# STRINGS METHODS

CSC 111

# Outline

1. Predefined Methods
2. String Predefined Methods
    - 2.1 String Manipulation
    - 2.2 String Extraction
    - 2.3 String Comparison
3. Parsing Numeric Strings

# 1. PREDEFINED METHODS

- A predefined method is a method whose statements are written (defined) by Java. The programmer uses it immediately.
- In fact, we already used predefined methods in previous lectures such as nextInt(), nextDouble(), etc…

REMINDER

```
1  import java.util.*;          //contains the class Scanner
2  static Scanner console = new Scanner (System.in);
3  int input;
4  input = console.nextInt();//nextInt() is a method predefined in Scanner
```

In our course, predefined methods are written from now on in green

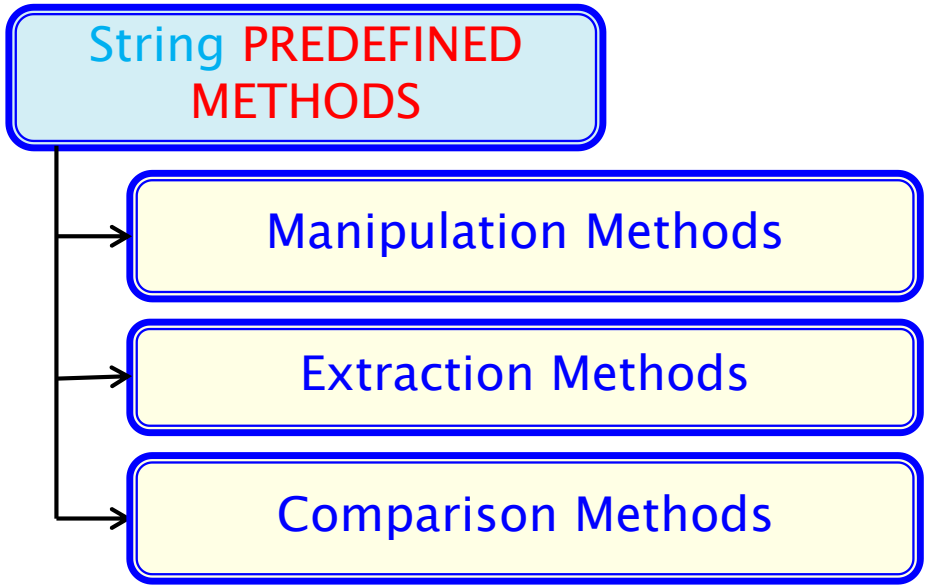
# 2. String PREDEFINED METHODS

- The class String contains many predefined methods.
- The syntax of using a String predefined method is as follows:

