INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação Orientada a Objeto

Aula 4 – Objetos e Classes com BlueJ

Prof. Pedro Baesse
pedro.baesse@ifrn.edu.br

Roteiro – Conceitos fundamentais

- Objeto
- Classe
- Método
- Parâmetro
- Tipo de dados

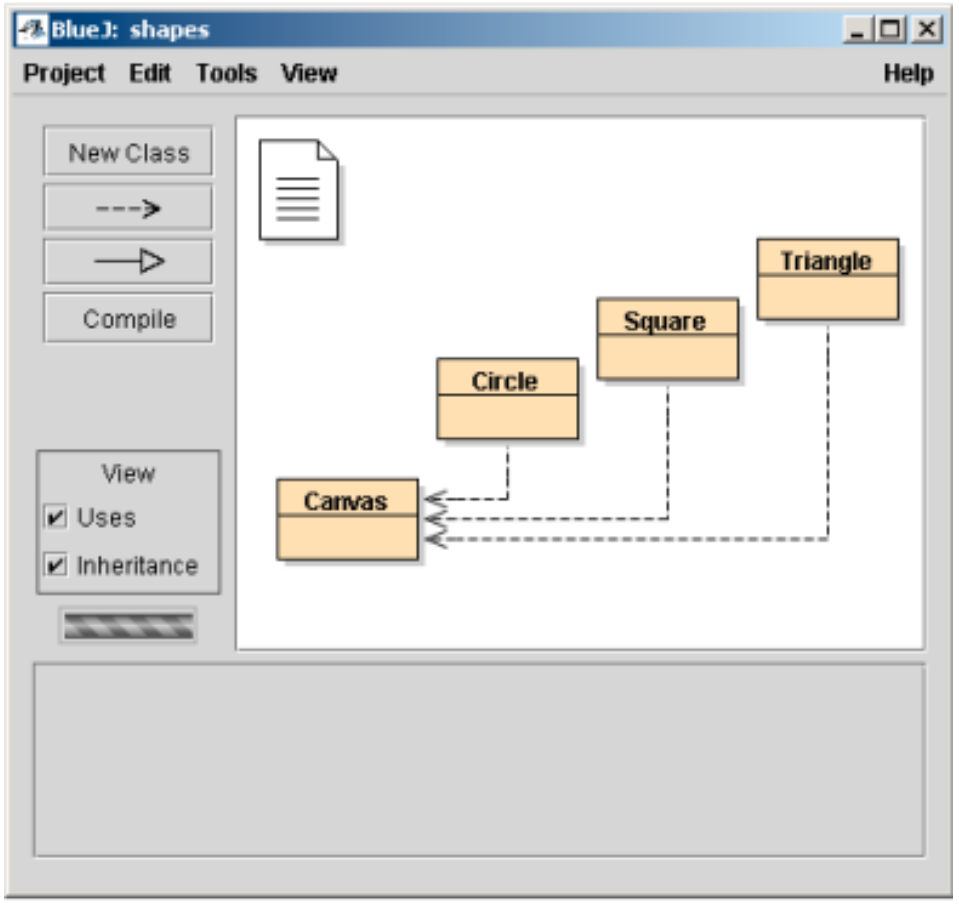
Objetos e classes

- Objetos
  - Representam ‘coisas’ do mundo real ou do domínio de algum problema (exemplo: “o carro vermelho ali no estacionamento”).

- Classes
  - Representam todos os tipos de objetos (exemplo: “carro”).

Criando objetos

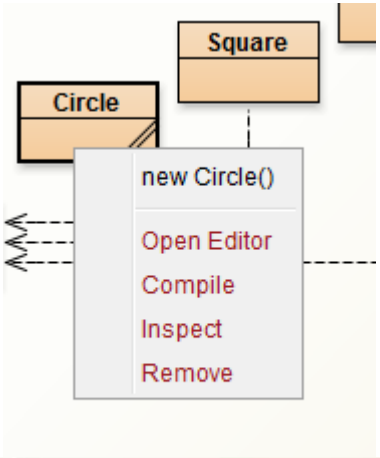
- Iniciar o BlueJ
- Abrir o exemplo chamado *shapes*
- Cada retângulo representa uma classe

