DEVSUPERIOR

Java e Orientação a Objetos
Capítulo: Memória, vetores, listas

https://devsuperior.com.br

Prof. Dr. Nelio Alves

1

Tipos referência vs. tipos valor

http://educandoweb.com.br

Prof. Dr. Nelio Alves

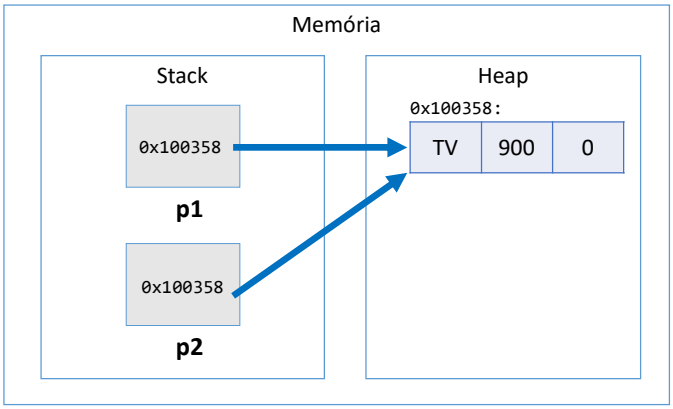
2

Classes são tipos referência

Variáveis cujo tipo são classes não devem ser entendidas como caixas, mas sim "tentáculos" (ponteiros) para caixas

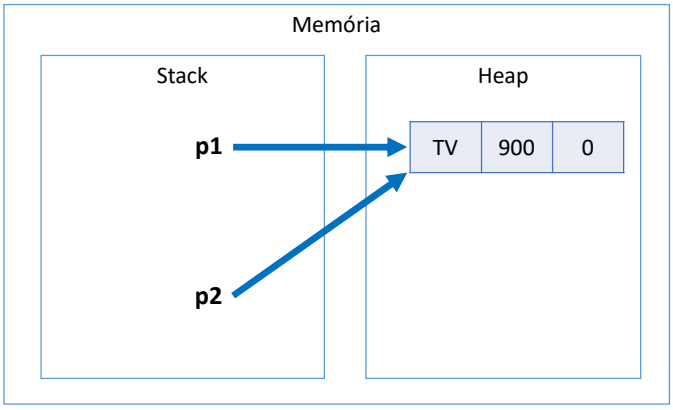
```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = p1;
```

p2 = p1;
"p2 passa a apontar para onde p1 aponta"



Desenho simplificado

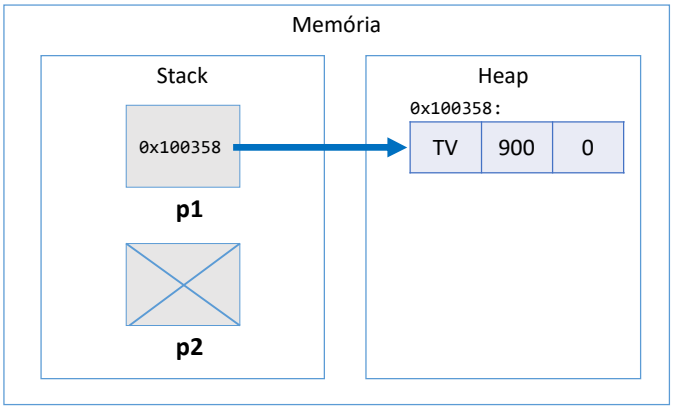
```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = p1;
```



Valor "null"

Tipos referência aceitam o valor "null", que indica que a variável aponta pra ninguém.

```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = null;
```

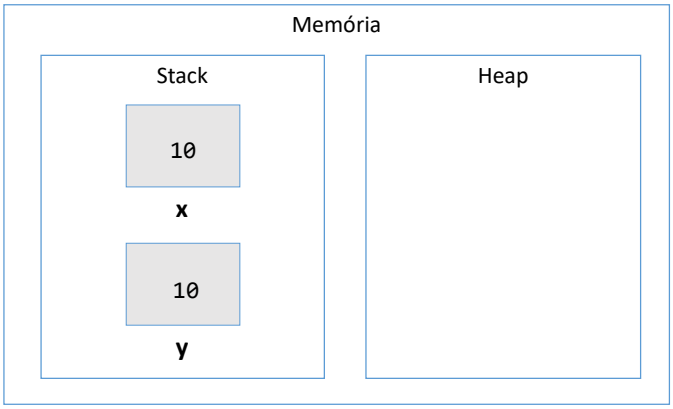


Tipos primitivos são tipos valor

Em Java, tipos primitivos são tipos valor. Tipos valor são CAIXAS e não ponteiros.

```
double x, y;

x = 10;

y = x;
```

y = x;
"y recebe uma CÓPIA de x"