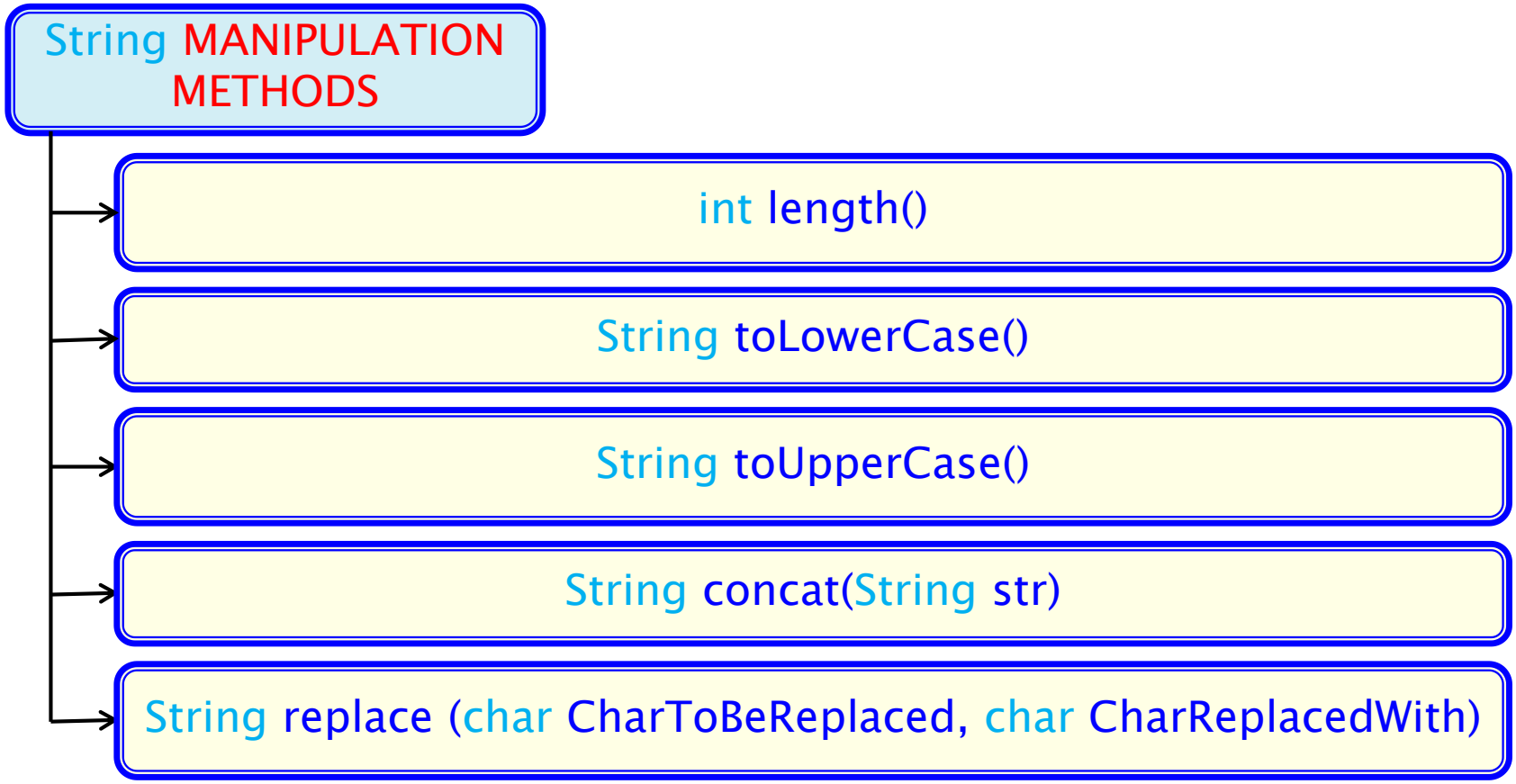
| SYNTAX | stringVariable.stringMethodName (parameters) |
|---|---|

- The stringMethodName is applied on stringVariable.
- The dot (.) is called member access operator.
- Not all methods need parameters. Others may need more than one parameter.
- Parameters have predefined types that we should follow.
- In case the method has more than one parameter, the order is important.
- When a method returns a value of a specific type, then the result should be stored in a variable of the same type.

# 2. String PREDEFINED METHODS



# 2.1 MANIPULATION METHODS



# 2.1.1 MANIPULATION METHODS

## int length()

- This method returns the length of the stringVariable.
- This method has no parameters. Yet, we write the parenthesis.
- It returns an int value. So, the result should be stored in a variable of type int.

```
1  String firstName = "Ghadeer", courseName = "Java I";
2  int len1, len2;                    //used to store the output of the method
3  len1 = firstName.length();     //store the length of firstName in len1
4  len2 = courseName.length();//store the length of courseName in len2
5  System.out.println (len1);
6  System.out.println (len2);
```

```
1  7
2  6
```

- In line 3, the stringVariable is firstName: length() method is applied on the String firstName.
- In line 4, the stringVariable is courseName: length() method is applied on the String courseName.

When the method has no parameters, the parenthesis are still written.

# 2.1.1 MANIPULATION METHODS

## int length()

- Consider the following code:

```
1  String emptyString = "";    //with no space between quotes
2  int len;
3  len = emptyString.length();    //store the length of emptyString in len
4  System.out.println (len);
```

```
1  0
```

- In line 3, the stringVariable is emptyString: length() method is applied on the String emptyString.

Maximum position = length – 1

The length of the empty (null) string is zero.

# 2.1.2 MANIPULATION METHODS

## String toLowerCase()

- This method converts all characters in the stringVariable into lower case letters.
- This method has no parameters. Yet, we write the parenthesis.
- It returns a String value. So, the result should be stored in a variable of type String.

```
1 String institution = "King SAUD University";
2 String newString;
3 newString = institution.toLowerCase();
4 System.out.println (newString);
```

```
1 king saud university
```

- In line 3, the stringVariable is newString: toLowerCase() method is applied on the String institution.
- The result is stored in a variable of type String: newString.

