Objetos e classes

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

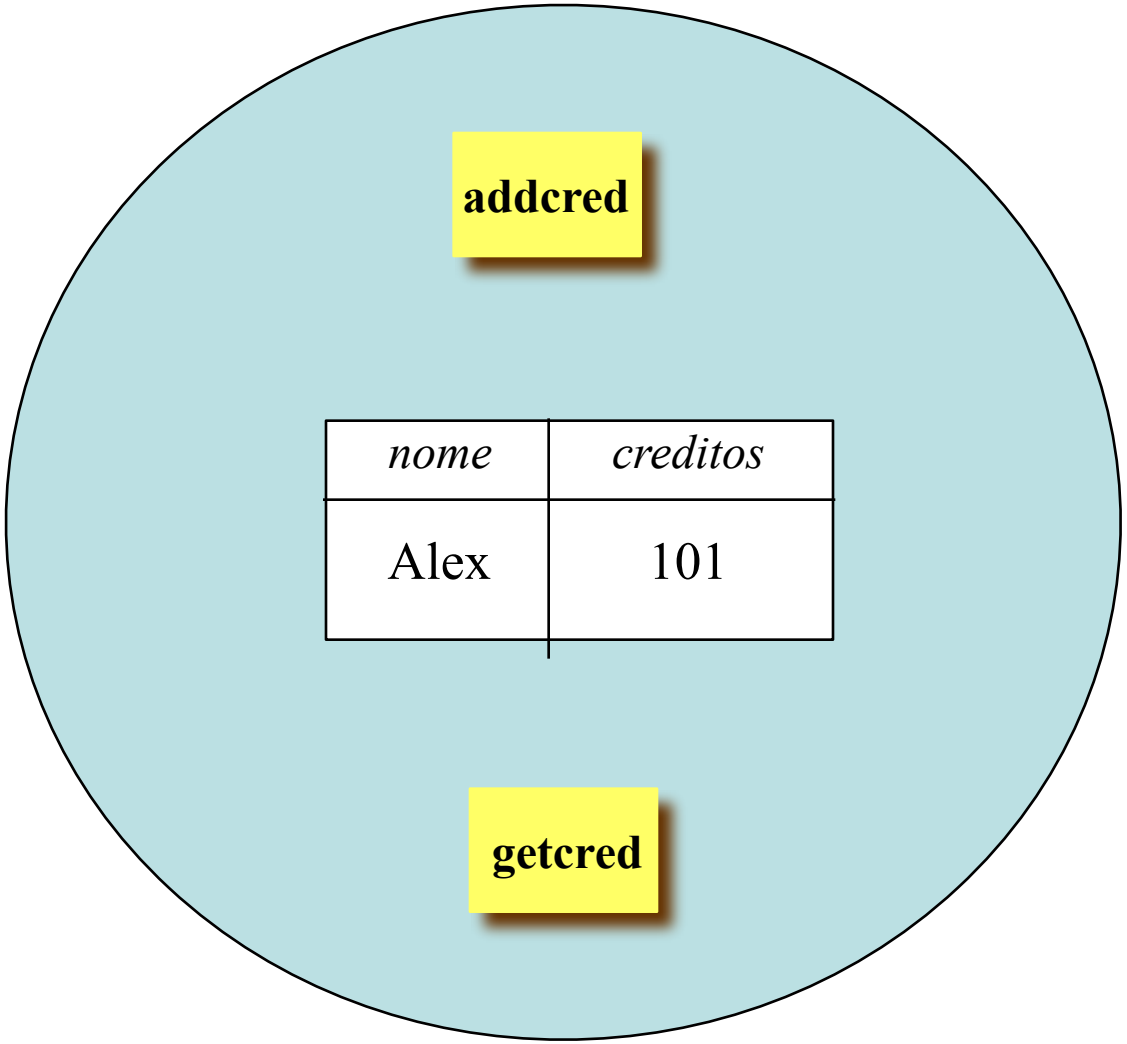
Universidade: um aplicativo

- Em uma universidade, os professores ensinam estudantes que são matriculados em cursos. Um curso é uma coleção de tópicos relacionados. Estudantes se matriculam para cursar diferentes cursos. Desde que existem limites da quantidade de estudantes que um professor possa ensinar de uma só vez (a universidade tem como norma não possuir grandes turmas), quando muitos estudantes precisando tomar um curso poderá haver várias turmas do mesmo curso ministradas pelo mesmo professor ou por mais de um professor.