Type	Contains	Default	Size	Range
`boolean`	`true` or `false`	`false`	1 bit	NA
`char`	Unicode character	`\u0000`	16 bits	`\u0000` to `\uFFFF`
`byte`	Signed integer	0	8 bits	-128 to 127
`short`	Signed integer	0	16 bits	-32768 to 32767
`int`	Signed integer	0	32 bits	-2147483648 to 2147483647
`long`	Signed integer	0	64 bits	-9223372036854775808 to 9223372036854775807
`float`	IEEE 754 floating point	0.0	32 bits	±1.4E-45 to ±3.4028235E+38
`double`	IEEE 754 floating point	0.0	64 bits	±4.9E-324 to ±1.7976931348623157E+308

Tipos primitivos e inicialização

- Demo:

```
int p;
System.out.println(p); // erro: variável não iniciada

p = 10;
System.out.println(p);
```

Valores padrão

- Quando alocamos (new) qualquer tipo estruturado (classe ou array), são atribuídos valores padrão aos seus elementos
  - números: 0
  - boolean: false
  - char: caractere código 0
  - objeto: null

```
Product p = new Product();
```

Tipos referência vs. tipos valor

CLASSE	**TIPO PRIMITIVO**
Vantagem: usufrui de todos recursos OO	Vantagem: é mais simples e mais performático
Variáveis são ponteiros	Variáveis são caixas
Objetos precisam ser instanciados usando new, ou apontar para um objeto já existente.	Não instancia. Uma vez declarados, estão prontos para uso.
Aceita valor null	Não aceita valor null
Y = X; "Y passa a apontar para onde X aponta"	Y = X; "Y recebe uma cópia de X"
Objetos instanciados no heap	"Objetos" instanciados no stack
Objetos não utilizados são desalocados em um momento próximo pelo garbage collector	"Objetos" são desalocados imediatamente quando seu escopo de execução é finalizado

Desalocação de memória - garbage collector e escopo local

http://educandoweb.com.br

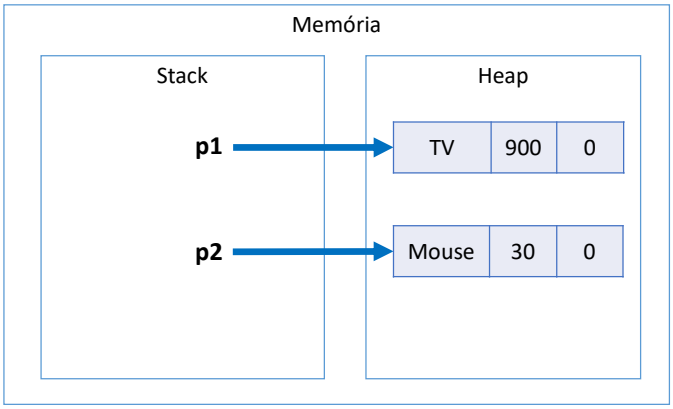
Prof. Dr. Nelio Alves

Garbage collector

- É um processo que automatiza o gerenciamento de memória de um programa em execução
- O garbage collector monitora os objetos alocados dinamicamente pelo programa (no heap), desalocando aqueles que não estão mais sendo utilizados.

Desalocação por garbage collector

```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = new Product("Mouse", 30.00, 0);
```

