquanto outras requerem uma certa experiência e a utilização de algumas técnicas por parte do desenvolvedor.

Classes

- Imagine que desejamos implementar um editor gráfico que irá manipular figuras geométricas em um plano cartesiano:
  - ✔ Retângulo
  - ✔ Círculo
  - ✔ Triângulo
- Quais as classes candidatas para esse projeto ?

Classes

- Imagine que desejamos implementar um editor gráfico que irá manipular figuras geométricas em um plano cartesiano:
  - ✔ Retângulo
  - ✔ Círculo
  - ✔ Triângulo
- Quais as classes candidatas para esse projeto ?
  - ✔ Retângulo
  - ✔ Círculo
  - ✔ Triângulo
  - ✔ Plano Cartesiano
  - ✔

Classes

- Tomando por base a implementação da classe Retângulo:
  - ✔ Quais as características de um retângulo?
  - ✔ Que operações gostaríamos de realizar com o retângulo ?

Classes

- Tomando por base a implementação da classe Retângulo:
  - ✔ Quais as características de um retângulo?
    - Altura, largura, posição (x, y) no plano, cor da borda, cor de preenchimento, etc.
  - ✔ Que operações gostaríamos de realizar com o retângulo ?
    - Desenhar, mover, rodar, redimensionar, etc.
- Vamos começar, então, definindo a classe Retangulo.

Classes

- Para criar uma classe precisamos definir o lugar onde iremos colocar essa classe.
- No Java, esse lugar é chamado de pacote.
- Podemos imaginar o pacote como sendo um diretório ou uma pasta onde colocamos a classe.
- Podemos definir quantos pacotes quisermos.
- O nome dos pacotes é definido pelos próprios desenvolvedores.
- De acordo com a convenção adotada no Java, os nomes dos pacotes devem ter somente letras minúsculas.

Classes

- Da mesma forma que podemos ter uma hierarquia de pastas, também podemos ter uma hierarquia de pacotes.
- Quando temos um pacote dentro de outro usamos o ponto (.) para definir a hierarquia de pacotes

Classes

- Sintaxe da definição de Classe:

```
[visibilidade] [abstract | final] class identificador
                                        [extends identificador2]
                                        [implements identificador3*] {
   [Atributos]
   [Métodos]
}
```

- ✔ visibilidade: public, private ou package.
- ✔ abstract: não podem ser criadas instâncias da classe (será vista posteriormente).
- ✔ final: não pode ter subclasses (será vista posteriormente).

Classes

- Visibilidade da classe:

  - ✔ **public**: é visível por qualquer outra classe em qualquer pacote.
  - ✔ **private**: só é visível no arquivo onde foi criada.
  - ✔ **package**: só é visível por outras classes do mesmo pacote. É a visibilidade *default* (não precisa escrever package).

- Versão inicial da classe Retangulo:

```
package principal;

public class Retangulo {

}
```

Preferencialmente, toda classe deve ser declarada dentro de um pacote

Por ser pública, a classe Retangulo é visualizada por todas as demais classes do sistema.

Atributos

- Atributos são declarados da mesma forma que as variáveis locais, porém com uma sintaxe ligeiramente diferente:

[*visibilidade*] [static] [final] *tipo atributo* [= *inicialização*];

- ✔ visibilidade: public, private, package ou protected.
- ✔ static: define que esse atributo é da classe (será visto posteriormente)
- ✔ final: atributo constante (valor inicial não pode ser alterado)

Atributos

- Visibilidade dos atributos:
  - ✔ **public**: pode ser livremente lido ou alterado por qualquer classe.
  - ✔ **private**: só pode ser lido ou alterado na própria classe, ou seja, esse atributo não é visível fora da classe.
  - ✔ **package**: só pode ser lido ou alterado pelas classes do mesmo pacote. É a visibilidade *default* (não precisa escrever package).
  - ✔ **protected**: só pode ser lido ou alterado pelas classes descendentes. Será visto com mais detalhes quando falarmos de herança.

Principais Características de OO

- Classificação ✔
- Abstração ✔
- Encapsulamento
- Relacionamentos
- Herança
- Polimorfismo