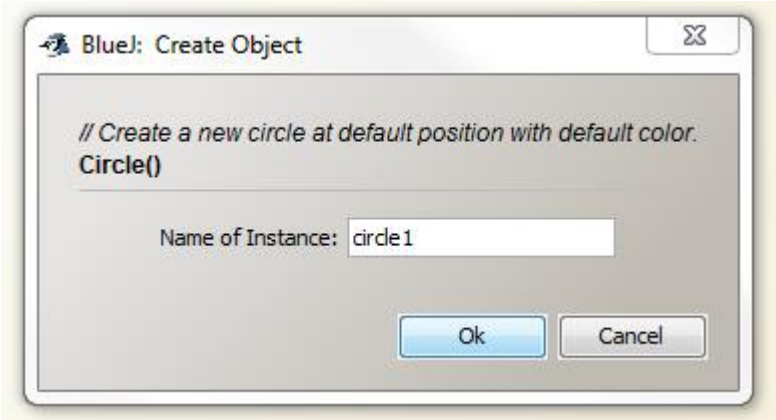


INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA RIO GRANDE DO NORTE

Criando objetos

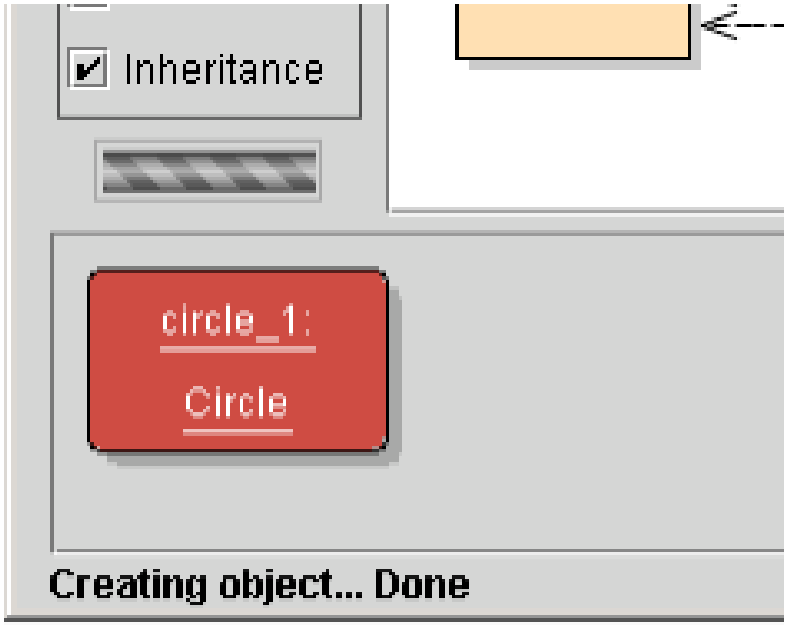
- Para criar um novo objeto
  - Clicar com botão direito do mouse e escolher
    - New Circle()
  - Preencher o nome desejado (pode usar o nome sugerido)





Criando objetos

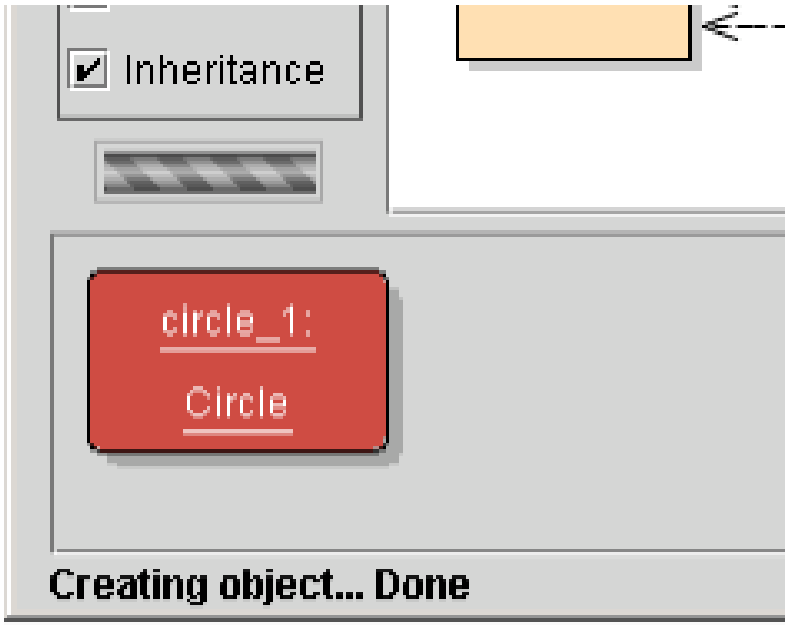
- Seu PRIMEIRO OBJETO está criado!