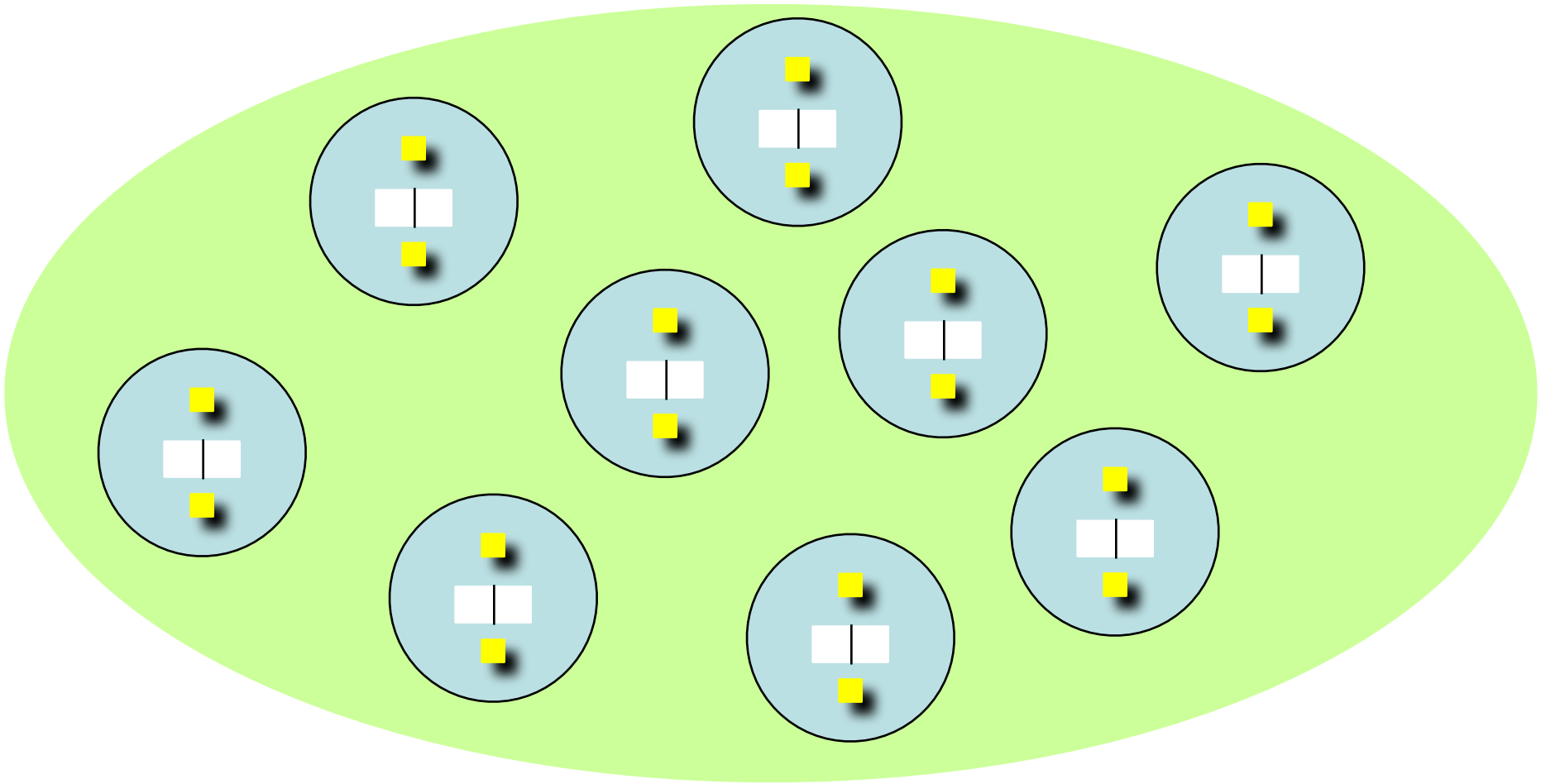
Objetos e classes

- Objetos
  - Representam ‘coisas’ do mundo real ou do domínio de algum problema (exemplo: “um estudante de Sistemas de Informação de nome Alex”).
- Classes
  - Representam todos os tipos de objetos (exemplo: “Estudante”).

