Resumo da aula

- Comparator objeto de classe separada
- Comparator objeto de classe anônima
- Comparator objeto de expressão lambda com chaves
- Comparator objeto de expressão lambda sem chaves
- Comparator expressão lambda "direto no argumento"

https://github.com/acenelio/lambda1-java

Programação funcional e cálculo lambda

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Paradigmas de programação

- **Imperativo** (C, Pascal, Fortran, Cobol)
- **Orientado a objetos** (C++, Object Pascal, Java (< 8), C# (< 3))
- **Funcional** (Haskell, Closure, Clean, Erlang)
- **Lógico** (Prolog)
- Multiparadigma (JavaScript, Java (8+), C# (3+), Ruby, Python, Go)

Paradigma funcional de programação

Baseado no formalismo matemático Cálculo Lambda (Church 1930)

	Programação Imperativa	**Programação Funcional**
Como se descreve algo a ser computado (*)	comandos ("como" - imperativa)	expressões ("o quê" - declarativa)
Funções possuem transparência referencial (ausência de efeitos colaterais)	fraco	forte
Objetos imutáveis (*)	raro	comum
Funções são objetos de primeira ordem	não	sim
Expressividade / código conciso	baixa	alta
Tipagem dinâmica / inferência de tipos	raro	comum

Transparência referencial

Uma função possui transparência referencial se seu resultado for sempre o mesmo para os mesmos dados de entrada. Benefícios: simplicidade e previsibilidade.

```
package application;

import java.util.Arrays;

public class Program {

     public static int globalValue = 3;

     public static void main(String[] args) {

          int[] vect = new int[] {3, 4, 5};
          changeOddValues(vect);
          System.out.println(Arrays.toString(vect));
     }

     public static void changeOddValues(int[] numbers) {
          for (int i=0; i<numbers.length; i++) {
               if (numbers[i] % 2 != 0) {
                    numbers[i] += globalValue;
               }
          }
     }
}
```

Exemplo de função que não é referencialmente transparente

Funções são objetos de primeira ordem (ou primeira classe)

Isso significa que funções podem, por exemplo, serem passadas como parâmetros de métodos, bem como retornadas como resultado de métodos.

```
public class Program {

     public static int compareProducts(Product p1, Product p2) {
          return p1.getPrice().compareTo(p2.getPrice());
     }

     public static void main(String[] args) {

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

          list.add(new Product("TV", 900.00));
          list.add(new Product("Notebook", 1200.00));
          list.add(new Product("Tablet", 450.00));

          list.sort(Program::compareProducts);

          list.forEach(System.out::println);
     }
}
```

Utilizamos aqui "method references"

Operador ::

Sintaxe:
`Classe::método`

Tipagem dinâmica / inferência de tipos

```
public static void main(String[] args) {

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

    list.add(new Product("TV", 900.00));
    list.add(new Product("Notebook", 1200.00));
    list.add(new Product("Tablet", 450.00));

    list.sort((p1, p2) -> p1.getPrice().compareTo(p2.getPrice()));

    list.forEach(System.out::println);
}
```

Expressividade / código conciso

```
Integer sum = 0;
for (Integer x : list) {
    sum += x;
}
```

vs.

```
Integer sum = list.stream().reduce(0, Integer::sum);
```

O que são "expressões lambda"?

Em programação funcional, expressão lambda corresponde a uma função anônima de primeira classe.

```
public class Program {

    public static int compareProducts(Product p1, Product p2) {
        return p1.getPrice().compareTo(p2.getPrice());
    }

    public static void main(String[] args) {

        (...)

        list.sort(Program::compareProducts);

        list.sort((p1, p2) -> p1.getPrice().compareTo(p2.getPrice()));

        (...)
```

Resumo da aula

	Programação Imperativa	**Programação Funcional**
Como se descreve algo a ser computado (*)	comandos ("como" - imperativa)	expressões ("o quê" - declarativa)
Funções possuem transparência referencial (ausência de efeitos colaterais)	fraco	forte
Objetos imutáveis (*)	raro	comum
Funções são objetos de primeira ordem	não	sim
Expressividade / código conciso	baixa	alta
Tipagem dinâmica / inferência de tipos	raro	comum

Cálculo Lambda = formalismo matemático base da programação funcional

Expressão lambda = função anônima de primeira classe