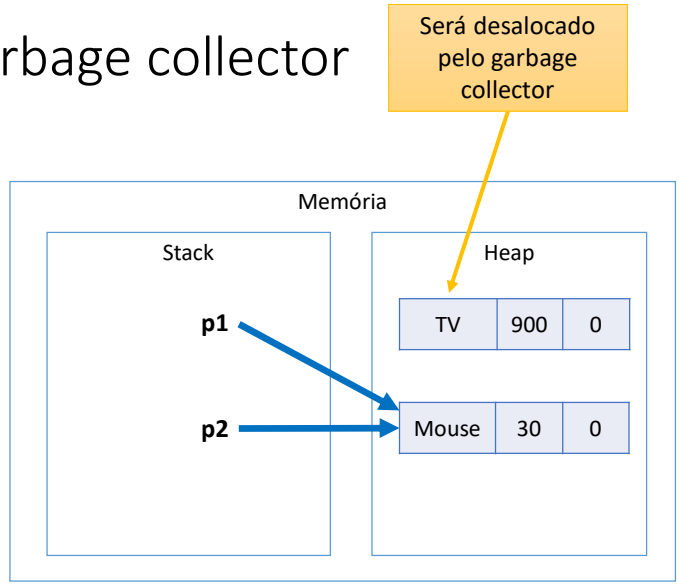


13

Desalocação por garbage collector

```
Product p1, p2;

p1 = new Product("TV", 900.00, 0);

p2 = new Product("Mouse", 30.00, 0);
```

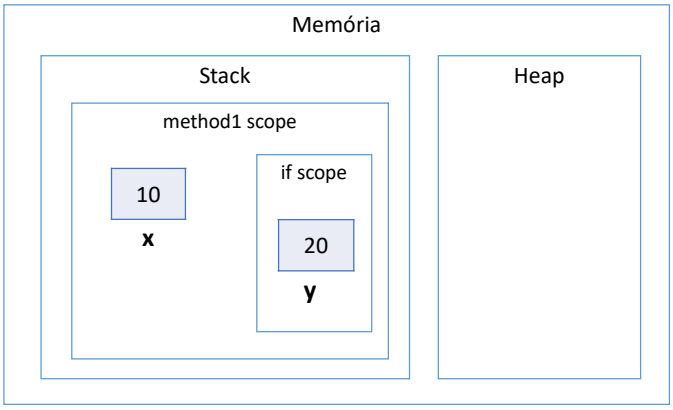
```
p1 = p2;
```



14

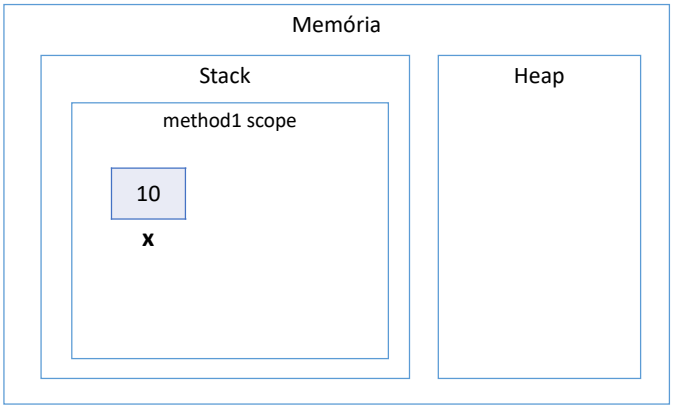
Desalocação por escopo

```
void method1() {
    int x = 10;
    if (x > 0) {
        int y = 20;
    }
    System.out.println(x);
}
```



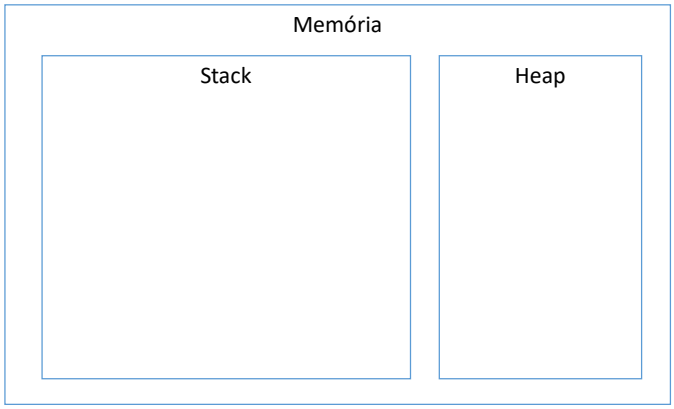
Desalocação por escopo

```
void method1() {
    int x = 10;
    if (x > 0) {
        int y = 20;
    }
    System.out.println(x);
}
```



