# IFRN

## Tecnologia em Análise e Desenvolvimento de Sistemas

# Valores e Referências

Prof. Gilbert Azevedo

# Conteúdo

- Organização da Memória
- Tipos por Valor
- Tipos por Referência
- Boxing e Unboxing

# Organização da Memória

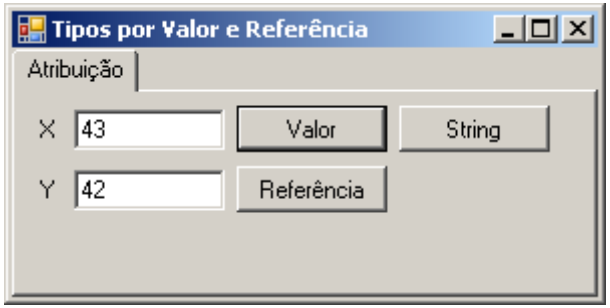
- Sistemas operacionais e *runtimes* dividem a memória para armazenar dados em dois blocos: *stack* e *heap*
  - Parâmetros e variáveis locais são adquiridas da *stack* na chamada de um método
  - A *stack* é liberada quando o método termina
  - Objetos alocados com *new* adquirem memória da *heap*
  - A memória é liberada pelo *Garbage Collection* quando o objeto não possui mais nenhuma referência
- A *stack* é organizada como uma pilha de caixas uma sobre as outras
- A *heap* é como um grande monte de caixas espalhadas com um rótulo indicando o seu uso

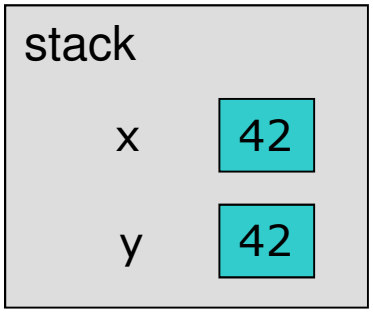
# Tipos por Valor e por Referência

- O C# apresenta dois tipos de dados: tipos por valor (*Value Types*) e tipos por referência (*Reference Types*)
- Tipos por valor
  - Armazenam diretamente os dados
  - São alocados na região de memória *stack* (pilha)
  - Devem ser iniciados antes de utilizados
  - Não podem receber o valor *null*
- Tipos por referência
  - Contêm uma referência para os dados
  - São alocados na região de memória *heap* (monte)
  - São iniciados através do operador *new*
  - São destruídos pelo coletor de lixo

# Exemplo de Tipos por Valor

- As variáveis *x* e *y* armazenam os dados em locais distintos da memória *stack*
  - private void button1_Click(object sender, EventArgs e) {
  - int x, y;
  - x = 42;
  - y = x;
  - x = 43;
  - textBox1.Text = x.ToString();
  - textBox2.Text = y.ToString();
  - }



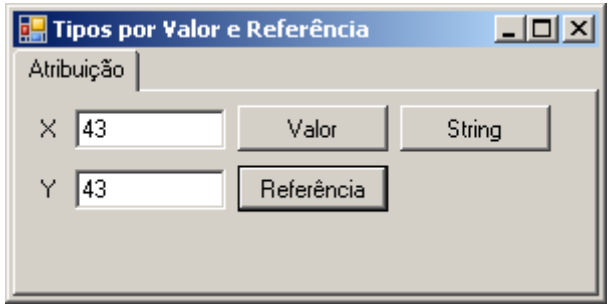



# Tipos por Valor

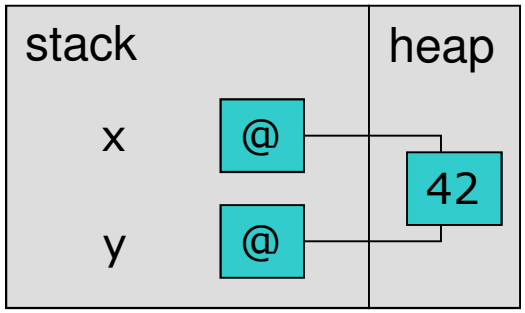
- Booleanos, inteiros, reais e caracteres são tipos por valor
- São descendentes da classe *System.ValueType*
- Quando uma variável é declarada, o C# aloca um bloco de memória grande o suficiente para conter seu valor
- Quando uma variável é atribuída a outra, os valores ficam duplicados na memória

# Exemplo de Tipos por Referência

- Dada a classe WrappedInt
  - class WrappedInt {
  - public int Number;
  - }




- As variáveis *x* e *y* compartilham os dados na *heap*
  - WrappedInt x = new WrappedInt();
  - WrappedInt y;
  - x.Number = 42;
  - y = x;
  - x.Number = 43;
  - textBox1.Text = x.Number.ToString();
  - textBox2.Text = y.Number.ToString();

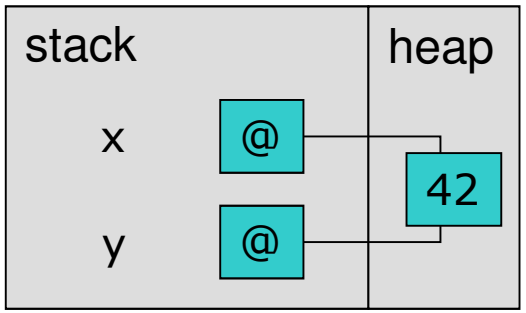


# Tipos por Referência

- Todos os tipos de classe (definidos com *class*) são tipos por referência
- O tipo primitivo *string* é um tipo por referência especial
- Quando uma variável é declarada, o C# aloca memória apenas para armazenar o endereço do objeto
- A memória que contém os dados do objeto somente é alocada através do operador *new* (exceto para *strings*)
- Atribuições entre tipos por referência não implicam na duplicação de valores na memória

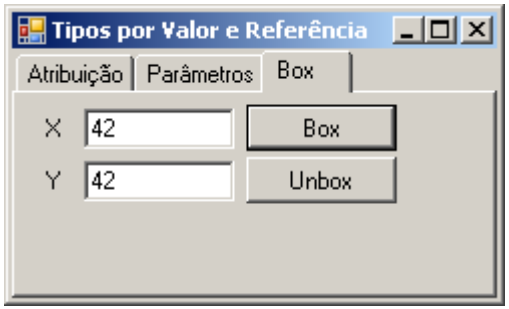
# Boxing e Unboxing

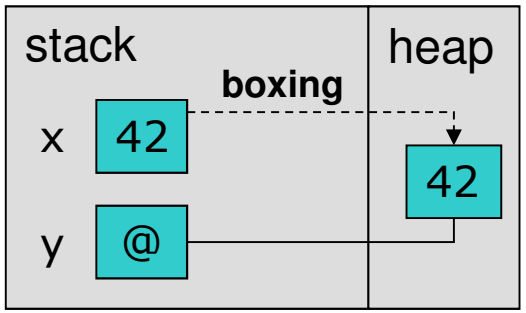
- Variáveis do tipo *object* podem referenciar tipos por referência e tipos por valor
- Quando um *object* referencia um tipo por referência ele armazena o endereço do objeto que é alocado na memória *heap*
  - WrappedInt x = new WrappedInt();
  - x.Number = 42;
  - object y;
  - y = x;



# Boxing

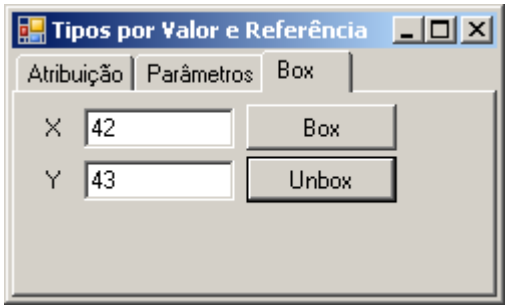
- Quando um *object* referencia um tipo por valor uma operação *Boxing* é realizada
- A operação realiza uma cópia do tipo por valor para a memória *heap* e a variável *object* armazena o endereço do valor copiado
  - int x = 42;
  - object y = x;
  - textBox6.Text = y.ToString();

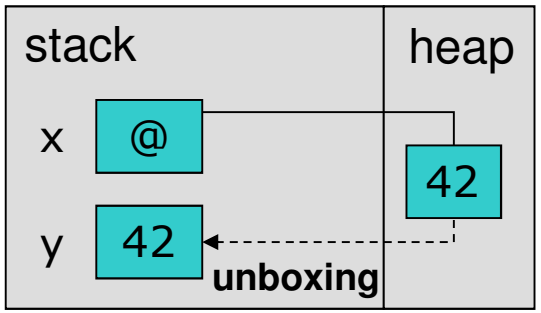


# Unboxing

- Para acessar novamente o valor *boxed* na memória *heap* como um tipo por valor é necessário fazer uma operação de *unboxing*
- A operação *unboxing* deve ser realizada com um *cast*
  - object x = 42;
  - int y = (int)x;
  - y++;
  - textBox6.Text = y.ToString();



# Referencias Bibliográficas


- Introduction to C# Programming with Microsoft .Net
  - Microsoft Official Course 2609A
- Microsoft Visual C# 2005 - Passo a passo
  - John Sharp, Bookman, 2007
- Microsoft Asp.Net – Passo a passo
  - George Sheperd, Bookman, 2007
- Microsoft VS 2005 Express Edition Documentation