Fundamentos (parte 3)

Prof. Dr. João Paulo Lemos Escola

Tópicos da aula

- Nesta aula vamos falar das estruturas de repetição;
- Vamos conhecer as classes Math e DecimalFormat;
- Também iremos conhecer novos métodos da classe String e aprender a trabalhar com expressões regulares.

Laço de repetição FOR

- Utilizamos o laço do tipo FOR quando é conhecido o favor inicial e final do contador de iterações:

```
for (inicialização; condição; incremento ou decremento){
   // código a ser executado repetidamente
}
```

Exemplo de laço FOR

```
(A4ex1.jar)
public static void main(String[] args){
   for (int i=0; i<=10; i++){
      System.out.println(“valor de i=”+i);
   }
}
```

Laço de repetição While

- Utilizamos o laço do tipo WHILE quando é conhecido o favor inicial e o valor final depende de instruções/condições a serem executadas/validadas:

```
while (condição){
   // código a ser executado repetidamente
}
```

- É possível forçar a execução de, ao menos, uma iteração:

```
do{
   // código a ser executado repetidamente
}while (condição);
```

Exemplo de laço While

```
(A4ex2.jar)
public static void main(String[] args){
   // contador regressivo
   int c = 10;
   while (c >= 0){
      System.out.println(“valor de c=”+c);
      c--;
   }
}
```

Classe Math

- A classe Math faz parte da biblioteca padrão da linguagem Java;
- Para utilizá-la, basta digitar o nome da classe, seguido do nome do método (classe abstrata com métodos estáticos);

Exemplos de métodos da classe Math

- Math.PI
  - Constante que retorna o valor 3,14;
- Math.ceil(valor)
  - Arredondar para cima;
- Math.floor(valor)
  - Arredondar para baixo;
- Math.max(valor1, valor2)
  - Retorna o maior valor;
- Math.min(valor1, valor2);
  - Retorna o menor valor;

Mais métodos da classe Math

- Math.sqrt(valor)
  - Retorna a raiz quadrada;
- Math.pow(base, potencia);
  - Retorna a potência;
- Math.random();
  - Retorna um valor aleatório maior ou igual a 0 e menor que 1;

Exemplo da classe Math

```
(A4ex3.jar)
public static void main(String[] args){
   double pi = Math.PI;
   System.out.println(pi);
   System.out.println(Math.ceil(pi));
   System.out.println(Math.floor(pi));
   // sorteio
   int valor = (int)(Math.random()*10);
   System.out.println(valor);
}
```

Classe DecimalFormat

- Permite criar um formato para apresentação de casas decimais;
- Resultando na limitação do número de casas decimais que será apresentada ao usuário:
- Para criar um padrão, basta utilizar o caractere # para representar um ou mais dígitos, além do “ponto” para representar o valor decimal e o sinal de positivo ou negativo, caso necessário:

  “#.#” → um dígito decimal;

  “#.##” → dois dígitos decimais;

  “#.###” → três dígitos decimais;

Exemplo da classe DecimalFormat

```
(A4ex4.jar)
public static void main(String[] args){
     // cria o objeto do tipo DecimalFormat
     DecimalFormat df = new DecimalFormat("#.###");
     // cria o objeto do tipo DecimalFormat com o formato padrão do país do Sistema Operacional
     //DecimalFormat df = (DecimalFormat)NumberFormat.getNumberInstance(Locale.getDefault());
     // calcula a dízima
     double n = (double)1/6;
     // mostra o valor sem formatar
     System.out.println(n);
     // mostra o valor formatado
     System.out.println(df.format(n));
}
```

Classe String

- Como vimos, é possível criar objetos do tipo String, para armazenar textos;
- Existem diversos métodos para manipular variáveis do tipo String (objetos String):

  length(): retorna o tamanho da String;

  charAt(índice): retorna o caractere de acordo com o índice especificado;

  - Exemplo: “IFSP”, I é índice 0, F está no índice 1...

  toUpperCase(): converte para maiúsculo;

  toLowerCase(): converte para minúsculo;

Classe String (cont.)

substring(índice inicial, índice final): retorna uma cópia da String a partir do índice inicial especificado. O índice final é opcional;

trim(): remove espaços em branco no início e fim do texto;

replace(texto a substituir, texto novo): substitui o texto antigo pelo novo;

indexOf(texto a localizar, posição inicial): retona o índice do texto. A posição inicial é opcinal;

equals(): compara valores em strings;

matches(): compara strings (utilizado em expressões regulares).

