# Java OO
# LAB 5

## Objetivo

O objetivo deste laboratório é para você habituar-se ao básico da linguagem Java.

Neste laboratório faremos uso de objetos Java, objetos de bibliotecas básica, reforçando experiência de programação em Java e outros conceitos básicos como invocação de métodos (de classe e de instância), classes básicas do pacote java.lang

A partir deste laboratório você poderá utilizar sua IDE para realizar as atividades.

Duração prevista: 60 minutos

## Exercícios

Exercício 1: Criação de objetos (Instância de uma classe)
Exercício 2: Métodos de instância e métodos estáticos
Exercício 3: Argumentos passado por valor e por referência
Exercício 4: Escopo de variáveis
Exercício 6: Comparando objetos (30 minutos)
Exercício 7: Método getClass() e o operador instanceof (20 minutos)

## Criando Objetos

1 - Usando sua **IDE** ou editor de arquivo crie a classe **ExemploObjeto** com base na **Listagem-5.1** abaixo.

*ClasseEObjeto.java

```
1  public class ClasseEObjeto {
2
3      public static void main(String[] args) {
4
5          // Para criar uma instância de uma classe use a palavra chave new
6          // Por exemplo, para criar uma instância da classe String
7          // procedemos como segue
8          String strObjeto1 = new String("Este objeto e uma instância da classe String");
9          System.out.println("estado da instância de strObjeto1 = " + strObjeto1);
10
11         // A classe String e uma classe especial que permite a criação de uma
12         // instância pela atribuição de um literal string. Nenhuma outra classe
13         // em java permite este tipo de criação. Além disso para cada literal
14         // string e criada uma única instância desta string
15         String strObjeto2 = "Este objeto é uma instância da classe String";
16         System.out.println("estado da instância de strObjeto2 = " + strObjeto2);
17     }
18 }
```

Listagem 5.1 – ExemploObjeto.java

2 - Compile e execute o programa **ClasseEObjeto**, observe o resultado da execução:

3 - Modifique o programa **ClasseEObjeto.java** para criar outra instância da classe String contendo o literal string **"Eu sou outra instância da classe String"** e imprimir seu estado usando o método **System.out.println(...)**

4 - Com base no programa **ClasseEObjeto.java**, modifique o programa para criar um objeto da classe **Integer** (classe **Wrapper** do tipo primitivo **int**) cujo valor é 20, veja na **Listagem-5.2.**

```
3  public class ClasseEObjeto {
4
5      public static void main(String[] args) {
6
7          String strObjeto1 = new String("Este objeto e uma instância da classe String");
8          System.out.println("estado da instância de strObjeto1 = " + strObjeto1);
9
10
11         String strObjeto2 = "Este objeto é uma instância da classe String";
12         System.out.println("estado da instância de strObjeto2 = " + strObjeto2);
13
14         // cria uma instância de objeto da classe Integer
15         Integer intObjeto1 = new Integer(20);
16         System.out.println("estado da instância de intObjeto1 = " + intObjeto1);
17     }
18 }
```

Listagem 5.2 – ClasseEObjeto.java criando objeto Interger

***Desafio para o Aluno***

1 - Modifique o programa anterior para criar e imprimir uma instância da classe Double e imprimir seu estado.

## Métodos estáticos e métodos de instância (não - estático)

Neste exercício, você aprenderá como invocar métodos **estáticos** (de classe) e **não-estáticos** (de instância) de uma classe. Para invocar métodos estáticos usamos a seguinte forma:

***<Nome da Classe>.<Nome do método estático>***

Por exemplo,

***Integer.parseInt("25"); //parseInt é um método estático da classe Integer***

Um método não-estático (instância) de uma classe só pode ser invocado a partir de uma instância do objeto da classe usando a seguinte forma:

***<Nome da instância de uma classe>.<Nome do método não-estático>***

por exemplo,

***String str = new String("Brazil com z"); //cria instância do objeto***

***str.charAt(0); //chama método de instância charAt()***

***//da classe String através da instância do objeto***

1 - Crie o programa **MetodosInstanciaEEstatico.java** com base na **Listagem-5.3** a abaixo.

```
2  public class MetodosInstanciaEEstatico {
3
4      public static void main(String[] args) {
5
6          //Cria duas instâncias da classe String
7          String strInst1 = new String("Sou uma instância de objeto da classe String");
8          String strInst2 = "Viva com paixão!";
9
10         /* Invoca o método de instância charAt()
11          * através das instâncias da classe String
12          */
13         char x = strInst1.charAt(2);
14         char y = strInst2.charAt(1);
15         char z = strInst2.charAt(0);
16
17         System.out.println("O 3º caracter da strInst1 = " + x);
18         System.out.println("O 2º caracter da strInst2 = " + y);
19         System.out.println("O 1º caracter da strInst2 = " + z);
20
21         //Invoca o método de instância equalsIgnoreCase(...)
22         boolean b = strInst1.equalsIgnoreCase(strInst2);
23         String strInst3 = b ? "Sim" : "Não";
24
25         System.out.println("As variáveis strInst1 " + " e strInst2 tem o mesmo" + " conjunto de caracteres? " + strInst3);
26         //Invoca um método estático valueOf da classe String
27         int i = 23;
28         String strInst4 = String.valueOf(i);
29         System.out.println("valor de strInst4 = " + strInst4);
30
31         //Invoca o metodo estático parseInt da classe Integer
32         String strInst5 = new String("34");
33         int ii = Integer.parseInt(strInst5);
34         System.out.println("valor de ii = " + ii);
35
36     }
37 }
```

Listagem 5.3 – MetodoInstanciaEEstatico.java

2 - Compile, rode o programa e observe o resultado.

***Desafio para o Aluno***

1 - Modifique o programa **MetodosInstanciaEEstatico.java** para que o mesmo contenha as instruções abaixo no final do método **main().**

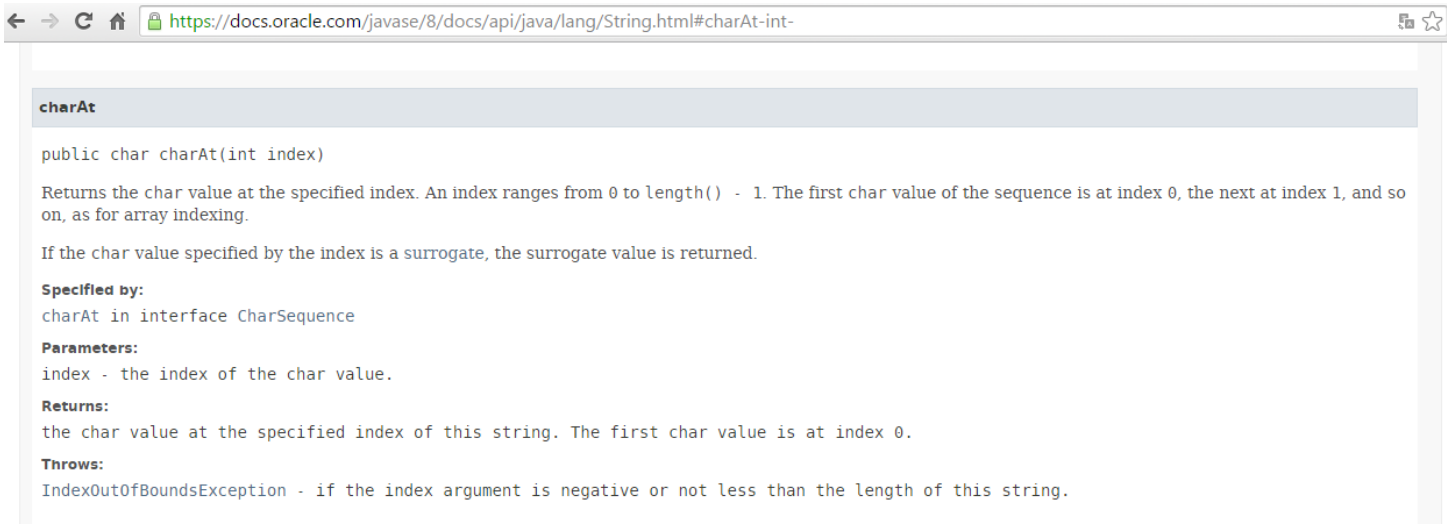
```
/* O seguinte código irá gerar erro de compilação
 *  uma vez que ele tentará invocar um método de instância
 *  através do nome da classe. Corrija o este erro de compilação
 */
char f = String.charAt(2);
```