Desalocação por escopo

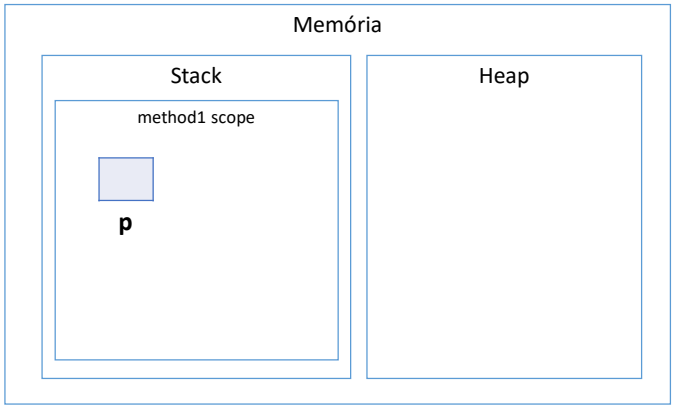
```
void method1() {
    int x = 10;
    if (x > 0) {
        int y = 20;
    }
    System.out.println(x);
}
```



17

Outro exemplo

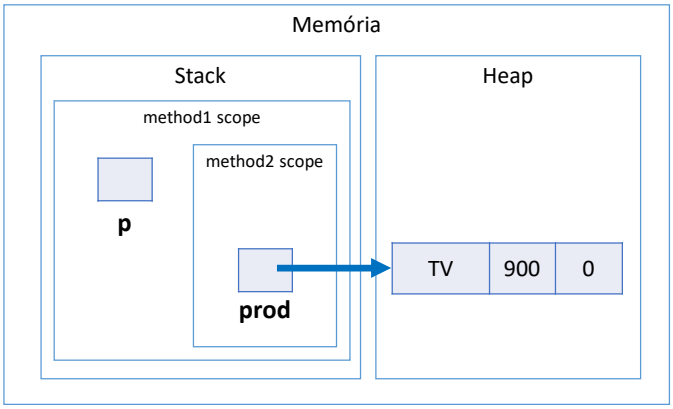
```
void method1() {
  Product p = method2();
  System.out.println(p.Name);
}

Product method2() {
  Product prod = new Product("TV", 900.0, 0);
  return prod;
}
```



18

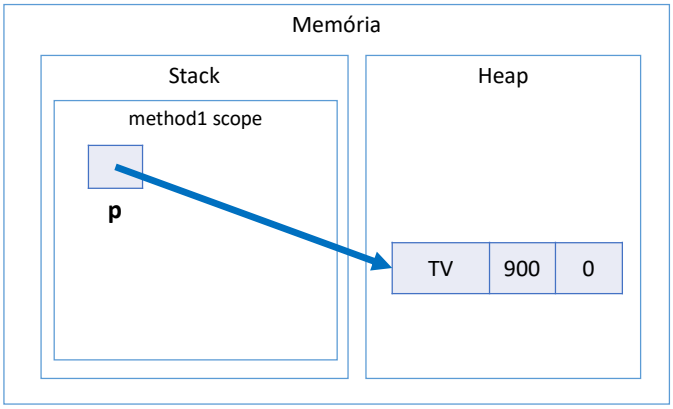
Outro exemplo

```
void method1() {
  Product p = method2();
  System.out.println(p.Name);
}

Product method2() {
  Product prod = new Product("TV", 900.0, 0);
  return prod;
}
```



Outro exemplo

```
void method1() {
  Product p = method2();
  System.out.println(p.Name);
}

Product method2() {
  Product prod = new Product("TV", 900.0, 0);
  return prod;
}
```



Resumo

- Objetos alocados dinamicamente, quando não possuem mais referência para eles, serão desalocados pelo garbage collector
- Variáveis locais são desalocadas imediatamente assim que seu escopo local sai de execução

Vetores - Parte 1

http://educandoweb.com.br

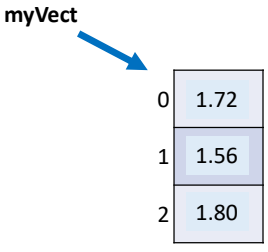
Prof. Dr. Nelio Alves

Checklist

- Revisão do conceito de vetor
- Declaração e instanciação
- Manipulação de vetor de elementos tipo valor (tipo primitivo)
- Manipulação de vetor de elementos tipo referência (classe)
- Acesso aos elementos
- Propriedade length

