Linguagem de Programação II

Conceitos Básicos 2

Universidade do Estado do Rio de Janeiro-UERJ
Instituto de Matemática e Estatística-IME
Ciência da Computação
Professor: Alexandre Sztajnberg

Classe String

- *String* é uma classe Java
  - Por isso começa com letra maiúscula
- Variáveis do tipo *String* guardam referências a objetos contendo uma sequências de caracteres.
  - Sequencia é grudada em um array interno
- O compilador Java e a JVM dão tratamento especial à classe String
  - Facilita o programador em muitos aspectos
  - Por permitir coisas menos “ortodoxas” em OO, ele não é muito bem visto pelos puristas
    - O mesmo para os tipos primitivos

Declaração de objetos String

- Existem 2 formas de declarar um objeto da classe *String*
  - `String nome = "Alexandre";`
  - `String nome = new String("Alexandre"); // instanciando um objeto`
- Em princípio o efeito é o mesmo ...
  - Uma referência para um "futuro" objeto String é criado
  - O objeto é criado
    - Se foi usado "Alexandre", a cadeia de caracteres interna ao objeto é iniciada com "Alexandre"
    - Se foi usado new String ("Alexandre"), o construtor também é iniciado com "Alexandre"
- Mas ...
  - A criação de um objeto String com "=" é *otimizado*

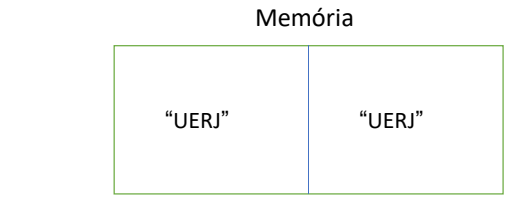
Classe String: declaração com *new*

- **Ao se comparar dois objetos com "==" estamos comparando as REFERÊNCIAS!**
  - ... e não os objetos ...
  - Como fazer a comparação? Veja a documentação da classe *String*.
  - Qual o resultado do código abaixo?

```
public class ExemploString1{
  public static void main(String[] args){
    String x = new String("UERJ");
    String y = new String("UERJ");
    if (x==y){
       System.out.println("São iguais");
    } else {
       System.out.println("São diferentes");
    }
  }
}
```

Classe String

- A resposta é “São diferentes”.
- *String* é uma classe e não um tipo primitivo.
- As variáveis, x e y, são referências
  - Apontam para um endereço de memória e não para um valor



- Assim, estamos comparando o endereço 1000 com o endereço 1001
  - `São diferentes.`
- Então como comparar Valores?

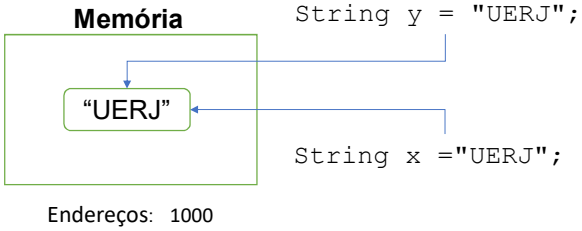
Classe String: declaração com “=“

- Ao se declarar uma variável *String* usando o “açúcar sintático” =
  - O uso de memória é otimizado. **MUITO CUIDADO**
  - Observe que depois de criada e inicializada, o conteúdo de um objeto String não pode ser mudado!
    - Nem acrescentar? Nem mudar a “caixa”? ... **Não!**
    - Mesmo declarado com new ...
    - Esta otimização se vale disso ... **Veja a documentção!**

```
public class ExemploString2{
  public static void main(String[] args){
     String x ="UERJ";
     String y = "UERJ";
     if (x==y) {
        System.out.println("São iguais");
     } else {
        System.out.println("São diferentes");
     }
  }
}
```

Classe String

- Quando declaramos uma variável String, usando "="
  - O conteúdo da inicialização é verificado
  - Se já existir um objeto String com o mesmo valor, o objeto é reusado
  - Ou seja, as duas variáveis apontam para o mesmo objeto na memória



- As duas variáveis possuem a mesma referência de memória(1000)
- Isso acontece pois o conteúdo de ambas é o mesmo ... E não pode mudar...