2 - Compile o programa. O seguinte erro será gerado pelo compilador:

***Cannot make a static reference to the non-static method charAt(int) from the type String***

3 - Corrija o erro de compilação e novamente compile e execute o programa.

**Dica:** Como posso identificar se um método qualquer da biblioteca de Java é estático ou de instância?
Consulte a documentação da **API Java** **https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#charAt-int-** e veja se o método em questão possui em sua assinatura a palavra chave **static**. Veja na figura:




4 - Acrescente o trecho de código abaixo no programa anterior. Consulte na documentação da API se os métodos são estáticos ou de instância e faça a correta invocação dos métodos.

```
// método endsWith()
String str = "Hello";
System.out.println( str.endsWith( "slo" ) );

// método floor()
System.out.println( Math.floor(3.14));

// método isDigit()
System.out.println( "0=" + Character.isDigit('0'));
System.out.println( "A=" +Character.isDigit('A'));
```

## Argumentos de métodos passados por valor e por referência

Neste exercício, você ira praticar o conceito da passagem de parâmetro por valor. Note que os parâmetros de tipos primitivos são passados o valor, enquanto parâmetros do tipo referência são passados a referência. Um array é considerado um tipo referência mesmo que os valores armazenados nele sejam de um tipo primitivo.

**Passagem de parâmetros por valor:**

1. Escreva, compile e execute a **Listagem-5.4, TestaPassagemValor.java**, preste bastante atenção aos comentários.

```
1
2  public class TestaPassagemValor {
3
4      //Método estático na classe
5      public static void teste(int j) {
6          System.out.println("inicia método test e j = " + j);
7
8          //muda valor parâmetro i
9          j = 33;
10
11         System.out.println("termina método test e j = " + j);
12     }
13
14     public static void main(String[] args) {
15
16         int i = 10;
17         System.out.println("Inicia com o I valendo = " + i);
18
19         /* Chama o método teste e passa o valor inteiro como um parâmetro.
20         *  Uma vez que int é um tipo primitivo, este argumento e passado
21         *  por valor.
22         */
23         TestaPassagemValor.teste(i);
24
25         // imprime o valor de i, note que valor de i não muda
26         System.out.println("termina o método main e i = " + i);
27     }
28 }
```

Listagem 5.4 – Passagem de parâmetro por valor

***Desafio para o Aluno:***

Modifique **TestaPassagemValor.java** como segue abaixo e execute a classe.

1 - Crie um segundo parâmetro de tipo primitivo para o método **teste**, exemplo **test(int j, int k)** - você pode escolher qualquer parâmetro de tipo primitivo (tal como long ou boolean).

2 – Agora ao chamar o método **teste()**, passe os valores conforme os parâmetros que o método aguarda receber.

3 - Modifique os métodos **System.out.println(...)** para indicar os valores do primeiro parâmetro e do segundo parâmetro.