Vetores

- Em programação, "vetor" é o nome dado a arranjos unidimensionais
- Arranjo (array) é uma estrutura de dados:
  - Homogênea (dados do mesmo tipo)
  - Ordenada (elementos acessados por meio de posições)
  - Alocada de uma vez só, em um bloco contíguo de memória
- Vantagens:
  - Acesso imediato aos elementos pela sua posição
- Desvantagens:
  - Tamanho fixo
  - Dificuldade para se realizar inserções e deleções

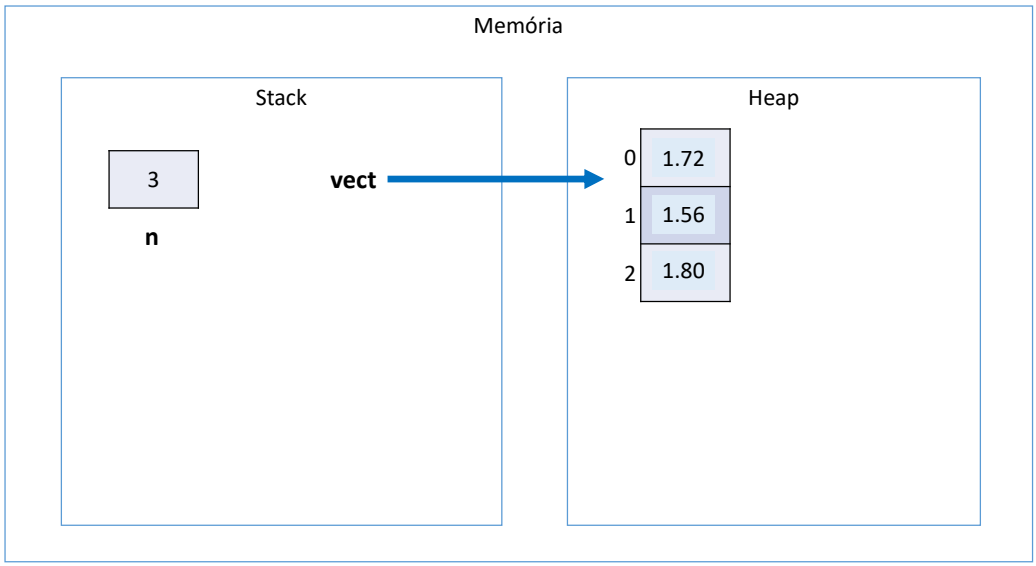


Problema exemplo 1

Fazer um programa para ler um número inteiro N e a altura de N pessoas. Armazene as N alturas em um vetor. Em seguida, mostrar a altura média dessas pessoas.

Exemplo

Input:	**Output:**
3 1.72 1.56 1.80	AVERAGE HEIGHT = 1.69



27

```java
package application;

import java.util.Locale;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

        Locale.setDefault(Locale.US);
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        double[] vect = new double[n];

        for (int i=0; i<n; i++) {
            vect[i] = sc.nextDouble();
        }

        double sum = 0.0;
            for (int i=0; i<n; i++) {
            sum += vect[i];
        }
        double avg = sum / n;

        System.out.printf("AVERAGE HEIGHT: %.2f%n", avg);

        sc.close();
    }
}
```