# 2.1.3 MANIPULATION METHODS

## String toUpperCase()

- This method converts all characters in the stringVariable into upper case letters.
- This method has no parameters. Yet, we write the parenthesis.
- It returns a String value. So, the result should be stored in a variable of type String.

```
1 String institution = "King SAUD University";
2 String newString;
3 newString = institution.toUpperCase();
4 System.out.println (newString);
```

```
1 KING SAUD UNIVERSITY
```

- In line 3, the stringVariable is newString: toUpperCase() method is applied on the String institution.
- The result is stored in a variable of type String: newString.

# 2.1.4 MANIPULATION METHODS

## String concat(String str)

- This method concatenates the stringVariable with the parameter str.
- This method has one parameter only of type String.
- This method returns a String value.

```
1 String firstName, familyName, fullName;
2 firstName = "Noha~";                    //~ represents a space character
3 familyName = "Al-Otaibi";
4 fullName = firstName.concat(familyName);//firstName+familyName
5 System.out.println (fullName);
```

```
1 Noha Al-Otaibi
```

- The concat method is applied on the StringVariable: firstName.
- The concat method has one parameter (familyName): this is the string to be joined to the first string (firstName in this example)

Note that we DO NOT write the type of the parameter

# 2.1.5 MANIPULATION METHODS

## String replace (char CharToBeReplaced, char CharReplacedWith)

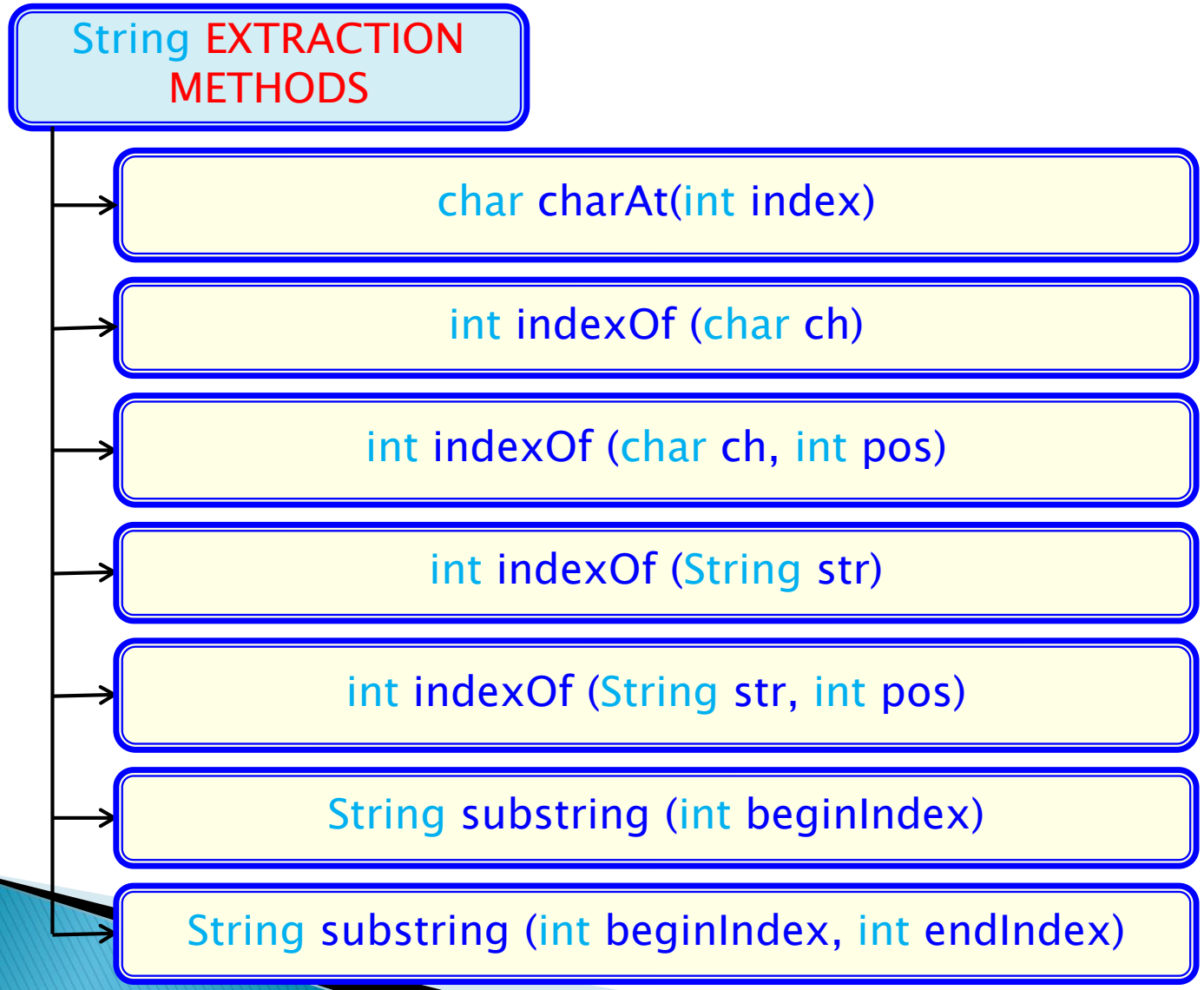
- This method seeks for all occurrences of the first parameter charToBeReplaced in stringVariable and replaces it with the second parameter CharReplaceWith.
- This method returns a String value.

```
1 String institution = "King Saud University";
2 String newString;
3 newString = institution.replace ('i', '*');
4 System.out.println (newString);
```

```
1 K*ng Saud Un*vers*ty
```

| K | i | n | g |  | S | a | u | d |  | U | n | i | v | e | r | s | i | t | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

# 2.2 EXTRACTION METHODS



# 2.2.1 EXTRACTION METHODS

## char charAt(int index)

- This method extracts the character at the position index.
- The method has one parameter of type int: the target position.
- Returns a char value.

```
1  char ch;
2  int position = 1;
3  String institution = "KSU";
4  ch = institution.charAt(position); //extract the character at position=1
5  System.out.println (ch);
6  ch = institution.charAt(position - 1); //extract the character at position=0
7  System.out.println (ch);
8  ch = institution.charAt(institution.length() -1);//extract the last character
9  System.out.println (ch);
```

```
1  S
2  K
3  U
```

| String | "KSU" | | |
|---|---|---|---|
| Character in the String | 'K' | 'S' | 'U' |
| Position of the character | 0 | 1 | 2 |

Parameters may be arithmetic expressions

The value of index <= maximum position of stringVariable (length-1)

# 2.2.2 EXTRACTION METHODS

## int indexOf (char ch)

- This method returns the position of the character ch (the parameter) in the stringVariable.
- If ch is repeated in the stringVariable, the method returns the position of the first occurrence.
- If ch does not exist in the stringVariable, the method returns -1.
- The method returns an int type.

```
1  int ndx1, ndx2;
2  char ch1='n', ch2 = 'b';
3  String institution = "King Saud University";
4  ndx1 = institution.indexOf(ch1); //returns the position of the first 'n'
5  ndx2 = institution.indexOf(ch2); //institution doesn't contain ch2 ('b')
6  System.out.println (ndx1);
7  System.out.println (ndx2);
```

```
1  2
2  -1
```

# 2.2.3 EXTRACTION METHODS

## int indexOf (char ch, int pos)

- This method searches for the character ch in stringVariable. The search starts from the position pos.
- If ch appears more than once after pos in stringVariable, the method returns the position of the first occurrence after pos.
- If ch does not appear after pos, the method returns -1.
- The method returns an int value.

```
1  int ndx1, ndx2, pos = 7;
2  char ch1='i', ch2 = 'K';
3  String institution = "King Saud University";
4  ndx1 = institution.indexOf(ch1, 7);
5  //ndx1= position of the first 'i' starting from position 7
6  System.out.println (ndx1);
7  ndx2 = institution.indexOf(ch2, pos); //there is no 'K' after pos = 7
8  System.out.println (ndx2);
```

```
1  12
2  -1
```

| K | i | n | g |  | S | a | u | d |  | U | n | i | v | e | r | s | i | t | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

The value of index <= maximum position of stringVariable (length-1)

# 2.2.4 EXTRACTION METHODS

## int indexOf (String str)

- If the string str (parameter) exists in the stringVariable, this method returns the position of its first character.
- If str appear more than once, the method returns the position of the first occurrence.
- If str does not appear in the stringVariable, the method returns -1.

```
1  int ndx1, ndx2;
2  String institution = "King Saud University";
3  String str1 = "Saud";
4  String str2 = "Norah";
5  ndx1 = institution.indexOf(str1);
6  //ndx1= position of the first character in str1 found in institution
7  System.out.println (ndx1);
8  ndx2 = institution.indexOf(str2); //there is no str2 in institution
9  System.out.println (ndx2);
```

```
1  5
2  -1
```

| K | i | n | g |  | S | a | u | d |  | U | n | i | v | e | r | s | i | t | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

# 2.2.5 EXTRACTION METHODS

## int indexOf (String str, int pos)

- This method seeks for the string str (the first parameter) starting from the position pos (the second parameter).
- If str appear more than once, the method returns the position of the first character of str in the first occurrence.
- If str does not appear in the stringVariable after the position pos, the method returns -1.

```
1  int ndx1, ndx2, pos = 7;
2  String str1="University", str2 = "King";
3  String institution = "King Saud University";
4  ndx1 = institution.indexOf(str1, pos);
5  //ndx1= position of the first character of str1 starting from position 7
6  System.out.println (ndx1);
7  ndx2 = institution.indexOf(str2, pos); //str2 doesn't exist after pos = 7
8  System.out.println (ndx2);
```

```
1  10
2  -1
```

| K | i | n | g |  | S | a | u | d |  | U | n | i | v | e | r | s | i | t | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

# 2.2.6 EXTRACTION METHODS

## String substring (int beginIndex)

- This method extracts a substring from stringVariable starting from the position beginIndex (parameter) till the end of stringVariable.

```
1 String institution = "King Saud University";
2 String newString;
3 newString = institution.substring (10);
4 //extract the substring starting from position 10 till the end
5 System.out.println (newString);
6 newString = institution.substring(institution.indexOf("Saud"));
7 // = institution.substring(5) = Saud University
8 System.out.println (newString);
```

```
1 University
2 Saud University
```

| K | i | n | g | | S | a | u | d | | U | n | i | v | e | r | s | i | t | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

0 <= beginIndex <= stringVariable (length-1)

# 2.2.7 EXTRACTION METHODS

## String substring (int beginIndex, int endIndex)

- This method extracts a substring from stringVariable starting from the position beginIndex (the first parameter) till endIndex-1 (the second parameter).

```
1 String institution = "King Saud University";
2 String newString;
3 newString = institution.substring (10, 14);
4 //extract the substring starting from position 10 till position 13
5 System.out.println (newString);
```

```
1 Univ
```

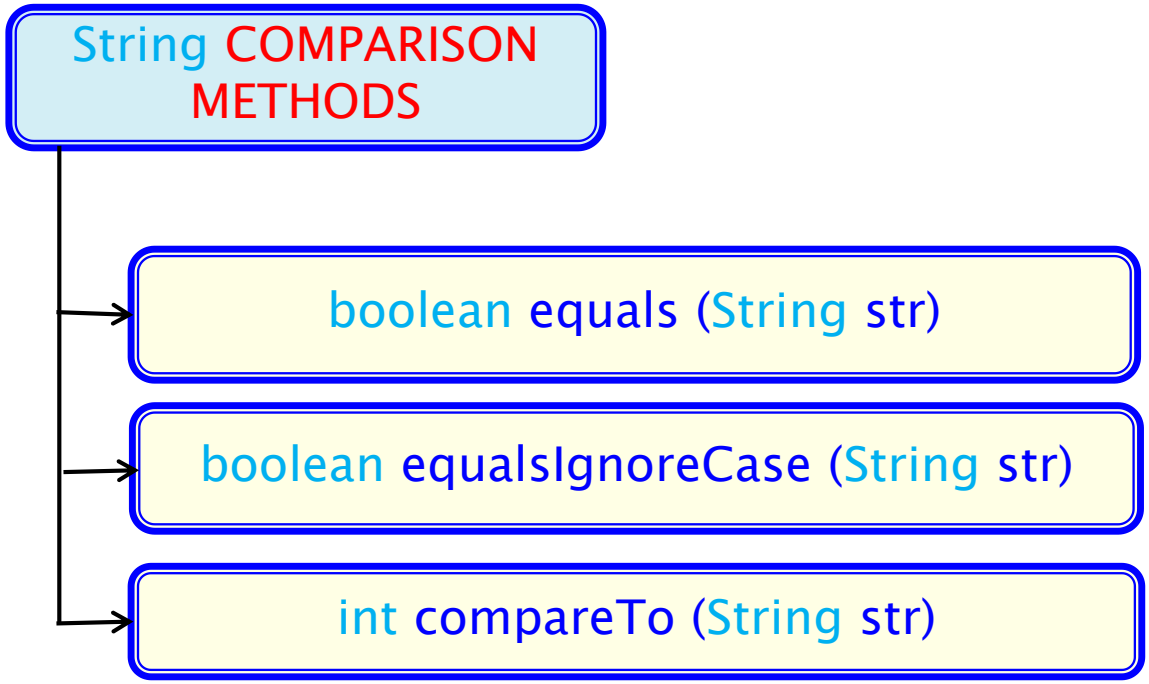
| K | i | n | g |  | S | a | u | d |  | U | n | i | v | e | r | s | i | t | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

0 <= beginIndex

**endIndex** <= stringVariable (length-1)

What if **endIndex == stringVariable (length)** ?
what if **beginIndex < endIndex** ?

# 2.3 COMPARISON METHODS



# 2.3.1 COMPARISON METHODS

## boolean equals (String str)

- This method checks if str is equal to stringVariable.
- The method returns true if str is equal to the stringVariable; otherwise, it returns false.

```
1  boolean same;
2  String str1="King Saud University", str2 = "King";
3  String institution = "King Saud University";
4  same = institution.equals(str1); //institution = str1
5  System.out.println (same);
6  same = institution.equals(str2); //str2 doesn't equal institution
7  System.out.println (same);
```

```
1  true
2  false
```

| K | i | n | g |  | S | a | u | d |  | U | n | i | v | e | r | s | i | t | y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

java accept the following expression for string compasision

same = (str1 == str2)

, but you need to understand that string is reference type and such comparison should be avoided

# 2.3.2 COMPARISON METHODS

## boolean equalsIgnoreCase (String str)

- This method checks if str is equal to stringVariable.
- The method returns true if str is equal to the stringVariable; otherwise, it returns false.

```
1  boolean same;
2  String str1="KSU";
3  String institution = "ksu";
4  same = institution.equalsIgnoreCase (str1); //institution = str1
5  System.out.println (same);
6  same = institution.equals(str1); //str1 doesn't equal institution
7  System.out.println (same);
```

```
1  true
2  false
```

| k | s | u |
|---|---|---|
| 0 | 1 | 2 |

# 2.3.3 COMPARISON METHODS

## int compareTo (String str)

- This method compares str with the stringVariable.
- The method returns a negative value if stringVariable is less than str; i.e. when stringVariable comes before str alphabetically.
- It returns 0 if the stringVariable is equal to str.
- It returns a positive value if stringVariable is greater than str; i.e. when stringVariable comes after str alphabetically.

```
1 String name1 = “Hend”, name2 = “Salma”, name3 = “Maha”;
2 String myName=“Maha”;
3 int out1, out2, out3;
4 out1 = myName.compareTo(name1); //myName > name1
5 System.out.println (out1);
6 out2 = myName.compareTo(name2); //myName < name2
7 System.out.println (out2);
8 out3 = myName.compareTo(name3); //myName = name3
9 System.out.println (out3);
```

```
1 5
2 -6
3 0
```

# 3. PARSING NUMERIC STRINGS

- A string that consists of only an integer or a floating-point number is called a numeric string.

- A numeric string can contain a minus sign.

- Examples of numeric strings include:
  - "2015"
  - "-20"
  - "144.45"
  - "-53.99"

- In order to process numeric strings as numbers (addition, multiplication, etc…), they must be converted first into numbers.

# 3. PARSING NUMERIC STRINGS

- Integer.parseInt (strExpression) converts a numeric string to an int number.
  - Integer.parseInt ("2015") = 2015
  - Integer.parseInt ("-20") = -20

- Float.parseFloat (strExpression) converts a numeric string to a float number.
  - Float.parseFloat ("144.45") = 144.45
  - Float.parseFloat ("-53.99") = -53.99

- Double.parseDouble (strExpression) converts a numeric string to a double number.
  - Double.parseDouble ("144.45") = 144.45
  - Double.parseDouble ("-53.99") = -53.99

# Self-Check Exercises (1)

- What is the output of the following code?

```
1  String message = "I am a CS student";
2  String newString;
3  int pos = 5, len;
4  newString = message.substring (7);
5  System.out.println (newString);
6  newString = message.charAt(pos + 1);
7  System.out.println (newString);
8  len = newString.length();
9  System.out.println (len);
10 pos = newString.indexOf ('t');
```

- Perform the following operations on:

  String myHobbies = "I like programming with Java";

  - Extract the word "Java"
  - Replace all 'g' characters with '?'
  - Compare with the String str = "I like reading"
  - Exchange the word "Java" in myHobbies with "C++"