- A JVM tem uma região da memória especial para armazenamento de String nestes casos, chamada de Java String Pool.
- Com isso se torna mais fácil varrer a memória e encontrar o endereço com o mesmo conteúdo.

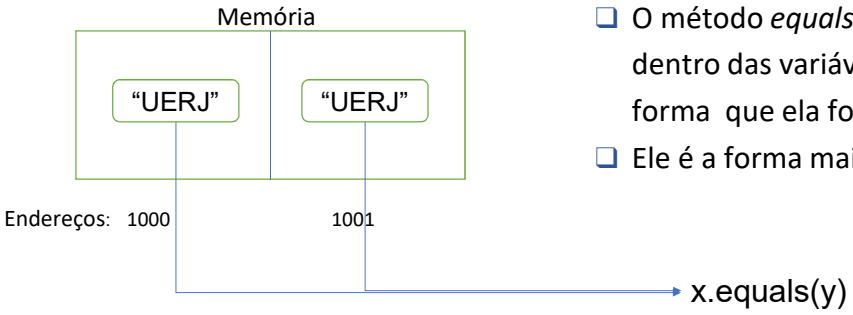
Classe String: comparando "do jeito certo"

- A classe String oferece vários métodos
- Um deles é o "equals" ...

```
public class ExemploString3{
  public static void main(String[] args){
     String x ="UERJ";
     String y = "UERJ";
     if (x.equals(y)) {
        System.out.println("São iguais");
     } else {
        System.out.println("São diferentes");
     }
  }
}
```

Classe String

- No exemplo acima a comparação é feita usando o valor contido que cada objeto



- O método *equals* compara o conteúdo de dentro das variáveis independente da forma que ela foi declarada.
- Ele é a forma mais segura de comparação

String: alguns métodos

- **Atenção! Sempre que o método "devolve" um String, na verdade, outro objeto é criado para produzir o resultados ...** Por que mesmo?

charAt (int index)	Retorna o valor do char na posição especificada em *index* especificado
compareTo (String str)	Compara a strig atual com a *str*
concat (String str)	Concatena a cadeia do objeto sendo chamado com a cadeia de *str*
endWith (String sufixo)	Observa se a String termina com o valor especificado
equalsIgnoreCase (String str)	Compara ignorando maiúsculo e minúsculo
indexOf (int ch)	Retorna o índice da primeira ocorrência do caractere passado
isEmpty ()	Retorna se a String esta vazia ou não
lenght ()	Retorna o tamanho da String
split (String str)	Divide a String (procure os detalhes na documentação)
toUpperCase()	Retorna a String em maiúsculo
toLowerCase()	Retorna a String em minúsculo

Classe String

```
public class ExemploString4{
    public static void main(String[] args){
          String x ="uerj";
          String y = "UERJ";
            System.out.println(x.charAt(3));
            System.out.println(x.compareTo(y));
            System.out.println(x.concat(y));
            System.out.println(x.isEmpty());
            System.out.println(x.toUpperCase());

    }
}
```

Operadores Aritméticos

- Operações fundamentais da matemática entre duas variáveis
  - Soma +
  - Subtração –
  - Divisão /
  - Multiplicação *
  - Modulo(resto da divisão) %

Operadores Aritméticos

```
public class ExemploOperadores {
        public static void main(String[] args) {
            int numero1 = 36;
            int numero2 = 6;
            //soma
            System.out.println(numero1 + numero2);
            //subtração
            System.out.println(numero1 - numero2);
            //divisão
            System.out.println(numero1 / numero2);
            //multiplicação
            System.out.println(numero1 * numero2);
            //modulo
            System.out.println(numero1 % numero2);
    }
 }
```