**Passagem de parâmetros por referência**

1 - Escreva, compile e execute o programa **TestaPassagemReferencia.java**, preste bastante atenção aos comentários.

```
2  import java.util.Arrays;
3
4  public class TestaPassagemReferencia {
5
6      //Método estático na classe
7      public static void teste(int[] j) {
8
9          System.out.println("inicia método test e j = " + Arrays.toString(j));
10
11         //Muda valor do parâmetro i
12         j[0] = 33;
13         j[1] = 66;
14
15         System.out.println("termina método test e j = " + Arrays.toString(j));
16     }
17
18     public static void main(String[] args) {
19
20         //Criando um array de inteiros
21         int[] i = { 10, 20, 30 };
22
23         //Imprime valor de i
24         System.out.println("inicia metodo main e i = " + Arrays.toString(i));
25
26         /* Chama método teste e passa por referência o array de inteiro como um parâmetro.
27          * Uma vez que um array é do tipo referência, este argumento e passado por referência.
28          */
29         teste(i);
30
31         // imprime o valor de i. Note que o valor de i não muda
32         System.out.println("termina o método main e i = " + Arrays.toString(i));
33     }
34 }
```

Listagem 5.5 – Passagem de parâmetros por referência

2 - Observe que a classe utiliza o método **toString()** da classe **java.util.Arrays** para poder imprimir o conjunto de elementos do Array.

**Escopo de variáveis**

Neste exercício você praticará o conceito de escopo de variável. Você também aprenderá como declarar três tipos de variáveis: **variável estática, variável de instância** é a **variável local.**

1 - Escreva, compile e execute o programa **EscopoDeVariavel.java**, preste bastante atenção aos comentários.

```
2  public class EscopoDeVariavel {
3
4      public static void main(String[] args) {
5
6          int var1 = 10;
7
8          if (var1 < 100) {
9              int var2 = 20;
10         } else {
11             int var2 = 21;
12         }
13
14         // Acesso a var1 é permitido, então não há erro de compilação.
15         System.out.println("valor de var1 = " + var1);
16
17         // Acesso a var2 não é permitido, então erro de compilação será gerado
18         System.out.println("valor de var2 = " + var2);
19     }
20 }
```

Listagem 5.6 – Escopo de variável

2 - Observe que haverá um erro de compilação. Isto era esperado porque você está tentando ter acesso a variável **var2** e ela foi declarada dentro das **{ }** do bloco **if/else,** então o escopo de acesso é somente dentro das **{ }** e não pode usada fora de seu escopo.

***Desafio para o Aluno***

1 - Você deverá corrigir o erro no código.
Uma dica, a variável **var2** deve ser declarada no escopo global.

3 - Escreva, compile e execute o programa **TresTiposDeVariaveis.java**, preste bastante atenção aos comentários.

```
1
2  public class TresTiposDeVariaveis {
3
4      // Exemplo de variável estática
5      static String staticVariable = "Variável de classe, ou estática";
6
7      // Exemplo de variável de instância
8      String instanceVariable = "variável de instância";
9
10     public static void main(String[] args) {
11
12         String localVariable = "variável local";
13
14         System.out.println("Variável estática = " + staticVariable);
15         //System.out.println("Variável de instância = " + instanceVariable);
16         System.out.println("Variável local = " + localVariable);
17     }
18 }
```

Listagem 5.7 – TresTiposDeVariaveis.java

3 - Observe que variáveis de instância não podem ser referenciadas dentro de um contexto estático, o método **main()** é estático.

4 - Comente a linha de código contendo o erro, compile e execute o programa.

5 - Pense em outra forma de resolver este problema se for necessário acessar a variável de instância **instanceVariable**.

### Casting de tipos primitivos e classe Wrapper

Neste exercício você praticará a moldagem (**casting**) de tipos primitivos e como converter primitivos às classes de empacotamento (**Wrapper**) correspondentes e vice versa.

- **Casting** de tipos primitivos.
- Convertendo primitivos para classes **Wrapper**.