- É só olhar para bancada ou barra de objetos (*object bench*)

Criando objetos

- Observação
  - Classes inicia-se com letras MAÚISCULAS (como *circle*)
  - Nomes dos objetos, instâncias, com minúsculas

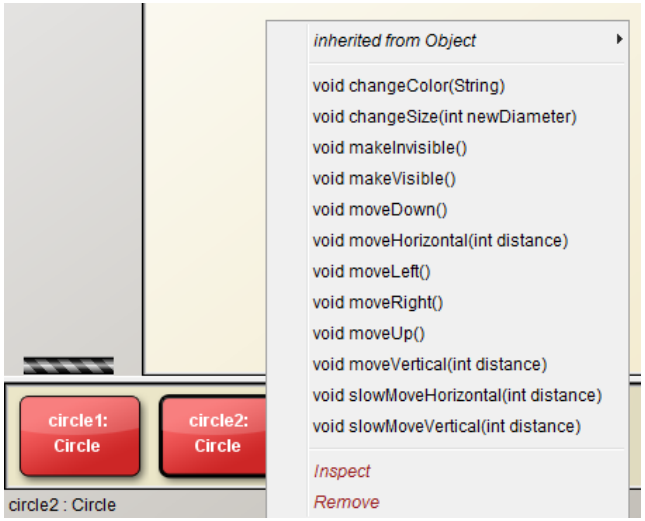


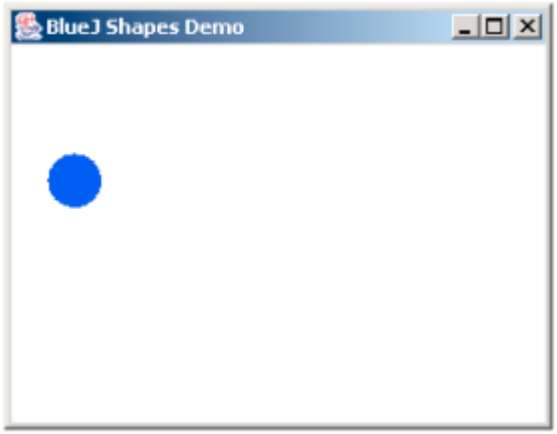
Exercício

- Crie outro círculo
- Crie outro quadrado
- Qual o resultado??

Chamando métodos

- Clicar com botão direito do mouse em um dos objetos de círculo (não na Classe!)
- Escolher *makeVisible*!
- Vamos usar as outras opções moveRight e moveDown para mover a bola para o centro
- Tente usar *makeVisible e makeInvisible* para ocultar ou mostrar o círculo





INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA RIO GRANDE DO NORTE

Exercício – Praticando

- O que acontece se você chamar o método *moveUp* duas vezes?? Ou mais??

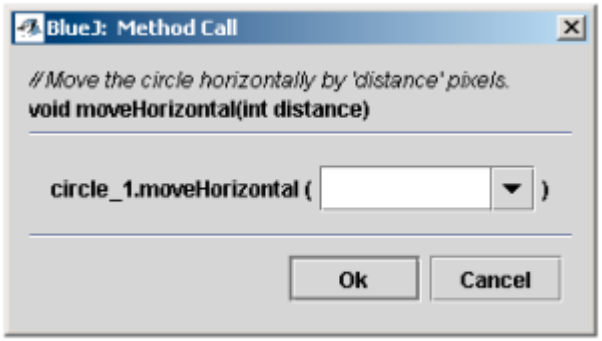
- O que acontece se você chamar o *makeInvisible* duas vezes??

Métodos

- Objetos têm operações que podem ser invocadas (o Java as chama de métodos)
- Os objetos geralmente fazem algo se invocamos um método

Usando parâmetros

- Invoque o método *moveHorizontal* e digite 54 e clique OK



- O que qual a diferença entre *moveRight* ou *moveLeft*??