Objeto Estudante



Classe de Estudantes



Classe de Estudantes em Python

```
class Estudante:

    def  addCred(self, pontosAdicionais):
        self.creditos +=  pontosAdicionais
```

Classe de Estudantes em Python

```
class Estudante:
    """attributes: nome, creditos"""
    def addCred(self, pontosAdicionais):
        self.creditos +=  pontosAdicionais
```

salve em um arquivo estudante.py

Operadores em Python

Operadores + - * / % // **

Operadores -= += *= /= %=

T -=5 é equivalente a T = T - 5

T+=5 é equivalente a T = T + 5

T*=5 **T=T*5**

T/=5 **T=T/5**

T%=5 **T=T%5**

igualdade ==

desigualdade !=

relacionais < <= >= >

lógicos and or not

Criando Instâncias (Objetos)

Forma geral da criação de objetos

- Criação

  <id> = <id-classe> (<args>)

- exemplo:

  aluno1 = Estudante()

Ex.: programa que cria e manipula um estudante (prog1.py)

```
from estudante import Estudante    # estudante.py

# Criando um objeto do tipo Estudante

 aluno1 = Estudante ()

 aluno1.nome = "Alexandre"  # referencia a atributos
 aluno1.creditos = 0

 aluno1.addCred (48)   # referencia a metodos

 print(aluno1.creditos)
```

Ex.: programa que cria e manipula um estudante (prog2.py)

```
from estudante import Estudante    # estudante.py
def  fprog():
     # Criando um objeto do tipo Estudante

     aluno1 = Estudante ()

     aluno1.nome = "Alexandre"  # referencia a atributos
     aluno1.creditos = 0

     aluno1.addCred (48)   # referencia a metodos

     print(aluno1.creditos)
```

Compilação e execução do programa

Usando o interpretador na linha de comando

No final do arquivo prog2.py insira:

```
if __name__=="__main__":
    fprog()

python prog2.py        // para interpretar o programa
```

Ex.: programa que cria e manipula um estudante (main.py)

```
from estudante import Estudante   # estudante.py
def  main():
    # Criando um objeto do tipo Estudante

    aluno1 = Estudante ()

    aluno1.nome = “Alexandre”  # referencia a atributos
    aluno1.creditos = 0

    aluno1.addCred (48)   # referencia a metodos

    print(aluno1.creditos)
```

Compilação e execução do programa

Usando o interpretador na linha de comando

No final do arquivo main.py insira:

```
if __name__=="__main__":
    main()

python main.py          // para interpretar o programa
```

Altere o programa principal

Altere o programa principal para criar um outro estudante, estabelecendo os créditos iniciais para 244. Acrescente mais 48. Imprima o novo valor dos créditos.

Criando outros objetos

- circle_1 = Circle()
- Crie outro círculo.
- Em seguida, crie um quadrado.

Métodos

- Objetos têm operações que podem ser invocadas (o Python as chama de *métodos*).
- Nos comunicamos com objetos invocando seus métodos.
- Os objetos fazem algo se invocarmos um método.

Chamando métodos

- circle_1.makeVisible()
- circle_1.moveRight()
- circle_1.moveDown()

- O cabeçalho de um método é chamado de assinatura – informações necessárias para invocar o método

- Escreva a assinatura dos métodos acima.

Parâmetros

- Métodos podem ter parâmetros para passar informações adicionais necessárias para sua execução.
- Assinatura do método com parâmetros:

  moveHorinzontal (self, distance)
- Invocação:

  circle_1.moveHorinzontal(50)

Tipos de dados

- Os parâmetros possuem tipos.
- O tipo define quais tipos de valores um parâmetro pode assumir
- O tipo inteiro significa números inteiros

```
i = 10
```

- O tipo string indica que uma seção de texto é esperada

```
cor = 'red'
```

Tipos de dados

Primitivos:

- **boolean**
- **int, long,**
- **float**
- **complex**

de referência:

