String

- Strings são sequências de caracteres, que podem conter letras, dígitos e caracteres especiais.
- No Java, uma string é um objeto da classe String.
- String literais são sequências de caracteres no código escritas entre aspas duplas. Ex.: "Amanda Costa".
- São armazenadas na memória como objetos da classe String
- No Java, strings são imutáveis: o seu valor não pode ser mudado depois que elas são criadas.
- A fim de conservar memória, o Java trata todas as string literais que possuem o mesmo conteúdo como um único objeto String.

Classe String – Construtores

A classe String provê diversos construtores, alguns dos quais estão listados abaixo.

Construtor	Descrição
`String`()	Instancia um objeto String que representa uma sequência de caracteres vazia.
`String`(String s)	Instancia um objeto String que é uma cópia da string do argumento.
`String`(char[] v)	Aloca uma nova String que representa a sequência de caracteres no array.
`String`(char[] v, int offset, int count)	Aloca uma nova String que contém caracteres de um subvetor do array de caracteres passado como argumento. `offset` é a posição inicial e `count` é o número de caracteres a ser copiado a partir dessa posição.

Classe String – Métodos

Método	Descrição
int `length`()	Retorna o número de caracteres da string.
char `charAt`(int index)	Retorna o caractere no índice especificado.
void `getChars`(int begin, int end, char[] dest, int destBegin)	Copia caracteres desta string para o array de caracteres dest passado por parâmetro. Os caracteres são copiados a partir da posição begin até a posição end-1. O último argumento especifica a posição inicial onde os caracteres copiados devem ser colocados no array dest

Comparando Strings

Instâncias da classe String não podem ser comparadas com o operador de igualdade ==

Método	Descrição
boolean equals(Object obj)	Compara esta string com o objeto especificado. O resultado é verdadeiro se e somente se o argumento não for null e for um objeto String que representa a mesma sequência de caracteres que este objeto.
boolean equalsIgnoreCase(String str)	Compara esta string com str, ignorando o case.
int compareTo(String str)	Compara duas strings lexicograficamente. Retorna um inteiro negativo se esta String for menor que o argumento str. Retorna um inteiro positivo se esta String for maior que o argumento. Retorna 0 se as strings forem iguais.
boolean regionMatches(int toffset, String other, int ooffset, int len)	Determina se uma substring desta String é igual a uma substring da String other do argumento. A substring desse objeto String a ser comparada começa no índice toffset e tem comprimento len. A substring de other a ser comparada começa no índice ooffset e tem comprimento len.

Substrings

Método	Descrição
int indexOf(int ch)	Retorna o índice da primeira ocorrência do caractere ch nesta string. Retorna -1 se ch não for encontrado
int indexOf(String str)	Retorna o índice da primeira ocorrência da string str dentro desta string. Retorna -1 se str não for encontrado.
int indexOf(String str, int index)	Retorna o índice da primeira ocorrência da string str dentro desta string, começando a partir do índice index. Retorna -1 se str não for encontrado.
int lastIndexOf(int ch)	Retorna o índice da última ocorrência do caractere ch nesta string. Retorna -1 se ch não for encontrado.
int lastIndexOf(String str)	Retorna o índice da última ocorrência da string str dentro desta string. Retorna -1 se str não for encontrado.
int lastIndexOf(String str, int i)	Retorna o índice da última ocorrência da string str dentro desta string, começando de trás para frente a partir do índice i. Retorna -1 se str não for encontrado.

Métodos Diversos

Método	Descrição
String concat(String str)	Retorna a concatenação desta string com a string str.
String toUpperCase()	Retorna uma cópia desta string com todos os caracteres em maiúsculo.
String toLowerCase()	Retorna uma cópia desta string com todos os caracteres em minúsculo.
String replace(char oldChar, char newChar)	Retorna uma string resultado da troca de todas as ocorrências de oldChar nesta string por newChar.
String trim()	Retorna uma string cujo valor é esta string, com todos os espaços iniciais e finais removidos.
char[] toCharArray()	Retorna um novo array de caracteres contendo os mesmos caracteres da string.

Tokenização de strings

- Tokens são palavras individuais que transmitem um significado dentro de um texto.
- Os tokens são separados entre si por delimitadores, em geral caracteres de espaçamento como espaço, tabulação e nova linha. Porém, outros caracteres também podem ser utilizados como delimitadores para separar tokens.
- Em Java, uma string pode ser dividida em tokens usando o método split da classe String

```java
import java.util.Scanner;

public class TokenTest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a sentence and press Enter: ");
        String sentence = scanner.nextLine();

        // process usar sentence
        String[] tokens = sentence.split(" ");
        System.out.printf("Number of elements: %d%nThe tokens are:%n", tokens.length);

        for(String token : tokens) {
            System.out.println(token);
        }
    }
```

```
}
```

A classe empacotadora Character

- A maioria dos métodos da classe Character são métodos estáticos projetados para processar valores char individuais.
- Esses métodos aceitam pelo menos um argumento caractere e realizam um teste ou uma manipulação do caractere.
- Consulte a API

Métodos estáticos da classe Character

Método	Descrição
boolean isDefined(char ch)	Determina se ch é um caractere Unicode.
boolean isJavaIdentifierStart(char ch)	Determina se ch é um caractere que pode ser o primeiro caractere de um identificador no Java.
boolean isUpperCase(char ch)	Determina se ch é caractere maiúsculo.
boolean isLowerCase(char ch)	Determina se ch é caractere minúsculo.
boolean isLetter(char ch)	Determina se ch é uma letra
boolean isDigit(char ch)	Determina se ch é um dígito (0..9)
boolean isLetterOrDigit(char ch)	Determina se ch é letra ou dígito.
boolean isWhitespace(char ch)	Determina se ch é um espaço em branco
char toUpperCase(char ch)	Retorna uma cópia de ch em maiúsculo.
char toLowerCase(char ch)	Retorna uma cópia de ch em minúsculo.
String toString(char ch)	Converte ch para String.