28

Vetores - Parte 2

http://educandoweb.com.br

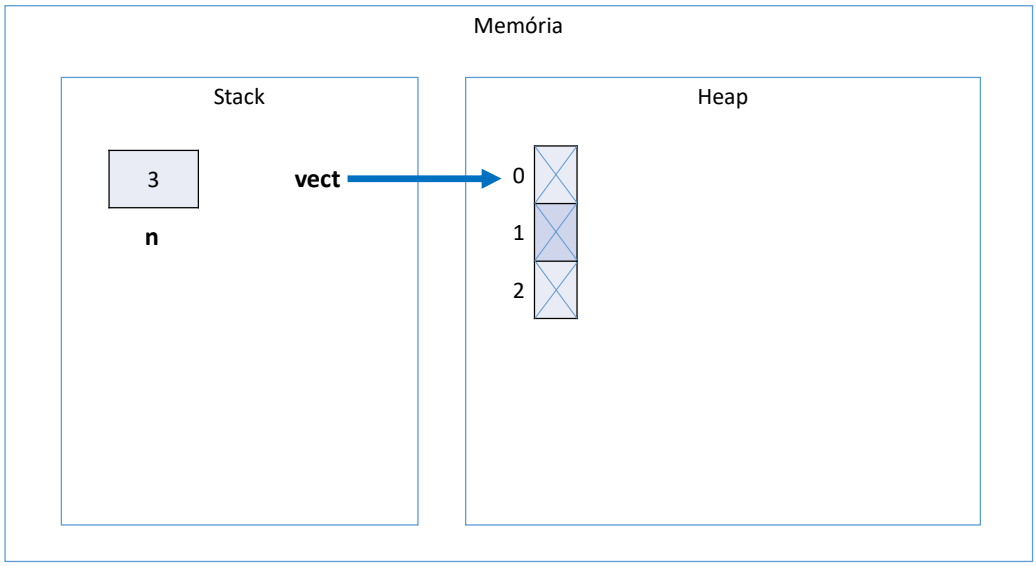
Prof. Dr. Nelio Alves

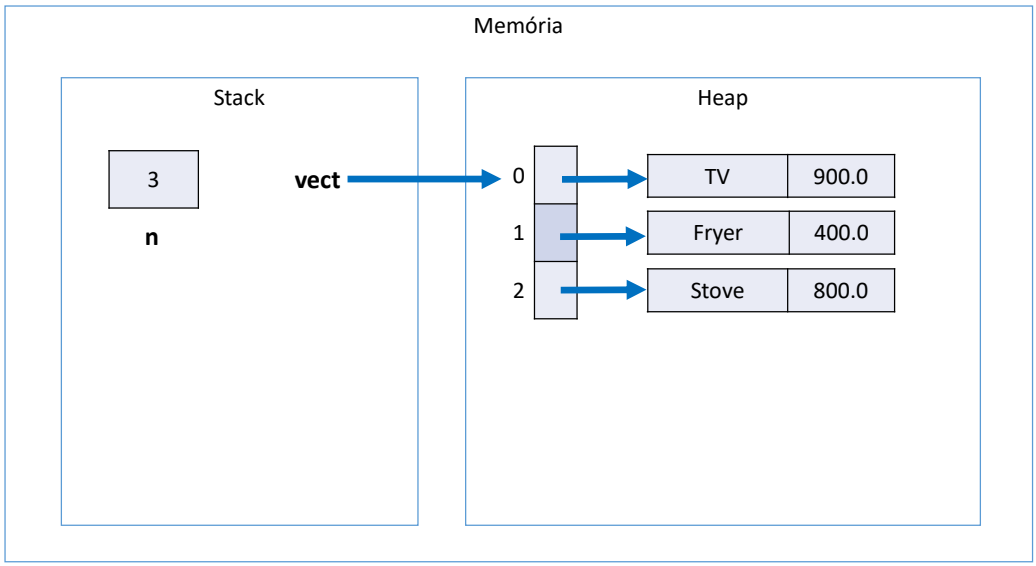
Problema exemplo 2

Fazer um programa para ler um número inteiro N e os dados (nome e preço) de N Produtos. Armazene os N produtos em um vetor. Em seguida, mostrar o preço médio dos produtos.

Exemplo

Input:	Output:
3 TV 900.00 Fryer 400.00 Stove 800.00	AVERAGE PRICE = 700.00





33

```java
package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Product;

public class Program {

	public static void main(String[] args) {

		Locale.setDefault(Locale.US);
		Scanner sc = new Scanner(System.in);

		int n = sc.nextInt();
		Product[] vect = new Product[n];

		for (int i=0; i<vect.length; i++) {
			sc.nextLine();
			String name = sc.nextLine();
			double price = sc.nextDouble();
			vect[i] = new Product(name, price);
		}

		double sum = 0.0;
			for (int i=0; i<vect.length; i++) {
			sum += vect[i].getPrice();
		}
		double avg = sum / vect.length;

		System.out.printf("AVERAGE PRICE = %.2f%n", avg);

		sc.close();
	}
}
```