Exemplo de métodos String

(A4ex5.jar)

```
public static void main(String[] args){
      String frase = "Eu gosto de banana";
     // retorna o tamanho da String
     System.out.println(frase.length());
     // retorna o caractere que está na posição solicitada (ignorando espaços)
     System.out.println(frase.charAt(3));
     // retorna o texto em letras maiúsculas;
     System.out.println(frase.toUpperCase());
     // retorna o texto em letras minúsculas;
     System.out.println(frase.toLowerCase());
     // retorna um texto parcial, a partir do índice informato até o índice final (excluindo o índice final)
     System.out.println(frase.substring(3, 8));
     // adiciona espaços antes e depois do texto;
     String novaFrase = "  "+frase+"  ";
     System.out.println(novaFrase);
     // retorna o texto sem espaços no início ou fim
     System.out.println(novaFrase.trim());
     // substitui o texto pelo novo
     System.out.println(frase.replace("banana", "abacaxi"));
     // mostra o índice da letra ou texto solicitado
     System.out.println(frase.indexOf("b"));
}
```

Expressões Regulares

- Utilizadas para formatar/filtrar Strings;
- Empregam os metacaracteres para filtrar seu conteúdo:

Caractere	Descrição	Metacaractere
.	Busca qualquer caractere	
\d	Busca qualquer número	[0-9]
\D	Busca qualquer caractere que não seja número	[^0-9]
\w	Busca qualquer caractere de letras e números	[a-zA-Z_0-9]
\W	Busca qualquer caractere que não sejam letras e números	[^\w]
\s	Busca qualquer caractere de espaço em branco, tabulações	[\t\n\x0B\f\r]
\S	Busca qualquer caractere sem espaço em branco	[^\s]

A04ex06

```java
// validação de 1 caracter
System.out.println("1".matches(".")); // true
System.out.println("a".matches(".")); // true
System.out.println("aa".matches(".")); // false
// validação de 2 caracteres
System.out.println("ab".matches("..")); // true
System.out.println("///".matches("..")); // false
// um dígito numérico
System.out.println("1".matches("\\d")); // true
System.out.println("a1".matches("\\d")); // false
System.out.println("-9".matches("\\d")); // false
// dois dígitos numéricos
System.out.println("10".matches("\\d\\d")); // true
System.out.println("100".matches("\\d\\d")); // false
// um caractere e um dígito numérico
System.out.println("a1".matches("\\w\\d")); // true
System.out.println("aa1".matches("\\w\\d")); // false
// dois caracteres e um dígito numérico
System.out.println("aa1".matches("\\w\\w\\d")); // true
System.out.println("a11".matches("\\w\\w\\d")); // true
System.out.println("111".matches("\\w\\w\\d")); // true
System.out.println("11a".matches("\\w\\w\\d")); // false
// um caractere especial
System.out.println("@".matches("\\W")); // true
System.out.println(".".matches("\\W")); // true
System.out.println("/".matches("\\W")); // true
System.out.println("%/".matches("\\W")); // false
// espaço em branco ou tabulação
System.out.println(" ".matches("\\s")); // true
System.out.println("  ".matches("\\s")); // false
```

Quantificadores

- Permitem declarar a quantidade de dígitos ou caracteres em uma expressão regular:

Expressão	Descrição
X{n}	X procura a ocorrência de um caractere n vezes
X{n,}	X pelo menos n vezes
X{n,m}	X pelo menos n mas não mais que m
X?	0 ou 1 vez
X*	0 ou mais vezes
X+	1 ou mais vezes
X{n}	X procura a ocorrência de um caractere n vezes