Operadores Relacionais

- verificam se o valor ou o resultado da expressão lógica à esquerda é igual ou diferente ao da direita. Estes são:
  - Igualdade ==
  - Diferença !=
  - Maior >
  - Menor <
  - Maior e Igual >=
  - Menor e Igual <=

Operadores Relacionais

```
public class ExemploOperadores {
   public static void main(String[] args) {
     int numero1 = 36;
     int numero2 = 6;

     System.out.println(numero1 == numero2); //igualdade

     System.out.println(numero1 != numero2); //diferença

     System.out.println(numero1 > numero2);  //maior

     System.out.println(numero1 < numero2); //menor

     System.out.println(numero1 >= numero2); //maior e igual

     System.out.println(numero1 <= numero2); //menor e igual
  }
}
```

Operadores Lógicos

- Eles representam o recurso que nos permite criar expressões lógicas maiores a partir da junção de duas ou mais expressões. Eles são:
  - E &&
  - Ou ||
- Eles são usados na tabela verdade:

&&			||		
operador1	operador2	resultado	operador1	operador2	resultado
Verdadeiro	Falso	Falso	Verdadeiro	Falso	Verdadeiro
Falso	Verdadeiro	Falso	Falso	Verdadeiro	Verdadeiro
Falso	Falso	Falso	Falso	Falso	Falso
Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro

Operadores Lógicos

```
public static void main(String[] args) {
    int numero1 = 36;
    int numero2 = 6;
    int numero3 =36;
    int numero4 =6;
    System.out.println(numero1 == numero2 && numero2 == numero4);
    System.out.println(numero1 == numero3 && numero2 == numero4);
    System.out.println(numero1 == numero2 && numero2 == numero3);
    System.out.println(numero1 == numero2 || numero2 == numero4);
    System.out.println(numero1 == numero3 || numero2 == numero4);
    System.out.println(numero1 == numero2 || numero2 == numero3);
}
```

Operadores Unários

- Os operadores unários são conhecidos como “de incremento”
  - Executados em cima de um operando fazendo a operação desejada. Eles são:
  - Pré-Incremento. Antes de fazer qualquer operação com o operando ele executa a ação de incremento/decremento sobre o próprio operando.
  - Pós-incremento. Faz antes todas as operações com o operando e depois executa a ação de incremento/decremento
  - Os operandos são:
    - `++operando1 (incremento)`
    - `--operando1 (decremento)`

Operadores Unários

```
public class ExemploOperadores {
   public static void main(String[] args) {
      int numero1 = 36; int numero2 = 6;  //pré-incremento

      System.out.println(numero1 * --numero2);
      System.out.println(numero2);
      System.out.println(++numero1 * numero2);
      System.out.println(numero1);

      numero1 = 36; numero2 = 6;          //pós-incremento
      System.out.println(numero1 * numero2--);
      System.out.println(numero2);
      System.out.println(numero1++ * numero2);
      System.out.println(numero1);
   }
}
```

Operador Ternário

Sintaxe: [expressão booleana] ? [se verdadeiro] : [se falso]

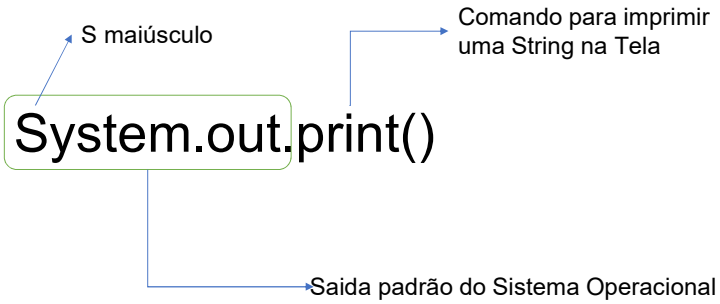
```
public static void main (String[] args) {
  int numero1 = 36;   int numero2 = 6;

  System.out.println (numero1 > numero2 ?
                      "essa condição é verdadeira" :
                      "essa condição é falsa");

  System.out.println (numero1 < numero2 ?
                       "essa condição é verdadeira" :
                       "essa condição é false");
}
```

Classe System

- Representa o sistema operacional
  - Contém 3 objetos
  - **err**, da classe PrintStream, representa a saída padrão para mensagens de erro do sistema.
  - **in**, da classe InputStream, representa a entrada padrão do sistema, que pode ser, por exemplo, um teclado ou um arquivo de texto.
  - **out**, da classe PrintStream, representa a saída padrão de mensagens da aplicação, geralmente a console, mas pode ser redirecionada para um arquivo

Comandos Output

- Usando a classe System e o canal de saída padrão já incluído nesta classe
- O objeto *out* é da classe PrintStream
  - Útil para imprimir strings, não só na saída padrão



Comandos Output

- println(). Mostra a mensagem na tela e pular uma linha
- printf() usado para formatar a saída de dados
  - %d-> inteiros
  - %f -> floats
  - %b-> booleans
  - %c -> char
- Dentro do comando Output é possível inserir caracteres que ajudam na formatação da String a ser impressa
  - \n ->Pula Linha- Posiciona o cursor de tela no início da próxima linha
  - \t -> Tabulação horizontal- Move o cursor de tela para a próxima parada de tabulação
  - \r->Posiciona o cursor da tela no início da linha atual –sobre escrevendo a linha anterior
  - \\-> caso queira imprimir \
  - \”-> para imprimir aspas

Comando Output

```
public class ExemploOutput {
   public static void main(String[] args) {
      int numero1 =36;
      System.out.print("Sem pular linha ...");
      System.out.println(" pulando linha");
      System.out.printf("total = %d bananas", numero1);
      System.out.println("\n linha 1 \n linha2");
      System.out.println("espaço 1 \t espaço2");
   }
}
```

Conversão de tipos

- Para converter tipos usamos a classe relativa ao tipo que irá ser convertido.
- Temos as seguintes classes:
  - Integer -> relativa aos inteiros
  - Double -> relativa ao tipo double
  - Float -> relativa ao tipo float
  - Boolean -> relativa ao tipo boolean
- Essas classes possuem métodos que são responsáveis pela conversão de tipos. Esses métodos são:
  - parseInt(string)
  - toString
  - valueOf(string).intValue()

Conversão de tipos

```
public class ExemploConversao {
    public static void main(String[] args){
      //De inteiro para String
        int i = 42;
        String str3 = Integer.toString(i);

      //De String para Inteiro
        String str4 = "25";
        int x = Integer.valueOf(str).intValue();
        //ou
        String str5 = "12";
        int y = Integer.parseInt(str);
        System.out.println(y);
    }}
```

Argumentos (ou parâmetros) por linha de Comando

- O método *main* tem um argumento de entrada `String[]`
  - Ele serve para passar dados ao programa através da linha de comando
  - `(String [] argumentos)`
    - Um array de objetos da classe String
    - Chamado "argumentos"
    - `argumentos[0], argumentos [1] ...`
- Ao executar o programa todas as coisas escritas após o nome da classe na linha de comando são argumentos.
- Os argumentos são limitados pelo espaço.
- Exemplo de uso:
  - `java ExemploLinhaDeComando arg0 arg1 arg2 arg3`
  - 4 argumentos
  - O array de String terá 4 elementos

Argumentos por linha de comando

```
public class ExemploLinhaDeComando{
  public static void main(String[] args) {
     System.out.println("Os parâmetros são:");
     for (int i = 0; i < args.length; i++)
        System.out.println(args[i]);
  }
}
```

- Para usar a linha de comando :
  - Abra o terminal (prompt de comando);
  - `javac ExemploLinhaDeComando.java`
  - `java ExemploLinhaDeComando casa carro joias`

Argumentos por linha de comando

```
public class ExemploLinhaDeComando{
  public static void main(String[] args) {
     System.out.println("Os parâmetros são:");
     for (int i = 0; i < args.length; i++)
        System.out.println(args[i]);
  }
}
```

- Ok...
  - Mas vou sempre querer receber *Strings*?
  - **E se eu precisar de um *int* ou de um *double*?**

- Para usar a linha de comando :
  - Abra o terminal (prompt de comando);
  - `javac ExemploLinhaDeComando.java`
  - `java ExemploLinhaDeComando casa carro joias`

Conversão de tipos

- Para converter tipos usamos a classe wrapper relativa ao tipo que irá ser convertido.
- Veja a documentação
  - Integer, relativa aos inteiros
  - Double, relativa ao tipo double
  - Float, relativa ao tipo float
  - Boolean, relativa ao tipo boolean
- Classes wapper possuem métodos para conversão.
  - int `parseInt(string)`
  - `valueOf(string).intValue()`

public static **int** **parseInt**(String s) **throws** NumberFormatException

Parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value or an ASCII plus sign '+' ('\u002B') to indicate a positive value. The resulting integer value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseInt(java.lang.String, int) method.

Parameters:
s - a String containing the int representation to be parsed
Returns:
the integer value represented by the argument in decimal.
Throws:
NumberFormatException - if the string does not contain a parsable integer.

Conversão de tipos

```
public class ExemploConversao {
  public static void main(String[] args){
    //De inteiro para String
    int i = 42;
    String str3 = Integer.toString(i);

    //De String para Inteiro
    String str4 = "25";
    int x = Integer.valueOf(str).intValue();
    //ou
    String str5 = "12";
    int y = Integer.parseInt(str);
    System.out.println(y);
  }
}
```

Exemplo, linha de comando

- Na chamada do programa na linha de comandos

```
alexszt@VM11:~$ Area 2.5 5
alexszt@VM11:~$ Area = 12.5

alexszt@VM11:~$ java Area 34
alexszt@VM11:~$

alexszt@VM11:~$ java Area 32 3H
Exception in thread "main" java.lang.NumberFormatException: For input string: "3H"
  at java.base/jdk.internal.math.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:2054)
  at java.base/jdk.internal.math.FloatingDecimal.parseDouble(FloatingDecimal.java:110)
  at java.base/java.lang.Double.parseDouble(Double.java:549)
  at Area.main(Area.java:7)
```

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

    //verifica se foi passado 2 argumentos
    if(args.length==2) {
     double a=Double.parseDouble(args[0]);
     double b=Double.parseDouble(args[1]);
     double area = a * b;

     System.out.println("Area = " +area);
    }
  }
}
```

Input usando fluxo de entrada de System.in

- Mais trabalhoso de usar que o dispositivos de saída
- Precisa se ajustar ao dispositivo primitivo de entrada
- O mesmo roteiro vai ser usado para ler do sistema de arquivos ou de um socket de rede
  - **Por isso estamos vendo esta alternativa primeiro ...**

```
// primeiro embrulhamos o fluxo primitivo de entrada em um leitor de bytes
InputStreamReader inStream;
inStream = new InputStreamReader(System.in);

// depois embrulhamos esse fluxo em um outro que lê Strings
BufferedReader inData;
inData = new BufferedReader(inStream)
```

Usando o dispositivo de entrada

- Podemos usar, por enquanto como receita
- Em uma linha resolve ...
- E agora podemos ler strings ...
  - readLine é o valor retornado na classe BufferedReader
  - readLine vai na janela do System.in e recebe como entrada o que o usuário digitou

```
//Criando o dispositivo em uma instrução
inData = new BuffredReader(new InputStreamReader(System.in));

String umaLinha;

// Armazena uma linha do texto em umaLinha
umaLinha = inData.readLine();
```

Entrada interativa

- Como o usuário sabe o que digitar?
  - O programa (você) informa o usuário System.out.print("informe...");

```
BufferedReader inData;
inData = new BufferedReader(new InputStreamReader(System.in));
String nome;