34

Boxing, unboxing e wrapper classes

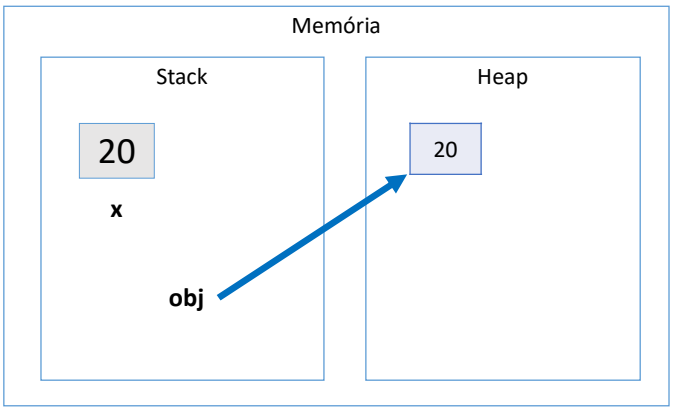
http://educandoweb.com.br

Prof. Dr. Nelio Alves

Boxing

- É o processo de conversão de um objeto tipo valor para um objeto tipo referência compatível

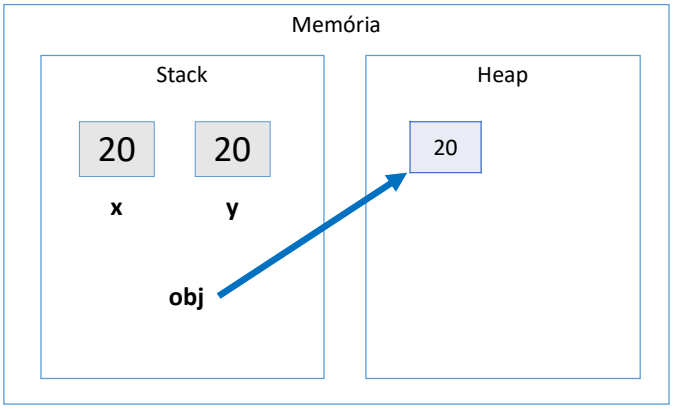
```
int x = 20;

Object obj = x;
```



Unboxing

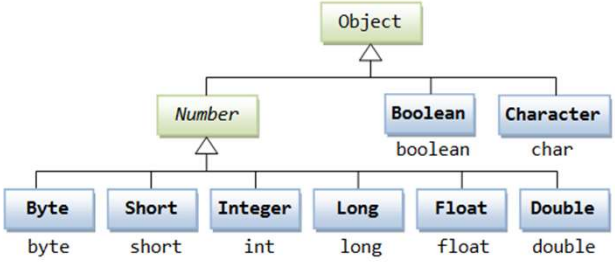
- É o processo de conversão de um objeto tipo referência para um objeto tipo valor compatível

```
int x = 20;

Object obj = x;

int y = (int) obj;
```



Wrapper classes

- São classes equivalentes aos tipos primitivos
- Boxing e unboxing é natural na linguagem
- Uso comum: campos de entidades em sistemas de informação (IMPORTANTE!)
  - Pois tipos referência (classes) aceitam valor null e usufruem dos recursos OO



Demo

```
Integer x = 10;

int y = x * 2;
```

```
public class Product {

    public String name;
    public Double price;
    public Integer quantity;

    (...)
```

39

Laço "for each"

http://educandoweb.com.br

Prof. Dr. Nelio Alves

40

Laço "for each"

Sintaxe opcional e simplificada para percorrer coleções

Sintaxe:

```
for (Tipo apelido : coleção) {
    <comando 1>
    <comando 2>
}
```


Demo

Leitura: "para cada objeto 'obj' contido em vect, faça:"

```
String[] vect = new String[] {"Maria", "Bob", "Alex"};

for (int i=0; i< vect.length; i++) {
   System.out.println(vect[i]);
}

for (String obj : vect) {
   System.out.println(obj);
}
```

Listas - Parte 1

http://educandoweb.com.br

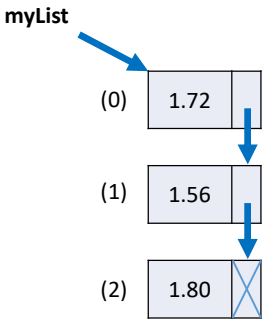
Prof. Dr. Nelio Alves

Checklist

- Conceito de lista
- Tipo **List** - Declaração, instanciação
- Demo
- Referência: https://docs.oracle.com/javase/10/docs/api/java/util/List.html
- Assuntos pendentes:
  - interfaces
  - generics
  - predicados (lambda)