Encapsulamento

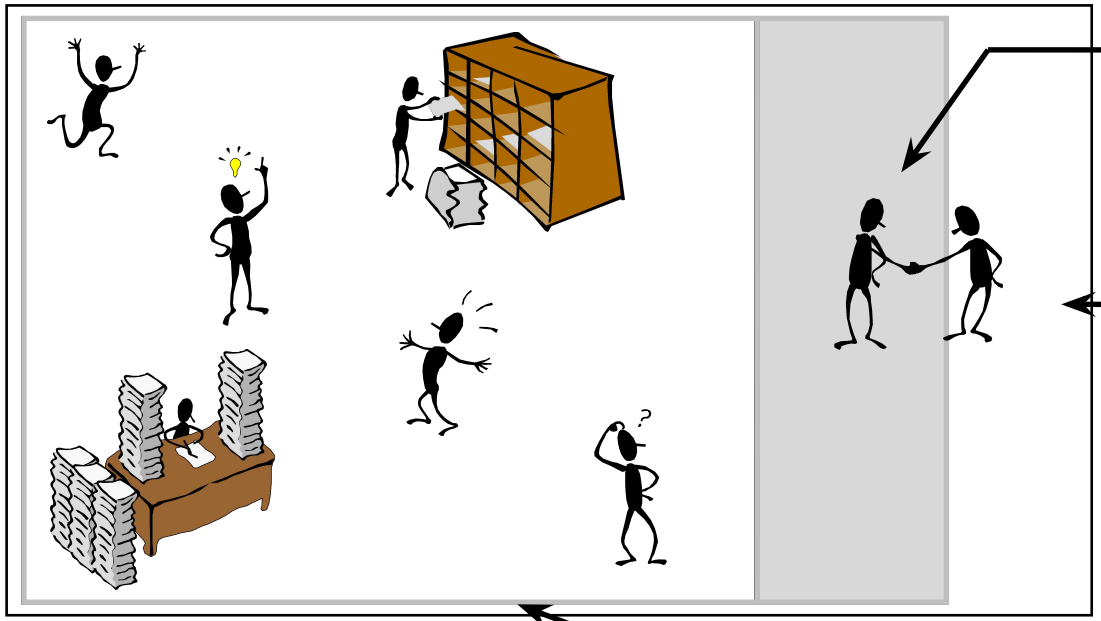
- Encapsulamento

  - ✔ Uma classe encapsula atributos e métodos, ocultando os detalhes de implementação do objetos.

  - ✔ Consiste em mostrar o que pode ser feito sem informar como é feito.

Encapsulamento

- Princípio do desenvolvimento orientado a objetos:



Encapsulamento

- Princípio do desenvolvimento orientado a objetos:

  - ✔ Um objeto deve esconder seus dados de outros objetos e permitir que esses dados sejam acessados somente por intermédio de seus próprios métodos.

  - ✔ Isso também é chamado de ocultação de informações (*information hiding*).

Encapsulamento

- Atributos e Métodos:

  - ✔ Os atributos não podem ser manipulados diretamente fora do objeto.

  - ✔ Os atributos só podem ser alterados ou consultados através dos métodos do objeto.

Atributos

- Princípio do Encapsulamento:

  - ✔ Atributos não devem ser visíveis por nenhum objeto que não seja um objeto da própria classe ou de uma classe descendente (herança).

  - ✔ Assim, devemos declarar nossos atributos sempre como **protected** ou **private**.

Atributos

- Segunda versão da classe Retangulo:

```
package principal;

public class Retangulo {
    private int x;
   private int y;
   private int largura;
   private int altura;
}
```

Atributos podem ser declarados em qualquer local dentro do corpo da classe, mas normalmente são colocados no início.

Atributos

- Inicialização de atributos:

  - ✔ Um atributo pode ser explicitamente inicializado, da mesma forma que as variáveis locais.
  - ✔ Quando não for inicializado explicitamente, a máquina virtual do Java (JVM) se encarregará de fazê-lo, da seguinte forma:
    - boolean: false
    - char: '\0'
    - byte, short, int, long, float, double: 0
    - referência para outro objeto: null

Métodos

- Agora que temos os atributos da classe Retângulo podemos implementar os métodos:
  - ✔ Desenhar
  - ✔ Mover
  - ✔ Redimensionar
  - ✔ etc.

Métodos

- Sintaxe de métodos da Classe:

visibilidade [static] [final] [abstract] *tipo método* ([*parâmetros*])
[throws *exceções*]
{
[*corpo do método*]
}

- ✔ visibilidade: public, private, package ou protected.
- ✔ static: método da classe (será visto posteriormente)
- ✔ final: método não pode ser sobrescrito (será visto posteriormente)
- ✔ abstract: método deve ser implementado nas subclasses (será visto posteriormente)

Métodos

- Método mover():

```
package principal;

public class Retangulo {
  public void mover(int novoX, int novoY) {
    x = novoX;
    y = novoY;
  }
}
```

Move o retângulo para uma nova posição (x, y)

Métodos

- Método desenhar():

```
package principal;
public class Retangulo {
  public void desenhar() {
    System.out.printf("Retangulo(%d, %d, %d, %d)\n",
                      x, y, largura, altura);
  }
}
```

Imprime os dados do retângulo

Métodos

- Método redimensionar():

```
package Principal;
public class Retangulo {
  public void redimensionar(float sx, float sy) {
    if (sx > 0 && sy > 0) {
      largura = (int) (sx / 100 * largura);
      altura  = (int) (sy / 100 * altura);
    }
  }
}
```

Calcula a nova largura/altura, que pode ser maior ou menor que largura/altura atual.

Se os novos valores de largura ou altura forem inválidos, então não redimensiona.

Classes

- Uma vez definida uma classe, podemos criar objetos ou instâncias dessa classe.
- Criação de objetos: o operador new
  - ✔ É um operador usado para criar um objeto, ou seja, uma instância de uma classe.
  - ✔ Ao declararmos uma variável do tipo de uma classe não estamos criando o objeto em si, mas somente uma referência para o objeto.
  - ✔ No momento da declaração da variável a referência está com valor null.
  - ✔ Precisamos usar o método new para que o Java aloque espaço na memória para o novo objeto.

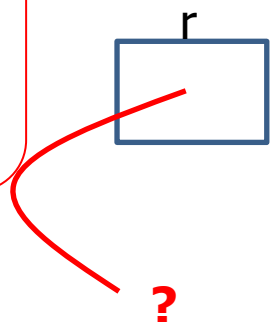
Classes

- Exemplo: criação de uma instância da classe Retangulo com o operador new.

```
Retangulo r;

r = new Retangulo();
```

Define uma variável **r** do tipo **Retangulo**. Inicialmente essa variável não referencia nenhum objeto, pois este ainda não foi criado.



Classes

- Exemplo: criação de uma instância da classe Retangulo com o operador new.

```
Retangulo r;
r = new Retangulo();
```

- Cria uma nova instância da classe **Retangulo**.
- A variável **r** passa a referenciar esse objeto.
- Os valores dos atributos são inicializados