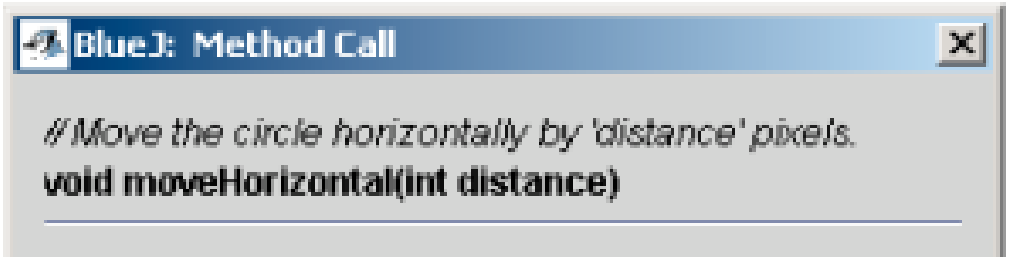
- Flexibilidade! Colocar o objeto aonde desejar!

Exercícios

- Tente invocar os métodos *moveVertical*, *slowMoveVertical* e *changeSize*
- Como você pode utilizar o moveHorizontal para mover o círculo 70 pixels para a esquerda??

Parâmetros

- Métodos podem ter **parâmetros** para passar informações adicionais necessárias para sua execução.



- O cabeçalho de um método é chamado de **assinatura.** Ela fornece informações necessárias para invocar esse métodos

- Acima da assinatura existem **comentários** que fornecem informações para o programador

Tipos de Dados

- Os parâmetros possuem **tipos**. O tipo define quais tipos de valores um parâmetro pode assumir

- O tipo *Int* indica inteiros

- O tipo *String* indica um seção de texto (palavra ou frase)
  - Sempre se usa aspas duplas (“”) para especificar o seu valor

Exercício

- Tentar usar o método *changeColor* que receber um parâmetro *String* usando “red”. Experimente outras cores.
- O que acontece quando uma cor não conhecida é especificada??
- Invoque o método changeColor e especifique a cor dos parâmetros sem as aspas

Cuidado!

- Um erro comum de iniciante é esquecer as aspas duplas ao digitar um valor do tipo *String.*

- Se você digitar *green* ao invés de "*green*", obterá uma mensagem de erro dizendo algo como 'Error: undifined variable" (ou algo parecido)

Exercício

- Crie vários círculos na bancada de objetos. Torne-os visíveis e os mova pela tela
- Faça um círculo grande amarelo
- Crie alguns triângulos e quadrados mudando suas posições, tamanhos e cores

Múltiplas instâncias

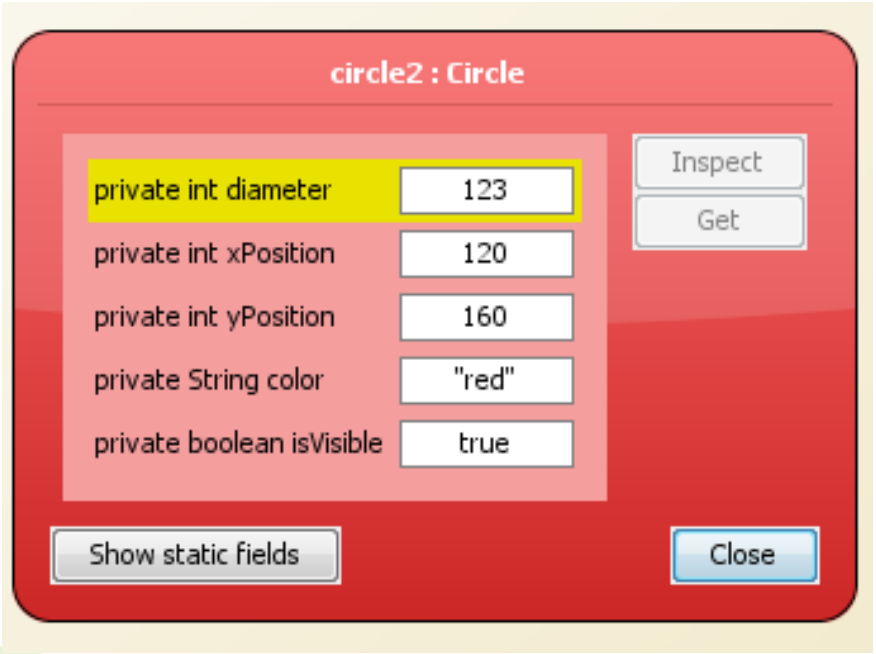
- **Várias instâncias** podem ser criadas a partir de uma única classe
- Cada instância vai ter sua própria posição, cor e tamanho
- Um chamada de método só altera os dados daquela única instância e não das outras

