Aula 18 - Métodos da Classe String

Os métodos da classe String, nos oferecem várias possibilidades para manipulação de Strings utilizando funções do próprio Java. É muito comum utilizarmos esses métodos durante os nossos desenvolvimentos.

Vamos estudar os principais métodos, são eles:

charAt
Retorna apenas um caractere em determinada posição da String.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        String valor = "Joseffe - Java";
        System.out.println(valor.charAt(0));
    }
}
```

codePointAt
Retorna o valor em UNICODE do caractere especificado no index do parâmetro.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        String valor = "Amazon AWS";
        System.out.println(valor.codePointAt(0));
    }
}
```

compareTo e compareToIgnoreCase
Ambos fazem comparação de duas Strings, sendo que o primeiro (compareTo) considera letras maiúsculas e minúsculas na comparação e o segundo (compareToIgnoreCase) ignora qualquer diferença de minúsculas ou maiúsculas. Ambos também retornam a quantidade de diferenças, sendo que nos importa saber, no momento, que quando o retorno for igual a zero significa que não há diferenças entre a String.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
    String valor = "FIAP - Joseffe";
```

```
        System.out.println(valor.compareTo("FIAP - Joseffe") == 0 ? true : false);
        System.out.println(valor.compareTo("FIAP - JOSEFFE") == 0 ? true : false);
        System.out.println(valor.compareToIgnoreCase("FIAP - JOSEFFE") == 0 ? true : false);
        }
}
```

endsWith e startsWith

O método endsWith verifica se a String termina com o valor especificado, por outro lado o startsWith verifica se a String começa com o valor especificado. Sendo que o método startsWith tem duas variações: uma com o parâmetro “int toffset” e outra sem, onde o método que contém o parâmetro “int toffset” serve para dizer de onde deve começar a verificação do inicio da String.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        String valor = "FIAP - Joseffe";

        System.out.println(valor.endsWith("Joseffe"));
        System.out.println(valor.startsWith("FI"));
        System.out.println(valor.startsWith("Joseffe"));
        System.out.println(valor.startsWith("AP", 2));
    }
}
```

toCharArray

Converte uma String em um Array de char, ou seja, uma String de 10 posições irá ser convertida em um vetor char[] de 10 posições.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        String valor = "FIAP - Joseffe";


        for(char c : valor.toCharArray()){
                System.out.println("Caractere: " + c);
        }
    }
}
```

getBytes
Converte a String em um vetor de byte[]. Este método é muito útil principalmente quando precisamos salvar caracteres no banco desconsiderando a codificação atual. No PostgreSQL, por exemplo, você pode utilizar o tipo “bytea” que é análogo ao “byte” em Java. Assim como você pode converter para byte, você também pode voltar para String ou char.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        String valor = "FIAP - Joseffe";

        for(byte b : valor.getBytes()){
            System.out.println("byte: " + b);
        }
    }
}
```

isEmpty
Um método muito utilizado e comum que verifica se uma String está ou não vazia, mas tenha atenção neste método, pois a verificação baseia-se se sua String possui tamanho = 0, ou seja, o String.length() = 0. Isso significa que caso você tente utilizar o isEmpty() em uma String que possui valor NULL você receberá um NullPointerException.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        //String valor = null;
        String valor = "";

        System.out.println(valor.isEmpty());
    }
}
```

split
O método split cria um array de Strings com base no “regex” passado via parâmetro, ou seja, ele divide a String em várias outras Strings com base no seu regex, o que é muito útil para separar tags em uma String completa, ex: “software, engenharia, computação”. Você pode transformar essa única String em um array com três Strings.

Este método também possui uma variação, que é o parâmetro “int limit”, onde você identifica quantas vezes o regex será aplicado em toda String.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {

        // === Uso do split SEM o LIMIT ===
        String valor = "FIAP - Joseffe";
        String[] valorComSplit = valor.split("-");

        for(String s : valorComSplit){
               System.out.println(s);
        }

        // === Uso do split COM o LIMIT ===
        String valor2 = "FIAP - Joseffe - Java";
        String[] valorComSplit2 = valor2.split("-", 2);

        for(String s : valorComSplit2){
               System.out.println(s);
        }
    }
}
```

substring e subSequence
Retorna uma parte específica de uma determinada String.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {

        String valor = "FIAP - Joseffe";

        System.out.println(valor.substring(0, 5));
    }
}
```

toLowerCase, toUpperCase e trim

O método toLowerCase converte toda a String para caixa baixa e o toUpperCase faz o inverso, convertendo toda a String para caixa alta. O método trim remove espaços em branco no inicial e no final da String.

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        String valor = "  FIAP - Joseffe";

        System.out.println(valor.toLowerCase());
        System.out.println(valor.toUpperCase());
        System.out.println(valor.trim());
    }
}
```

valueOf

Este método converte diversos tipos (booleano, inteiro, char, double, float, long, Object e etc) para String

Exemplo

```
public class Funcoes {

    public static void main(String[] args) {
        //boolean
        boolean myBoolean = true;
        System.out.println(String.valueOf(myBoolean));

        //float
        float myFloat = -10;
        System.out.println(String.valueOf(myFloat));

        //int
        int myInt = 9;
        System.out.println(String.valueOf(myInt));

        ///double
        double myDouble = 10.30;
        System.out.println(String.valueOf(myDouble));
    }
}
```

format
Como o próprio nome já sugere, o String.format realiza a formatação de uma String de acordo com as especificações passadas.