A04ex07

```java
// dois dígitos numéricos
System.out.println("12".matches("\\d{2}")); // true
System.out.println("12".matches("\\d{3}")); // false
// dois digitos ou mais
System.out.println("12".matches("\\d{2,}")); // true
System.out.println("123".matches("\\d{2,}")); // true
System.out.println("1".matches("\\d{2,}")); // false
// limitando de dois a três dígitos
System.out.println("12".matches("\\d{2,3}")); // true
System.out.println("123".matches("\\d{2,3}")); // true
System.out.println("1".matches("\\d{2,3}")); // false
System.out.println("1234".matches("\\d{2,3}")); // false
// validação de cep
System.out.println("14781-000".matches("\\d{5}-\\d{3}")); // true
System.out.println("14781000".matches("\\d{5}-\\d{3}")); // false
System.out.println("14781000".matches("\\d{5}-\\d{3}")); // false
System.out.println("14.781-000".matches("\\d{5}-\\d{3}")); // false
// validação de data
System.out.println("1/1/2000".matches("\\d{1,}/\\d{1,}/\\d{4}")); // true
System.out.println("01/01/2000".matches("\\d{1,}/\\d{1,}/\\d{4}")); // true
System.out.println("27/02/1889".matches("\\d{1,}/\\d{1,}/\\d{4}")); // true
System.out.println("27/02/89".matches("\\d{1,}/\\d{1,}/\\d{4}")); // false
```

Fronteiras

- Permitem definir onde se inicia ou termina uma String:

Metacaractere	Objetivo
* ^	Inicia
* $	Finaliza
* |	Ou (condição)

A04ex08

```java
// prontuário começa com 18
System.out.println("185566".matches("^18.*")); // true
System.out.println("18".matches("^18.*")); // true
System.out.println("181".matches("^18.*")); // true
System.out.println("155566".matches("^18.*")); // false
// termina com 2018
System.out.println("01/01/2018".matches(".*2018$")); // true
System.out.println("2018".matches(".*2018$")); // true
System.out.println("aaaa2018".matches(".*2018$")); // true
System.out.println("aaaa2018a".matches(".*2018$")); // false
// procura 2018 no texto
System.out.println("aaaa2018a".matches(".*2018.*")); // true
System.out.println("Eu nasci em 2018".matches(".*2018.*")); // true
System.out.println("Em 2018 farei 10 anos".matches(".*2018.*")); // true
System.out.println("Em 20 18 farei 10 anos".matches(".*2018.*")); // false
// pesquisa uma lista de palavras
System.out.println("sim".matches("sim|não")); // true
System.out.println("não".matches("sim|não")); // true
System.out.println("nao".matches("sim|não")); // false
System.out.println("talvez".matches("sim|não")); // false
```

Agrupadores

- Permitem expressões regulares avançadas, agrupando conjuntos e/ou intervalos de caracteres:

[...]	Agrupamento
[a-z]	alcance
[a-e][o-u]	união
[a-z&&[123]]	interseção
[^abc]	exceção
[a-z&&[^m-p]]	subtração

A04ex09

```java
// agrupadores
// uma letra entre a e z
System.out.println("a".matches("[a-z]")); // true
System.out.println("x".matches("[a-z]")); // true
System.out.println("A".matches("[a-z]")); // false
// idem anterior, permitindo maiúsculas
System.out.println("a".matches("[a-z]|[A-Z]")); // true
System.out.println("A".matches("[a-z]|[A-Z]")); // true
System.out.println("x".matches("[a-z]|[A-Z]")); // true
System.out.println("1".matches("[a-z]|[A-Z]")); // false
// não pode começar com número
System.out.println("João".matches("[^0-9].*")); // true
System.out.println("jp2018".matches("[^0-9].*")); // true
System.out.println("2018".matches("[^0-9].*")); // false
// não pode começar com letra
System.out.println("2018".matches("[^a-z][^A-Z].*")); // true
System.out.println("2018jp".matches("[^a-z][^A-Z].*")); // true
System.out.println("jp2018".matches("[^a-z][^A-Z].*")); // false
// validação de email
System.out.println("jpescola@ifsp.edu.br".matches(".*@\\w{2,}[.]\\w{2,}.*")); // true
System.out.println("jpescola@gmail.com".matches(".*@\\w{2,}[.]\\w{2,}.*")); // true
System.out.println("jpescola@gmail".matches(".*@\\w{2,}[.]\\w{2,}.*")); // false
System.out.println("jpescola@ifsp".matches(".*@\\w{2,}[.]\\w{2,}.*")); // false
```

A04ex10

- Crie um jogo da forca utilizando expressões regulares;
- Solicite uma palavra para começar o jogo;
- O usuário tem 7 chances, onde deve especificar uma letra e, a cada letra, o sistema deve completar a palavra ou diminuir uma vida;
- Se o usuário acertar a palavra, ou as vidas acabarem, o jogo termina.

O que aprendemos?

- Estruturas de repetição;
- Classes Math e DecimalFormat;
- Métodos da classe String e expressões regulares.

Na próxima aula...

- Vetores.