Estado

- Um objeto tem **atributos**: valores armazenados em campos
- A **classe** define quais **campos** um objeto tem, mas todo objeto armazena seu próprio conjunto de valores (**o estado do objeto**)

Estado

- No BlueJ, o estado de um objeto pode ser inspecionado usando a função do *Inspect* no menu de objeto. Essa janela é chamada *Object Inspector*



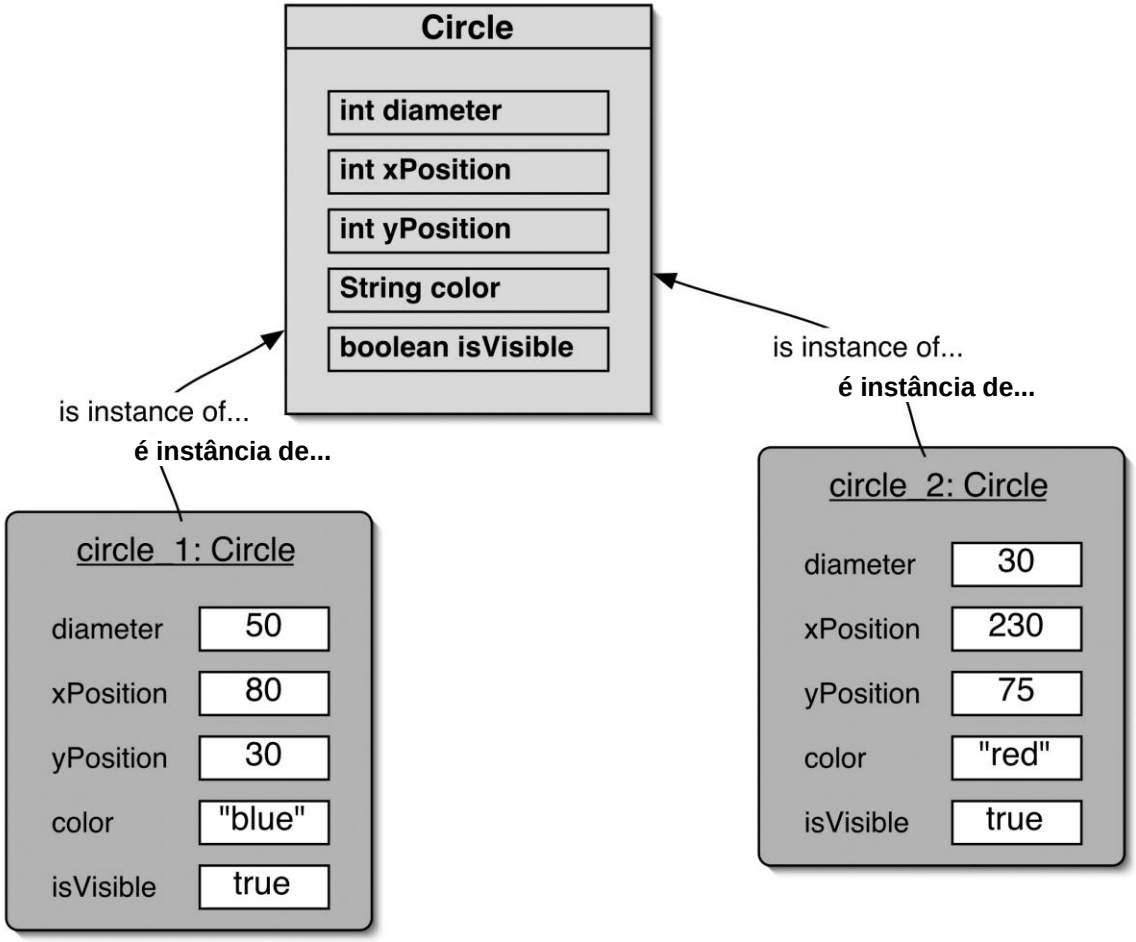
Exercício

- Com vários objetos criado, inspecione cada um deles por vez.
- Tente alterar o estado do objeto, usando qualquer um dos métodos, enquanto a janela Object Inspector está aberta
- Qual o resultado??

