

# Funções Lambda

# Funções lambda

- O conceito de função lambda foi adicionado no Java 8. Esse conceito busca adicionar ao Java técnicas de linguagens funcionais (ex: LISP). A grande vantagem de funções lambda é diminuir a quantidade de código necessária para a escrita de alguns métodos, como por exemplo para Threads.
- Em outras palavras, uma função lambda é uma função sem declaração, isto é, não é necessário colocar um nome, um tipo de retorno e o modificador de acesso.
- Funções Lambda têm a seguinte sintaxe:

(argumento) -> (corpo)

- Frequentemente o uso de funções lambda se dá associado ao conceito de generics. O exemplo dos próximos slides ilustrará essa relação

# Interfaces usando generics

```
public interface FuncaoMatematicaDinamica<R> {  
    R aplicar(int value);  
}
```

```
public interface FuncaoMatematicaDinamica2Params<R> {  
    R aplicar(int value1, int value2);  
}
```

- Acima vemos 2 interfaces que fazem uso de generics através do retorno R.

# Métodos com Interfaces como parâmetro

```
public class TesteLambda {
```

Ideologicamente objFuncao **DEVE** implementar a interface **FuncaoMatematicaDinamica** no método fazOperacaoMatematica

```
    public static int fazOperacaoMatematica(int num,  
        FuncaoMatematicaDinamica<Integer> objFuncao) {  
        return objFuncao.aplicar(num);  
    }
```

```
    public static int fazOperacaoMatematica2Params(int n1, int n2,  
        FuncaoMatematicaDinamica2Params<Integer> objFuncao) {  
        return objFuncao.aplicar(n1, n2);  
    }
```

# Funções lambda

```
public static void main(String[] args) {  
    System.out.println(fazOperacaoMatematica(5, (n) -> n * n));  
    System.out.println(fazOperacaoMatematica(7, (n) -> n + 4));  
    System.out.println(fazOperacaoMatematica(7, (n) -> {  
        n = n + 4;    n = n * 3; return n;  
    }));  
    System.out.println(fazOperacaoMatematica2Params( 5, 3, (n1, n2) -> {  
        int potencia=1;  
        for(int i=0; i<n2; i++){  potencia *= n1; }  
        return potencia;  
    }));  
}  
}
```

Quando usamos uma função lambda, ela é aceita como um objeto que implementa a interface indicada. Em `fazOperacaoMatematica2Params` a função lambda implementou a interface `fazOperacaoMatematica2Params`, sendo passados `n1 = 5` e `n2 = 3`. Daí o comportamento do método **aplicar** ficou a carga do código apresentado na função lambda.

A grande vantagem desse tipo de programação é a possibilidade de se dar comportamento dinamicamente a um dado método

# Funções lambda em Listeners

```
botao.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        System.out.println("O botão foi pressionado!");  
        //Realiza alguma ação quando o botão for pressionado  
    }  
});
```

```
botao.addActionListener( (e) -> {  
    System.out.println("O botão foi pressionado, e o código  
    executado utiliza uma função lambda!");  
    //Realiza alguma ação quando o botão for pressionado  
});
```

**Em vermelho classe anônima e em verde função lambda**

# Funções lambda em Arrays

- É possível usar funções lambda em Arrays de forma a deixar mais “elegante” a interação com elementos do array. O exemplo abaixo ilustra esse uso:

```
public class LambdaArray {  
    public static void main(String[] args) {  
        System.out.println("Imprime os elementos da lista!");  
        List<String> list = Arrays.asList("Pedro", "Maria", "João");  
        list.forEach(n -> {  
            System.out.println(n);  
        });  
    }  
}
```

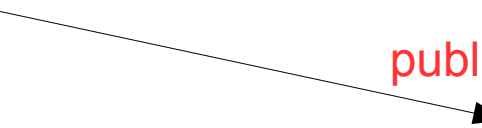
O código ao lado navegará no array e para cada um dos elementos do array será feita a Impressão na tela.

# Uso de Lambda em Arrays – Avançado

```
public class LambdaArray {
```

```
    public static void fazTratamentoString(List<String> list, FuncaoString<String> objFuncao) {  
        list.forEach(n -> {  
            String s = objFuncao.aplicar(n);  
            System.out.println(s);  
        });  
    }
```

```
public interface FuncaoString <R>  
    R aplicar(String value);  
}
```



```
    public static void main(String[] args) {  
        System.out.println("Imprime os elementos da lista!");  
        List<String> list = Arrays.asList("Pedro", "Maria", "João");
```

```
        System.out.println("Colocar Sr ou Sra");  
        fazTratamentoString(list, (n)-> {  
            if (n.charAt(n.length()-1) == 'o')  
                n = "Sr " + n;  
            else  
                n = "Sra " + n;  
            return n;  
        });
```

```
    }  
}
```



# Dúvidas?