- **objetos, listas, tuplas**

Múltiplas instâncias

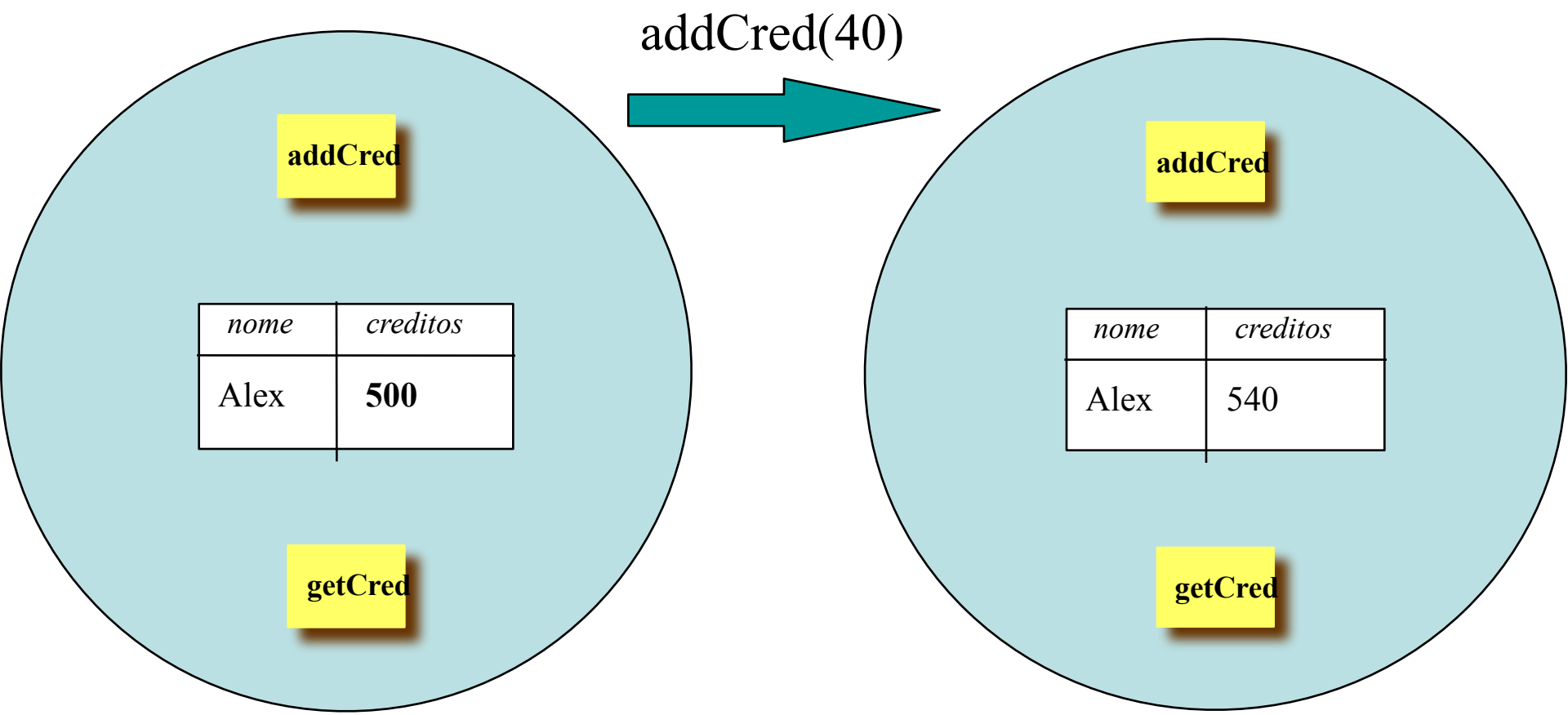
- Várias *instâncias* podem ser criadas a partir de uma única classe.

- Um objeto tem *atributos*:
  - valores armazenados em variáveis.

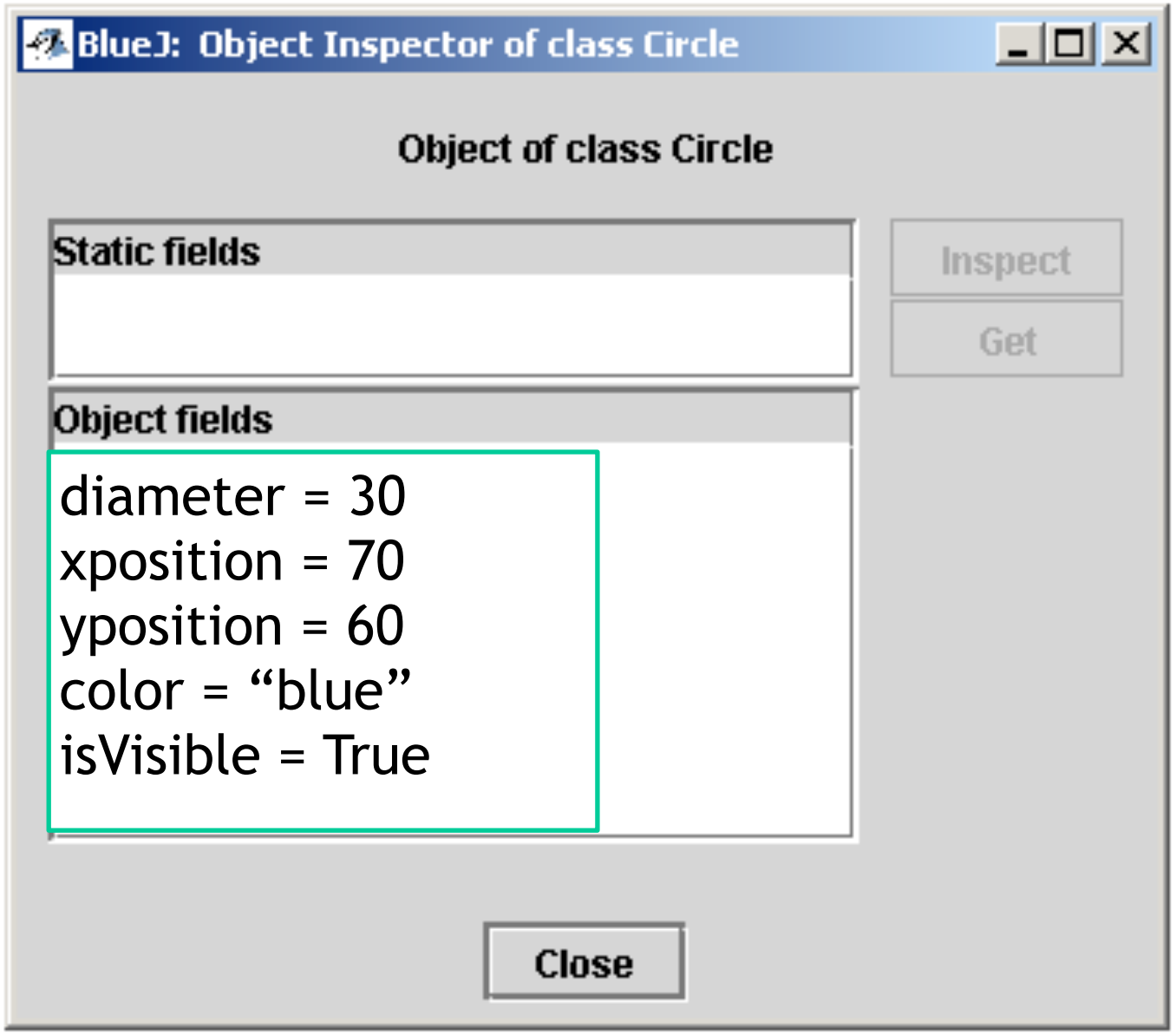
Múltiplas instâncias

- A classe define quais atributos um objeto tem
- mas todo objeto armazena seu próprio conjunto de valores
  - o e*stado* do objeto.