## Casting de tipos primitivos

1 - Escreva, compile e execute o programa **CastingPrimitivos.java**, preste bastante atenção aos comentários.

```
2  public class CastingPrimitivos {
3
4      public static void main(String[] args) {
5
6          // casting implicito exemplo 1
7          int numInt = 10;
8          double numDouble = numInt;
9          System.out.println("int " + numInt + " e implicitamente moldado para double " + numDouble);
10
11         // casting implicito exemplo 2
12         int numInt1 = 3;
13         int numInt2 = 2;
14         double numDouble2 = numInt1 / numInt2;
15         System.out.println("numInt1/numInt2 " + numInt1 / numInt2 + " e implicitamente moldado para " + numDouble2);
16
17         // casting explicito exemplo 1
18         double valDouble = 10.12;
19         int valInt = (int) valDouble;
20         System.out.println("double " + valDouble + " e explicitamente moldado para int " + valInt);
21
22         // casting explicito exemplo 2
23         double x = 10.2;
24         int y = 2;
25         int resultado = (int) ( x / y );
26         System.out.println("x/y " + x / y + " e explicitamente moldado para int " + resultado);
27     }
28
29 }
```

Listagem 5.8 – CastingPrimitivos.java

2 - Compile e execute o programa, veja o resultados:

***int 10 e implicitamente moldado para double 10.0***
***numInt1/numInt2 1 e implicitamente moldado para 1.0***
***double 10.12 e explicitamente moldado para int 10***
***x/y 5.1 e explicitamente moldado para int 5***

No primeiro exemplo ocorre uma conversão explicita dos tipos, ou seja, o Java o faz automaticamente.
No segundo exemplo ocorre uma divisão de números inteiros resultado em um número inteiro que é transformado em double.
Já no terceiro exemplo há uma conversão explicita de um double para um int.
Já no quarto exemplo há uma divisão de double/int resultando em um double. Neste caso dizemos que está ocorrendo uma promoção numérica ou um alargamento do tipo.

*Desafio para o Aluno*

1 - Modifique o programa para fazer casting dos tipos **long ->byte, float->short, int-> char.**

## Convertendo primitivos para classes Wrapper

1. Escreva, compile e execute o programa **CastingPrimitivos.java**, preste bastante atenção aos comentários.

```
2  public class PrimitivaParaWrapper {
3
4      public static void main(String[] args) {
5
6          //cria uma instância de objeto Integer
7          Integer intObjeto = new Integer(7801);
8
9          // Converte de Integer para primitivo int usando método intValue()
10         int intPrimitiva = intObjeto.intValue();
11         System.out.println("int intPrimitiva = " + intPrimitiva);
12
13         // Usando método estático da classe empacotadora Integer
14         // para converter uma String para o tipo primitivo int
15         String strInt = "65000";
16         int intConvertida = Integer.parseInt(strInt);
17         System.out.println("int intConvertida = " + intConvertida);
18
19         // Converte int primitivo para tipo Integer
20         Integer intObjeto2 = new Integer(intConvertida);
21         System.out.println("Integer intObjeto2 = " + intObjeto2);
22     }
23 }
```

Listagem 5.9 – *PrimitivaParaWrapper.java,* converte de primitivos para empacotadora e vsv

2 - Compile e rode o programa, observe e procure entender o resultado.

3 - Modifique o programa da **Listagem-5.9** para criar uma variável do tipo **Long** e convertê-la em seu tipo primitivo **long**, depois mostre seu valor.

4 – A partir do **Java 5.0** estas conversões ocorrem implicitamente, pois surgiu o recurso de **Autobox** (empacotar) e **AutoUnbox** (desempacotar). Modifique a **Listagem-5.9** para que fique como na **Listagem-5.10**. Perceba como isso torna menos burocrático o processo de conversão de primitivos para **Wrapper** e vice-versa.

```java
public class AutoBoxUnbox {

    public static void main(String[] args) {

        //cria uma instância de objeto Integer, autobox
        Integer intObjeto = 7801;

        //Converte de Integer para primitivo int, auto-unbox
        int intPrimitiva = intObjeto;
        System.out.println("int intPrimitiva = " + intPrimitiva);

        // Usando método estático da classe empacotadora Integer
        // para converter uma String para o tipo Integer, autobox
        String strInt = "65000";
        Integer intConvertida = Integer.parseInt(strInt);
        System.out.println("int intConvertida = " + intConvertida);

        // Converte Integer para primitivo int, autoUnbox
        int intPrimitiva2 = intConvertida;
        System.out.println("Integer intObjeto2 = " + intPrimitiva2);
    }
}
```

Listagem 2.10 - AutoBoxUnbox.java

## Comparando objetos

1 - Escreva, compile e execute o programa **TestaIgualdadeObjeto.java**, preste bastante atenção aos comentários.

```java
public class TestaIgualdadeObjeto {

    public static void main(String[] args) {

        //Declara duas variáveis do tipo String, str1 e str2
        String str1, str2;

        //Observerm que ambas as variaveis apontam para o mesmo objeto.
        str1 = "Viver sem Deus...não é viver!";
        str2 = str1;

        //Mostra o valor das variáveis str1 e str2
        System.out.println("String1: " + str1);
        System.out.println("String2: " + str2);

        /* O operador "==" quando é usado entre variáveis de referência
         * se retornar true significa que ambas as variáveis apontam para
         * mesma instância de um objeto, e se retornar false as variáveis
         * apontam para instâncias diferentes.
         */

        //Verifica se str1 e str2 apontam para a mesma instância de um objeto
        System.out.println("Mesmo objeto? " + ( str1 == str2 ));

        // Reinicializa variável str2. Ela agora aponta para um nova
        // instância de objeto String
        str2 = new String(str1);

        // Mostra valor das variáveis str1 e str2
        System.out.println("String1: " + str1);
        System.out.println("String2: " + str2);

        //Verifica se str1 e str2 apontam novamente para a mesma instância de um objeto
        System.out.println("Mesmo objeto? " + ( str1 == str2 ));

        //Checa se str1 e str2 tem o mesmo valor
        System.out.println("Mesmo valor? " + str1.equals(str2));
    }
}
```

Listagem 5.11 – TestaIgualdadeObjeto.java

2 - Compile e execute o programa.

***Desafio para o Aluno***

1 - Crie um programa como da **Listagem-5.11** para criar e comparar dois objetos da classe **Integer**.

### Método getClass() e o operador instanceof

Neste exercício você usará o método **getClass()** da classe **Object** para encontrar uma instância de um objeto fora da classe. Você aprenderá também como usar o operador **instanceOf** para testar se uma instância de objeto qualquer é de um tipo particular de classe.

## Usando método getClass()

1 - Escreva, compile e execute o programa **TestaGetClass.java**, preste bastante atenção aos comentários.

```
2  public class TesteGetClass {
3
4      public static void main(String[] args) {
5
6          //Cria instância de objeto String
7          String str1 = "A vida e para ser vivida... com Deus!";
8
9         /* Encontrar informação externa de uma instância de String
10         * via método getClass(). Note que ele retorna uma instância
11         * de objeto da classe Class
12         */
13         Class str1Class = str1.getClass();
14         System.out.println("A classe de str1 e uma instância de " + str1Class);
15
16         // O nome da classe da instância de objeto Class.
17         String str1ClassName = str1Class.getName();
18         System.out.println("Nome da classe e " + str1ClassName);
19
20         // Cria instância de objeto Integer
21         Integer int1 = new Integer(34);
22
23         // Encontrar informação externa de uma instância de Integer
24         // via método getClass(). Note que ele retorna uma instância
25         // de objeto da classe Class
26         Class int1Class = int1.getClass();
27         System.out.println("A classe de int1 é uma instância de " + int1Class);
28
29         // O nome da classe da instância de objeto Class.
30         String int1ClassName = int1Class.getName();
31         System.out.println("Nome da classe é " + int1ClassName);
32     }
33 }
```

Listagem 2.12 – TestaGetClass.java

2 - Modifique o programa para criar uma instância de objeto da classe **java.util.Date** e mostre informações desta classe.

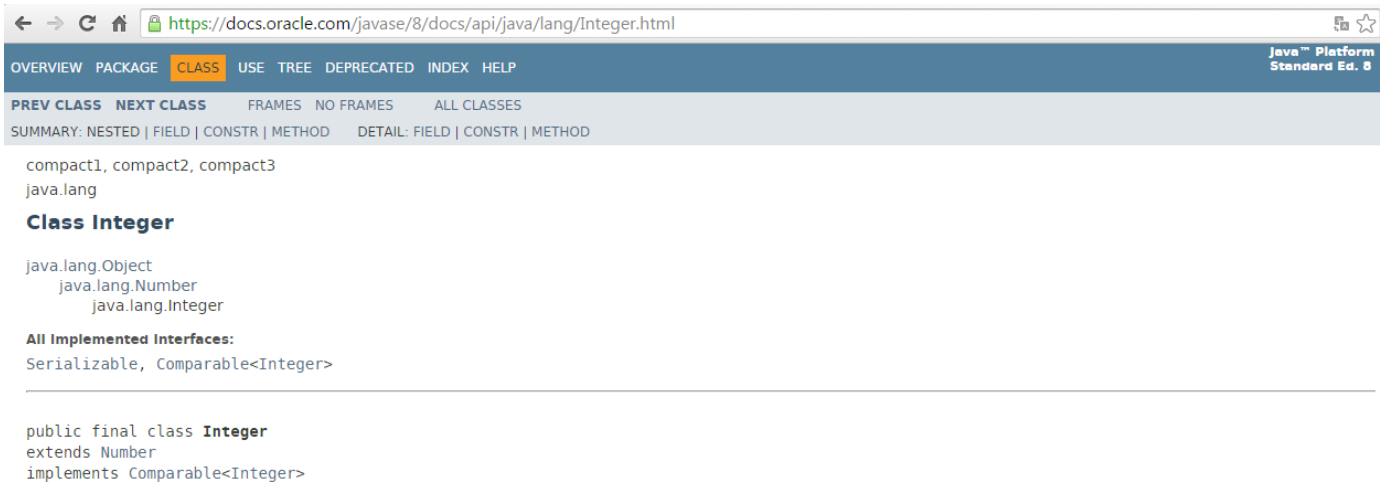
## Usando operador instanceOf

1 - Escreva, compile e execute o programa **TestaInstanceOf.java**, preste bastante atenção aos comentários.

```
2  public class TestaInstanceOf {
3
4      public static void main(String[] args) {
5
6          //Criar objeto String
7          String str1 = "Aprenda Java em Dez anos!";
8          Integer int1 = new Integer(40);
9
10         // Checa se str1 é do tipo String usando operador instanceof.
11         // Checa também se é do tipo Object.
12         boolean b1 = str1 instanceof String;
13         System.out.println("str1 é String: " + b1);
14         boolean b2 = str1 instanceof Object;
15         System.out.println("str1 é Object: " + b2);
16
17         // Checa se int1 é do tipo Integer usando operador instanceof.
18         // Checa também se é do tipo Object.
19         b1 = int1 instanceof Integer;
20         System.out.println("int1 é do tipo Integer: " + b1);
21         b2 = int1 instanceof Object;
22         System.out.println("int1 é Object: " + b2);
23         b2 = int1 instanceof Number;
24         System.out.println("int1 é do tipo Number: " + b2);
25     }
26 }
```

Listagem 2.13 – TestaInstanceOf.java

2. Consultando a documentação da **API Java**, você vai observar que classe **Integer** é uma classe filha da classe **java.lang.Number**, e que esta é filha de **java.lang.Object**. Por isto o teste de **int1 instanceof Number retorna true.**




3 - Modifique o programa para criar uma instância de objeto da classe **Long** e mostre que o objeto é do tipo **Object, Number** e da própria classe **Long**.