Listas

- Lista é uma estrutura de dados:
  - Homogênea (dados do mesmo tipo)
  - Ordenada (elementos acessados por meio de posições)
  - Inicia vazia, e seus elementos são alocados sob demanda
  - Cada elemento ocupa um "nó" (ou nodo) da lista
- Tipo (interface): List
- Classes que implementam: ArrayList, LinkedList, etc.
- Vantagens:
  - Tamanho variável
  - Facilidade para se realizar inserções e deleções
- Desvantagens:
  - Acesso sequencial aos elementos *



(desenho simplificado)

45

Listas - Parte 2

http://educandoweb.com.br

Prof. Dr. Nelio Alves

46

Demo

- Tamanho da lista: `size()`
- Obter o elemento de uma posição: `get(position)`
- Inserir elemento na lista: `add(obj), add(int, obj)`
- Remover elementos da lista: `remove(obj), remove(int), removeIf(Predicate)`
- Encontrar posição de elemento: `indexOf(obj), lastIndexOf(obj)`
- Filtrar lista com base em predicado:

```
List<Integer> result = list.stream().filter(x -> x > 4).collect(Collectors.toList());
```

- Encontrar primeira ocorrência com base em predicado:

```
Integer result = list.stream().filter(x -> x > 4).findFirst().orElse(null);
```

- Assuntos pendentes:
  - interfaces
  - generics
  - predicados (lambda)


```
package application;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class Program {

public static void main(String[] args) {

	List<String> list = new ArrayList<>();

	list.add("Maria");
	list.add("Alex");
	list.add("Bob");
	list.add("Anna");
	list.add(2, "Marco");

	System.out.println(list.size());
	for (String x : list) {
		System.out.println(x);
	}
	System.out.println("---------------------");
	list.removeIf(x -> x.charAt(0) == 'M');
	for (String x : list) {
		System.out.println(x);
	}
	System.out.println("---------------------");
	System.out.println("Index of Bob: " + list.indexOf("Bob"));
	System.out.println("Index of Marco: " + list.indexOf("Marco"));
	System.out.println("---------------------");
	List<String> result = list.stream().filter(x -> x.charAt(0) == 'A').collect(Collectors.toList());
	for (String x : result) {
		System.out.println(x);
	}
	System.out.println("---------------------");
	String name = list.stream().filter(x -> x.charAt(0) == 'J').findFirst().orElse(null);
	System.out.println(name);
	}
}
```

Exercício de fixação

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Exercício sobre OO e List (Java)

/devsuperior

Fazer um programa para ler um número inteiro N e depois os dados (id, nome e salario) de N funcionários. Não deve haver repetição de id.

Em seguida, efetuar o aumento de X por cento no salário de um determinado funcionário. Para isso, o programa deve ler um id e o valor X. Se o id informado não existir, mostrar uma mensagem e abortar a operação. Ao final, mostrar a listagem atualizada dos funcionários, conforme exemplos.

Lembre-se de aplicar a técnica de encapsulamento para não permitir que o salário possa ser mudado livremente. Um salário só pode ser aumentado com base em uma operação de aumento por porcentagem dada.

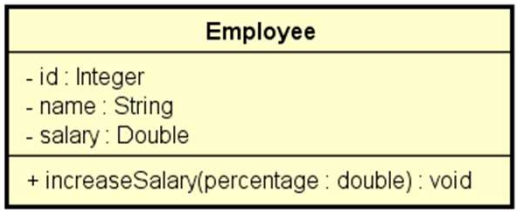
(exemplo na próxima página)

```
How many employees will be registered? 3

Emplyoee #1:
Id: 333
Name: Maria Brown
Salary: 4000.00

Emplyoee #2:
Id: 536
Name: Alex Grey
Salary: 3000.00

Emplyoee #3:
Id: 772
Name: Bob Green
Salary: 5000.00

Enter the employee id that will have salary increase : 536
Enter the percentage: 10.0

List of employees:
333, Maria Brown, 4000.00
536, Alex Grey, 3300.00
772, Bob Green, 5000.00
```

51

```
How many employees will be registered? 2

Emplyoee #1:
Id: 333
Name: Maria Brown
Salary: 4000.00

Emplyoee #2:
Id: 536
Name: Alex Grey
Salary: 3000.00

Enter the employee id that will have salary increase: 776
This id does not exist!

List of employees:
333, Maria Brown, 4000.00
536, Alex Grey, 3000.00
```

52



https://github.com/acenelio/list1-java