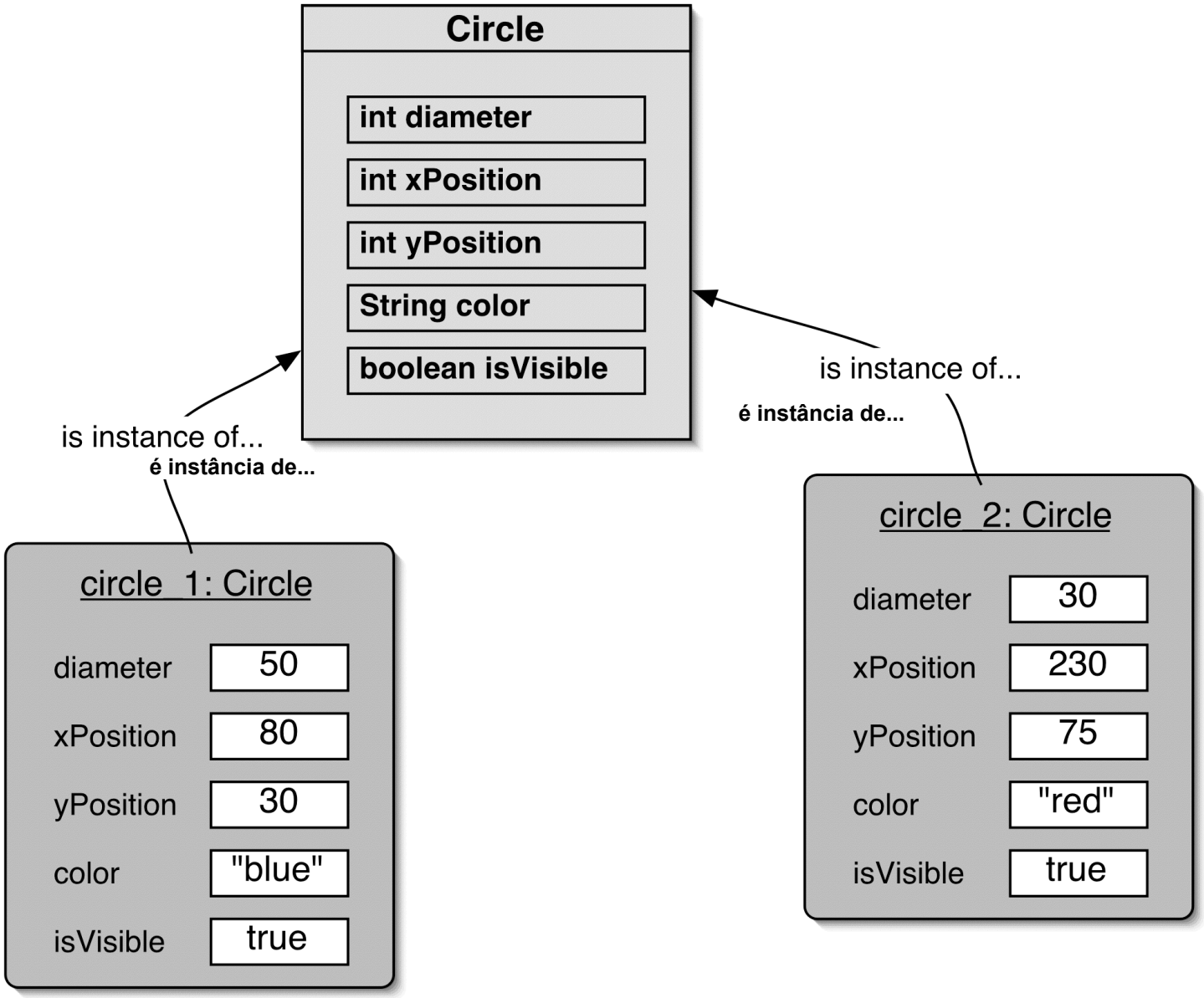
Estados do Objeto Estudante



Estado


BlueJ: Object Inspector of class Circle
Object of class Circle
Static fields
Inspect
Get
Object fields
diameter = 30
xposition = 70
yposition = 60
color = “blue”
isVisible = True
Close

Dois objetos circle



Interação entre objetos

- Os objetos podem criar outros objetos, e podem chamar métodos uns dos outros.
- Usuário apenas inicia o programa e todos os outros objetos são criados – direta ou indiretamente - por esse objeto.

Código-fonte

- Toda classe tem um código-fonte (código Python) associado a ela que define seus detalhes (atributos e métodos).
- A arte de programação OO consiste em aprender a escrever as definições de classe.

Valores de retorno

- Métodos podem retornar um resultado via um valor de retorno.
- getNome(self)
- changeNome(self, newnome)

Resumo dos conceitos

- **objeto** Objetos Python modelam objetos reais a partir do domínio de um problema.
- **classe** Os objetos são criados a partir de classes. A classe descreve o tipo de objeto; os objetos representam instanciações individuais de classe.

Resumo dos conceitos (2)

- **método** A comunicação com os objetos é através de seus métodos. Os objetos fazem algo se invocarmos um método.
- **parâmetro** Os métodos podem ter parâmetros para fornecer informações adicionais para uma tarefa.

Resumo dos conceitos (3)

- **assinatura** O cabeçalho de um método é chamado de assinatura e fornece as informações necessárias para invocar esse método.
- **tipo** Os parâmetros possuem tipos. O tipo define quais tipos de valores um parâmetro pode assumir.

Resumo dos conceitos (4)

- **múltiplas instâncias** Muitos objetos semelhantes podem ser criados a partir de uma única classe.
- **estado** O estado de um objeto é representado pelo valores armazenados em seus atributos.

Resumo dos conceitos (5)

- **chamada de método** Os objetos se comunicam chamando os métodos uns dos outros.
- **código-fonte** Determina a estrutura e o comportamento de cada um dos objetos nessa classe.
- **resultado** Os métodos podem retornar informações sobre um objeto por meio de um **valor de retorno**.

Hora-Trabalho_Aula03

- Defina uma classe em Python para representar um docente de uma universidade em um sistema acadêmico.
- Quais os atributos relevantes?
- Quais os métodos relevantes?