Conteúdo de um objeto

- Os objetos de uma **mesma** classe terão todos os mesmos campos. Porém o valor de um pode ser **diferente** em cada objeto
- Os tipo e nomes de campos são definidos em uma classe, não em um objeto
- Quando um objeto é criado, seus campos são criados automaticamente
- Os métodos funcionam da mesma maneira. Porém são invocados nos objetos informando qual objeto alterar

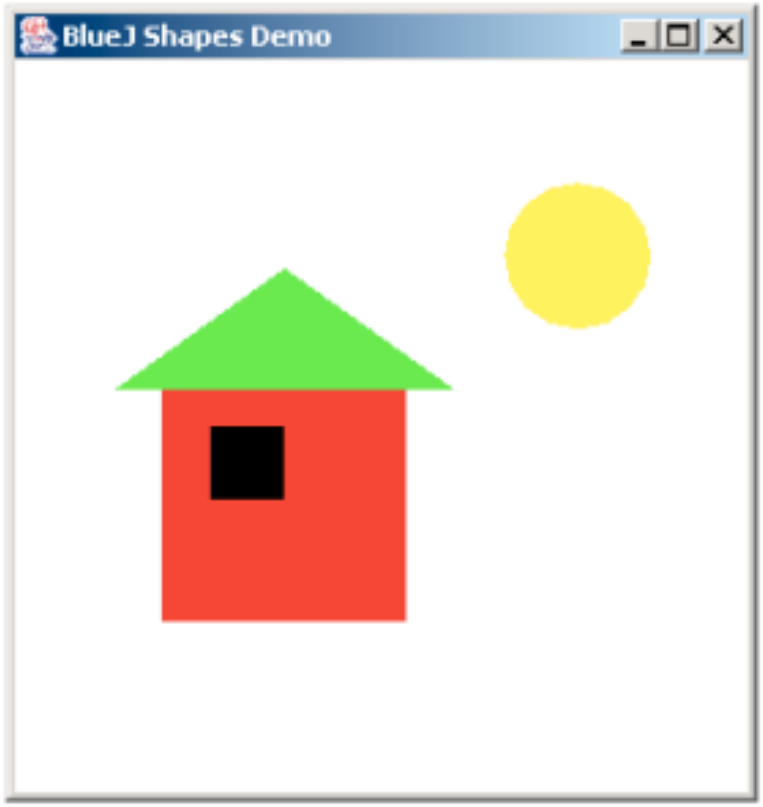
Dois objetos circle



Exercício

- Usando as formas do projeto *shape* crie uma imagem de uma casa e de um sol.
  - Anote os passos para chegar para ao resultado final
  - Isso poderia ser feito de maneira diferente??

Exercício



Interação entre objetos

- Feche o projeto *shapes* e abra o projeto *picture*
- Crie uma instância da classe *Picture* e invoque o método *draw*. Use também os métodos *setBlackAndWhite* e *setColor*
- Como a classe *Picture* desenha a figura??

Interação entre objetos

- A novidade é classe *Picture*. Uma classe que faz o mesmo do último exercício chamando apenas um método.

- Ela foi criada de uma maneira que quando instânciada, cria também outros formas e as move para formar a figura

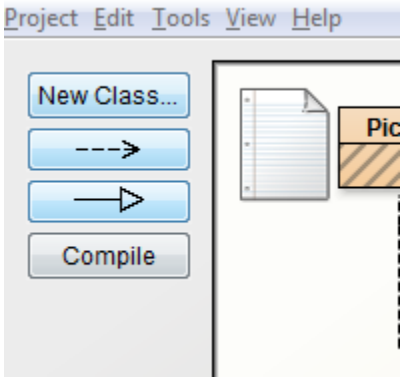
- **Objetos podem criar outros objetos!** E também chamar métodos uns dos outros. Em um programa Java isso pode acontecer centenas, até milhares de vezes