Classe StringBuilder

- StringBuilder é uma string modificável. Toda StringBuilder possui uma capacidade. Se sua capacidade for excedida, ela é expandida a fim de acomodar os caracteres adicionais.
- Se um programa executa muitas operações de concatenação, ou outras modificações de strings, pode ser mais eficiente implementar essas modificações com a classe StringBuilder.
- Documentação StringBuilder

Comparação temp

```java
public class StringBuilderTeste {
    public static void main(String[] args) {
        String s = "";
        int size = 100000;
        long startTime, elapsedTime;

        startTime = System.nanoTime();
        //startTime = System.currentTimeMillis();
```

```
        for(int i = 0; i < size; i++){
            s += i;
        }
        elapsedTime = System.nanoTime() - startTime;
        //elapsedTime = System.currentTimeMillis() - startTime;
        System.out.println("Total execution time  in millis: "
                + elapsedTime/1000000);
        //System.out.println("Total execution time  in millis: "
        //+ elapsedTime);
        StringBuilder s2 = new StringBuilder("");
        startTime = System.nanoTime();
        for(int i = 0; i < size; i++){
            s2.append(i);
        }
        elapsedTime = System.nanoTime() - startTime;

        System.out.println("Total execution time  in millis: "
                + elapsedTime/1000000);


        //System.out.println(s);


    }
}

/*
Output:
Total execution time  in millis: 4109
Total execution time  in millis: 3
*/
```

StringBuilder — Construtores

- StringBuilder fornece alguns construtores, três deles são exibidos abaixo.

Método	Descrição
StringBuilder()	Constrói uma StringBuilder vazia com capacidade inicial para 16 caracteres.
StringBuilder(int cap)	Constrói uma StringBuilder vazia com capacidade inicial igual a cap.
StringBuilder(String str)	Constrói uma StringBuilder e a inicializa com o valor de str.

StringBuilder — Métodos

Método	Descrição
int length()	Retorna o número de caracteres atualmente na StringBuilder
int capacity()	Retorna o número total de caracteres que pode ser armazenado.
void ensureCapacity(int min)	Garante que a capacidade seja pelo menos igual ao mínimo min especificado. A nova capacidade é o maior valor entre min e duas vezes a capacidade anterior mais 2.
void setLength(int n)	A sequência é alterada para uma nova sequência de caracteres cujo comprimento é especificado pelo argumento $n \geq 0$. Se n for maior que a string atual, o valor excedente é preenchido com null '\u0000'

StringBuilder — Método append

- A classe StringBuilder fornece versões sobrecarregadas do método append, que recebe um único valor como argumento e anexa a string representante deste valor à StringBulder atual. Este método retorna a StringBuilder resultante.
- Versões do método append são fornecidas para os tipos nativos, arrays de caracteres, Strings e Objects.

StringBuilder — Método insert

- StringBuilder fornece versões sobrecarregadas do método insert.
- Este método recebe dois argumentos: o primeiro é o índice em que o valor deve ser inserido e o segundo argumento é o valor a ser inserido.
  - O índice deve ser maior ou igual a 0 e menor ou igual ao comprimento da sequência.

StringBuilder — Removendo

- StringBuilder fornece os métodos delete e deleteCharAt para deletar caracteres em qualquer posição de uma StringBuilder.
- StringBuilder delete(int start, int end)
  - A substring a ser excluída começa na posição start e termina na posição end-1 ou vai até o fim desta sequência se essa posição não existir. Se start == end, nenhuma modificação é feita.
- StringBuilder deleteCharAt(int index)
  - Remove o char na posição especificada no argumento.

```
public class StringBuilderInsertDelete {
    public static void main(String[] args) {
        Object objectRef = "hello";
        String string = "goodbye";
        char[] charArray = {'a', 'b', 'c', 'd', 'e', 'f'};
        boolean booleanValue = true;
        char characterValue = 'Z';
        int integerValue = 7;
        long longValue = 10000000000L;
        float floatValue = 2.5f;
        double doubleValue = 33.333;

        StringBuilder buffer = new StringBuilder();
        buffer.insert(0, objectRef);
        buffer.insert(0, "  ");
        buffer.insert(0, string);
        buffer.insert(0, "  ");
        buffer.insert(0, charArray);
        buffer.insert(0, "  ");
        buffer.insert(0, booleanValue);
        buffer.insert(0, "  ");
        buffer.insert(0, characterValue);
        buffer.insert(0, "  ");
        buffer.insert(0, integerValue);
        buffer.insert(0, "  ");
        buffer.insert(0, longValue);
        buffer.insert(0, "  ");
        buffer.insert(0, floatValue);
        buffer.insert(0, "  ");
        buffer.insert(0, doubleValue);

        System.out.printf("buffer after inserts:%n%s%n%n", buffer.toString());

        buffer.deleteCharAt(10); // delete 5 in 2.5
        buffer.delete(2, 6); // delete .333 in 33.333

        System.out.printf("buffer after deletes:%n%s%n", buffer.toString());
    }
}
```