System.out.print("Digite seu nome: ");

nome = inData.readLine();
```

Exemplo

```
import java.io.*;

public class Area2 {
  public static void main(String[] args) throws IOException {
    BufferedReader inData;
    inData = new BufferedReader(new InputStreamReader(System.in));
    String aux;
    double a,b,area;

    System.out.print("Digite um lado: ");
    aux = inData.readLine();
    a = Double.parseDouble(aux);

    System.out.print("Digite outro lado: ");
    aux = inData.readLine();
    b = Double.parseDouble(aux);

    area = a * b; System.out.println("Area = " +area);
  }
}
```

```
> java Area2
> Digite um lado: 2.5
> Digite outro lado: 5
> Area = 12.5
```

Exemplo

```
import java.io.*;
public class Area2 {
  public static void main(String[] args) throws IOException {
    BufferedReader inData;
    inData = new BufferedReader(new InputStreamReader(System.in));
    String aux;
    double a,b,area;

    System.out.print("Digite um lado: ");
    aux = inData.readLine();
    a = Double.parseDouble(aux);

    System.out.print("Digite outro lado: ");
    aux = inData.readLine();
    b = Double.parseDouble(aux);

    area = a * b; System.out.println("Area = " +area);
  }
}
```

```
> java Area2
> Digite um lado: 2.5
> Digite outro lado: 5
> Area = 12.5
```

Classe *Scanner*

- Dentro do pacote de classes java.util
  - Facilita a entrada de dados
  - Evita a conversão de tipos
  - Também encapsula (ou embrulha) a classe *System.in*
- Métodos da classe Scanner:
  - String next(), retorna uma cadeia de caracteres simples, ou seja, que não usa o caractere espaço em branco;
  - double nextDouble(), retorna um número em notação de ponto flutuante normalizada em precisão dupla
  - boolean hasNextDouble(), retorna true se o próximo dado de entrada pode ser interpretado como um valor double;
  - int nextInt(), retorna um número inteiro;
  - boolean hasNextInt(), retorna true se o próximo dado de entrada pode ser interpretado como um valor int;
  - String nextLine(), retorna uma cadeia de caracteres;
  - long nextLong(), retorna um número long.

Comandos input

```
import java.util.Scanner;

public class ExemploInput {
  public static void main(String[] args){
    Scanner teclado = new Scanner(System.in);

    System.out.println("Digite um numero:");
    int num = teclado.nextInt();

    System.out.println("Digite um numero Ponto flutuante:");
    double nDouble = teclado.nextDouble();
    teclado.nextLine();

    System.out.println("Digite um frase:");
    String frase = teclado.nextLine();

    System.out.println("Echo: " + num + "\n" + nDouble + "\n" + frase);
  }
}
```

Preciso importar a classe

Instância o objeto 1 vez só

Comandos Input

- A classe Scanner emprega um *buffer* de entrada para intermediar o dispositivo de entrada de dados e o programa
- Podem haver erros nos dados lidos caso o buffer não esteja vazio e o programa solicite outro dado, por isso é importante esvazia-lo ou gerenciar o fluxo de dados.
- Para esvaziar o buffer usamos:
  - objetoDaClasseScanner.nextLine(), usamos essa linha entre os 2 comandos de entrada assim o buffer é esvaziado
- Existem várias outras alternativas para fazer uso da classe Scanner e gerenciar o buffer de entrada. Consulte e documentação e exemplos na literatura.
  - Exemplo: https://www.devmedia.com.br/como-funciona-a-classe-scanner-do-java/28448

Comando Input

```
import java.util.Scanner;
public class ExemploInput {
    public static void main(String[] args){

        Scanner teclado = new Scanner(System.in);

        System.out.println("Digite um numero:");
        int numero = teclado.nextInt();

        System.out.println("Digite uma Frase:");

        teclado.nextLine();

        String frase = teclado.nextLine();
        System.out.println("O que você digitou foi: "+numero+"\n"+frase);
    }
}
```

Tipo Inteiro

Limpando o Buffer

Tipo String