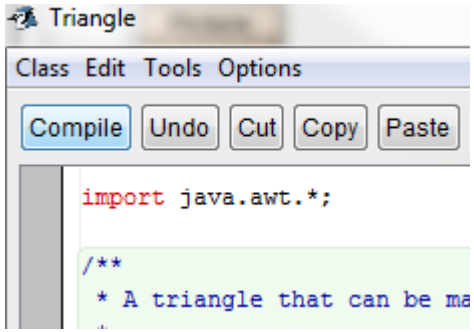
Código-fonte

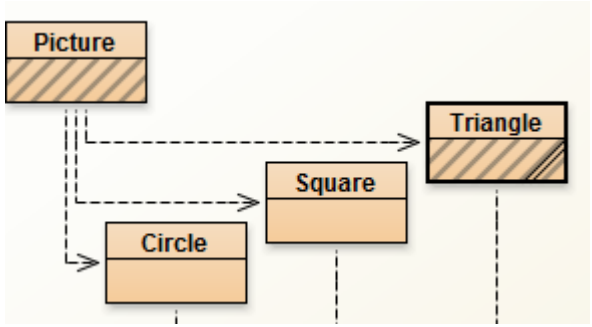
- Toda classe tem um **código-fonte** (código Java) associado a ela que define seus detalhes (campos e métodos)
- Acesse o menu da classe *Picture* com o botão direito do mouse e escolha a opção *Open Editor*
- Uma grande parte do aprendizado da arte da programação é aprender a escrever essas definições

Código-fonte

- Quando é feita alguma alteração no código, as classes se tornam listradas indicando que precisam ser compiladas novamente
- Isso pode ser feito no botão *compile*, no *Open Editor* e barra lateral







INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA RIO GRANDE DO NORTE

Exercício

- Localizar a parte que realmente desenha a figura e alterar o cor do sol de azul ao invés de amarelo
- Adicione um segundo sol a figura. Preste atenção às definições
- Será necessário acrescentar uma linha para o segundo sol
  - *Private Circule sun2*

```
private Square wall;
private Square window;
private Triangle roof;
private Circle sun;
```

Desafio

- Adicione um pôr-do-sol à ultima versão de *Picture*. Ou seja, faça o sol se por lentamente. Lembre-se do *slowMoveVertical* que pode ser usado para isso

Desafio

- Se adicionou o pôr-do-sol no fim do método *draw,* vamos mudar isso. Crie um método separado para possamos chamar o método *sunset,* separadamente de *draw*, e não ao mesmo tempo.

Outro exemplo

- Para ajudar na fixação dos conceitos vistos, vamos utilizar outro projeto, *lab classes.*
- Uma parte simplificada de um banco de dados de aluno

Exercício

- Crie um objeto da classe *Student.* Dessa vez vários parâmetros serão pedidos inicialização e não somente o nome da instância. Preencha todos antes de apertar em Ok

- Crie alguns objetos de alunos. Chame o método *getName* em cada objeto. O que está acontecendo??

Valores de retorno

- A assinatura do método *getName*
  - *String getName()*
- A palavra *String* antes do nome do método indica que retornará um resultado do tipo *String*

- E a assinatura do método *changeName*
  - *void changeName(String)*
- A palavra *void* indica que esse método não retorna nenhum valor

Métodos

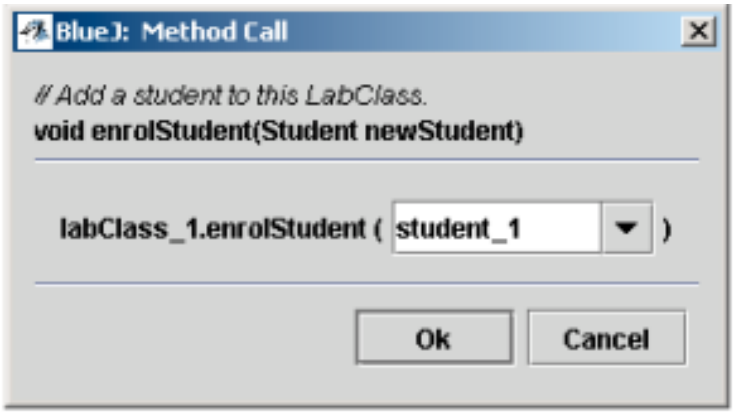
- Os métodos com retornos de valores nos permite obter informações via invocação de métodos
- Assim podemos usar métodos para alterar o estado do um objeto ou descobrir informações sobre o seu estado

Exercício

- Crie um objeto da classe *LabClass.* É preciso definir o número máximo de alunos
- Chame o método *numberOfStudents* dessa classe. O que ele faz??

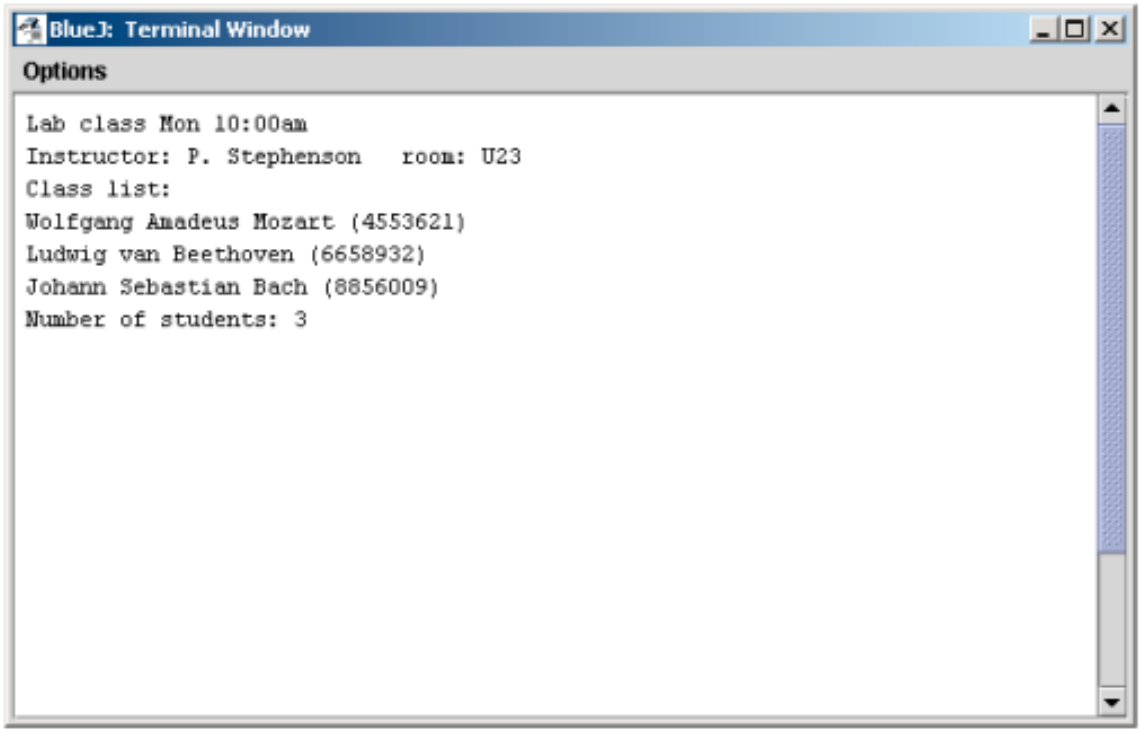
Exercício

- Veja a assinatura do método *enrollStudent*. O tipo é *Student*. Crie alguns alunos e invoque o método *enrollStudent*. Clique em alguns dos objetos alunos para inseri-lo no campo de entrada do método *enrollStudent*. Adicione vários alunos



Exercício

- Chame o método *printList* do objeto *LabClass*



Objetos como parâmetros

- Observa-se pelo exercícios que objetos podem ser parâmetros de métodos de outros objetos

- Quando um objeto é esperado como parâmetro a classe define o tipo e o nome do objeto desejado deve ser passado como parâmetro

Exercícios

- Crie três alunos de acordo com os detalhes e os insira em um laboratório

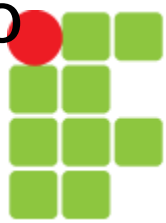
Snow White, student ID: *100234*, credits: *24*
Lisa Simpson, student ID: *122044*, credits: *56*
Charlie Brown, student ID: *12003P*, credits: *6*

- Use o inspetor em um objeto para descobrir os campos dele

- Configure o instrutor, a sala e hora de um laboratório e imprima a lista para verificar os novos detalhes

Resumo

- Objeto
- Classe
- Método
- Parâmetro
- Assinatura
- Tipo
- Múltiplas instâncias
- Estado
- Chamada de método
- Código-fonte
- Resultado



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Dúvidas

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Referências

David J. Barnes & Michael Kölling

Programação orientada a objetos com Java

Pearson Education do Brasil, 2004
ISBN 85-7605-012-9.