Exemplo

```
public class Funcoes {


    public static void main(String[] args) {
      String result = String.format("Até hoje o salgadelícia já vendeu %,d salgados", 100000);
      System.out.println(result);

      result = String.format("10 / 3 = %.2f", 10.0 / 3.0);
      System.out.println(result);
    }
}
```

Exercícios

Crie um programa que tenha uma variável String chamada **alunosVestibular**. Essa **variável** deve ter o conteúdo abaixo:

String alunosVestibular = Jose dos Santos,7,Sao Paulo**;**Sandra Silva,6.5,Sao Jose do Rio Preto**;**Augusto Soares,8,Sao Paulo**;**Vanderlei Azevedo,5.65,Santos**;**Vanessa Ferreira,9,Sao Paulo**;**Natan Cruz,10,Sao Paulo.

Essa variável contém a lista das pessoas que fizeram o vestibular extra que a faculdade ofereceu. Perceba que a variável possui 6 alunos, todos eles separados por ; (ponto e vírgula).

Nesse conteúdo temos o nome do candidato, nota da prova e a sua cidade de origem.

Com isso, criar um programa onde irá ler esse conteúdo, criar um objeto para cada linha do conteúdo e incluir todos em um HashMap. Apenas incluir no HashMap, os candidatos que tiraram nota igual ou superior a 7.

Ao final, exibir todos os candidatos incluídos no HashMap, nesse formato:

Nome:
Nota:
Cidade:

Nome:
Nota:
Cidade:

```
package ExrAlunosVestibular;

import java.util.HashMap;
```

```
public class programa {
   public static void main(String[] args) {
       int id = 0;
       HashMap<Integer, Aluno> alunosAprovados = new
HashMap<Integer, Aluno>();

       String alunosVestibular = "Jose dos Santos,7,Sao
Paulo;Sandra Silva,6.5,Sao Jose do Rio Preto;Augusto Soares,8,Sao
Paulo;Vanderlei Azevedo,5.65,Santos;Vanessa Ferreira,9,Sao
Paulo;Natan Cruz,6.9,Sao Paulo";

       String[] alunos = alunosVestibular.split(";");

       for (String aluno : alunos) {
           String[] infoAlunos = aluno.split(",");

           if (Double.parseDouble(infoAlunos[1]) >= 7){
               id++;

               Aluno a = new Aluno();
               a.setNome(infoAlunos[0]);
               a.setNota(Double.parseDouble(infoAlunos[1]));
               a.setCidade(infoAlunos[2]);

               alunosAprovados.put(id, a);
           }
       }

       System.out.println("Lista dos alunos aprovados");

       alunosAprovados.forEach((key, value) -> {
           System.out.println("Id: " + key);
           System.out.println("Nome: " + value.getNome());
           System.out.println("Nota: " + value.getNota());
           System.out.println("Cidade: " + value.getCidade());
       });

   }
}
```

Crie um programa que tenha uma variável String chamada **baseDados**. Essa **variável** deve ter o conteúdo abaixo:

String baseDados = CJose dos Santos,42,Sao Paulo**;**CSandra Silva,36,Sao Jose do Rio Preto**;**CAugusto Soares,22,Sao Paulo**;**CVanderlei Azevedo,45,Santos**;**CVanessa

Ferreira,27,Sao Paulo**;**PMouse,1,9.90**;**PTeclado,3,19.90**;**PMonitor,2,349.90**;**PHD SSD,2,199.90**;**PProcessador,1,350.00

Essa variável está reunindo informações de clientes (nome, idade e cidade) e produtos (nome, quantidade em estoque e preço). Perceba que os clientes começam com "C" e os produtos começam com "P". Além disso, observe também que todos eles são separados por ; (ponto e vírgula).

Criar uma classe Cliente e Produto com atributos baseados no que temos na variável String: clientes (nome, idade e cidade) e produtos (nome, quantidade em estoque e preço).

Com isso, criar um programa Java onde irá ler esse conteúdo, criar um objeto para cada registro e incluir em um ArrayList em sua respectiva classe.

Ao final, exibir todos as informações nesse formato:

Clientes:

Nome:
Idade:
Cidade:

Produtos:

Nome:
Qtd em estoque:
Preço: