Methods in JAVA

Computer Engineering

Yusramohammed@tiu.edu.iq

2022 - 2023

Methods in JAVA

- **Method:** A Java method is a collection of statements that are grouped together to perform an operation.

- So far, we used only one method to do all the processes for us. The method was ***main*** method.

- When we run a Java program, the main method is executed first. So to execute a code in Java, there must be a main method.

public static void main(String args[]){ }

- **public** is modifier.
- **static**: you access this function without making object.
- **void**: method will not return anything.
- **main** is the name of method

- Method **must** be declared inside class, it has ***name*** followed by ( ).

public void add(){ }

Methods in JAVA

- Java has some predefined method such as
  - **System.out.println();**
  - **Static Methods (Class Methods)**
  - You can create your own method → Study in the following slides

- **The benefits of method are:**
  - Code reuses
  - Code optimization

Static methods in JAVA

- **Static method:** Method applies to the class in which it's declared as a whole and is known as a **class method.**
- you can call the class's static methods by:

```
ClassName.methodName( arguments )
```

- We use **Math** class's and **String** class's methods as an example.

Math class and its methods

- Math class contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.
- **Math.methodName(argument);**

```
Math.sqrt( 900.0 )
```

- **Example:**

```
public class MathClass
{

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        System.out.println(Math.sqrt(900.0));
    }

}
```

output → 30.0

Math class and its methods

- **Example**:

```
public class Example {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("Enter the numbers: ");
        int num1=input.nextInt();
        int num2=input.nextInt();
                int add=Math.addExact(num1, num2);
                System.out.println("Add--> "+add);
                int sub=Math.subtractExact(num1, num2);
                System.out.println("Sub--> "+sub);

        }
}
```

```
run:
Enter the numbers:
3
2
Add--> 5
Sub--> 1
```

Math class and its methods

- several Math class methods are summarized in the table below

Method	Description	Example
`abs( x )`	absolute value of x	`abs( 23.7 )` is 23.7 `abs( 0.0 )` is 0.0 `abs( -23.7 )` is 23.7
`ceil( x )`	rounds x to the smallest integer not less than x	`ceil( 9.2 )` is 10.0 `ceil( -9.8 )` is -9.0
`cos( x )`	trigonometric cosine of x (x in radians)	`cos( 0.0 )` is 1.0
`exp( x )`	exponential method e^x	`exp( 1.0 )` is 2.71828 `exp( 2.0 )` is 7.38906
`floor( x )`	rounds x to the largest integer not greater than x	`floor( 9.2 )` is 9.0 `floor( -9.8 )` is -10.0
`log( x )`	natural logarithm of x (base e)	`log( Math.E )` is 1.0 `log( Math.E * Math.E )` is 2.0
`max( x, y )`	larger value of x and y	`max( 2.3, 12.7 )` is 12.7 `max( -2.3, -12.7 )` is -2.3
`min( x, y )`	smaller value of x and y	`min( 2.3, 12.7 )` is 2.3 `min( -2.3, -12.7 )` is -12.7
`pow( x, y )`	x raised to the power y (i.e., x^y)	`pow( 2.0, 7.0 )` is 128.0 `pow( 9.0, 0.5 )` is 3.0
`sin( x )`	trigonometric sine of x (x in radians)	`sin( 0.0 )` is 0.0
`sqrt( x )`	square root of x	`sqrt( 900.0 )` is 30.0
`tan( x )`	trigonometric tangent of x (x in radians)	`tan( 0.0 )` is 0.0

Activity using Math Class:

- Create a java class to enter 3 number from the user and determine the max number and its cube using Math class.

```
import java.util.Scanner;
public class ActivityOne {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner input=new Scanner(System.in);
        System.out.print("Enter Number1: ");
        int num1=input.nextInt();
        System.out.print("Enter Number2: ");
        int num2=input.nextInt();
        System.out.print("Enter Number3: ");
        int num3=input.nextInt();

        int maxNum=Math.max(num1,Math.max(num2,num3));
        double cube=Math.pow(maxNum, 3);
        System.out.println("maxNum:"+maxNum+"\ncube: "+cube);
    }
}
```

Characters and Strings in Java

- **String**: A *string* is a sequence of characters that is treated as a single value.
- All string literals such as “abc” are implemented as instances of **String** class. Strings are constant, their values cannot be changed after they are created.
- There are close to 50 methods defined in the String class. We will introduce
- some of them here:

- first lets declare and initialize two strings and work on them using methods in String class.

  - String text1=“HelloWorld”;
  - String text2=“Programmer”

Characters and Strings in Java (Continue)

Method	Description	Example
substring(beginning position, ending position)	extract a **substring** from a given string by specifying the beginning and ending positions.	text1.substring(2,5)
length()	find out the number of characters in a String object	text1.length()
indexOf()	locate the index position of a substring within another string. If there is more than one occurrence of the same substring, the index position of the first character of the first matching substring is returned.	text1.indexOf(Hello)
String Concatination: • "first string"+"secondstring" • firstString.concat(SecondString)	Concatenates the specified string to the end of this string.	text1.concat(text2);
charAt()	Returns the char value at the specified index. An index ranges from 0 to length() - 1.	Text1.charAt(name of index);
toLowerCase()	Converts all of the characters in this String to lower case using the rules of the default locale.	Text1.toLoowerCase()

Characters and Strings in Java (Continue)

Method	Description	Example
toUpperCase()	Converts all of the characters in this String to upper case using the rules of the default locale	Text1.toUpperCase()
equals()	Compares this string to the specified object.	Text1.equales(“specifie d string”)
equalsIgnoreCase()	Compares this String to another String, ignoring case considerations.	Text1.equalsIgnoreCase(“he llo”);
compareTo()	Compares two strings lexicographically. Each character of both the strings is converted into a Unicode value for comparison. If both the strings are equal then this method returns 0 else it returns positive or negative value. The result is positive if the first string is lexicographically greater than the second string else the result would be negative	Text1.compareTo(“hello”) ; Text1.compareTo(text2)
Replace(‘oldCharacter’,’ newCharacter’))	This method returns a new string resulting from replacing all occurrences of oldChar in this string with newChar.	text1.replace(‘oldCharacter ', ‘newCharacter')

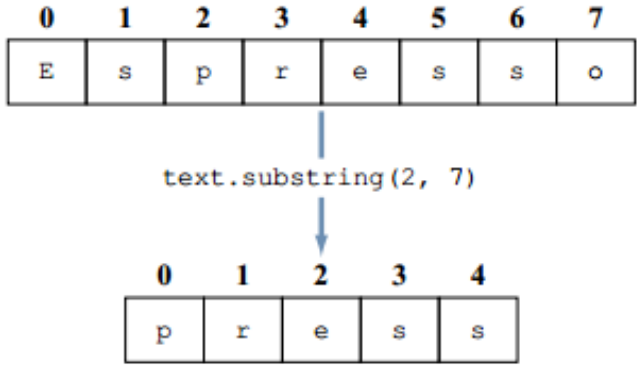
Characters and Strings in Java (Continue)

Method	Description	Example
replaceAll(“oldsubstring ”,”newsubstring”)	Replaces each substring of this string that matches the given regular expression with the given replacement.	text1.replaceAll(“he", “eh")
isEmpty()	This method returns true if, and only if, length() is 0.	text1.isEmpty()
hashCode()	This method returns a hash code for this string. Hash code value is used in hashing based collections like HashMap, HashTable etc	text1.hashCode()
trim()	returns a copy of the string and omits beginning and ending whitespace. Remove whitespace from both sides of a string:	text1.trim()
toString()	represent the result in textual format and returns the string itself. Example: int x=12; String y=int.toString(x);	type.toString(variable)
Split()	The string split() method breaks a given string around matches of the given regular expression. We create an array of type string to store the splitted string	String a=“hello world” String []b=a.split(“ “);

Examples

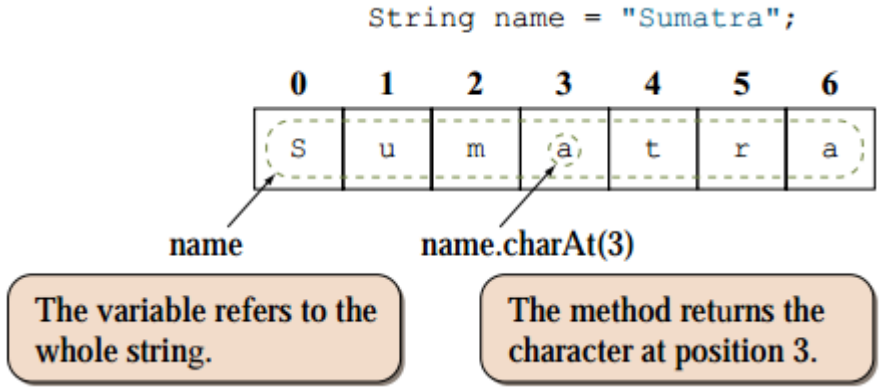
- substring(x,y);

```
text.substring( 6, 8 )  ⟶  "so"
text.substring( 0, 8 )  ⟶  "Espresso"
text.substring( 1, 5 )  ⟶  "spre"
```



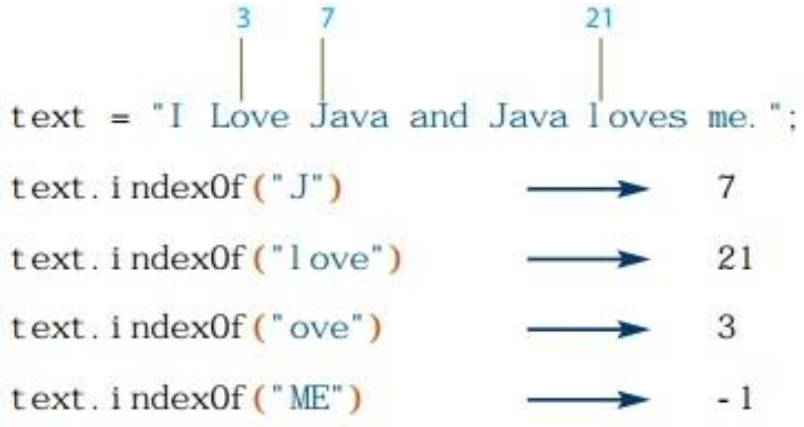
- charAt(index)

```
String name = "Sumatra";
int    size = name.length( );
for (int i = 0; i < size; i++) {
   System.out.println(name.charAt(i));
}
```



Examples

- indexOf()



- length()

```
text1 = "";          //empty string
text2 = "Hello";
text3 = "Java";

text1.length( ) ⟶ 0
text2.length( ) ⟶ 5
text3.length( ) ⟶ 4
```

- string concatenate

```
text1 = "Jon";
text2 = "Java";

text1 + text2                ⟶  "JonJava"
text1 + " " + text2          ⟶  "Jon Java"
"How are you, " + text1 + "?"
                             ⟶  "How are you, Jon?"
```

OR text1.concat(text2);

Example

- trim()

```
String myStr = "        Hello World!        ";
System.out.println(myStr);
System.out.println(myStr.trim());
```

```
        Hello World!
Hello World!
```

- hashCode()

```
String blogName = "howtodoinjava.com";

System.out.println( blogName.hashCode() );
```

```
1894145264
```

- split()

```
  String s="Hello world how are you";
String []a=s.split(" ");
for(String x: a)
{
    System.out.println(x);
}
}
```

output

```
Hello
world
how
are
you
```

Example

- Write a program to ask the user to enter his/her name and count the vowels in the name sequence

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    Scanner scanner = new Scanner(System.in);
    String name;
    int vowelCount = 0;
    char letter;
    System.out.print("What is your name?");
    name = scanner.next( );
    for (int i = 0; i < name.length(); i++) {
    letter = name.charAt(i);
    if (letter == 'a' || letter == 'A' ||
    letter == 'e' || letter == 'E' ||
    letter == 'i' || letter == 'I' ||
    letter == 'o' || letter == 'O' ||
    letter == 'u' || letter == 'U' ) {
    vowelCount++;
    }
    }
    System.out.println(name + ", your name has " +
    vowelCount + " vowels");
    }
}
```

```
What is your name?Yusra
Yusra, your name has 2 vowels
```

Example

- Write an application that uses String method **compareTo()** to compare two strings input by the user. Output whether the first string is less than, equal to or greater than the second.

```
public class CompareTwoStrings {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("Enter two sentences: ");
        String sentence1=input.next();
        String sentence2=input.next();

        int length=sentence1.compareTo(sentence2);
        if(length>0)
        System.out.println(sentence1 +" is greater than "+sentence2);
        else if(length<0)
            System.out.println(sentence2 +" is greater than "+sentence1);
        else
            System.out.println(sentence1 +" and "+sentence2+" are equale");

    }
}
```

Run the code and see the output

Sample Development

- Create a Scientific calculator to ask the user to enter 3 numbers and apply all mathematical operation in the Math class showed in the table (Slide 6). Let the user choose which option to be applied using switch-case

```
public class Example {
    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("Enter the numbers: ");
        int num1=input.nextInt();
        int num2=input.nextInt();
        int num3=input.nextInt();
        System.out.println("Add--> +\nSub--> -\nMax--> M\nSelect Options");
        char option=input.next().charAt(0);
        switch(option)
        {
            case '+':
                int add=Math.addExact(num1, num2);
                System.out.println("Add--> "+add);
            break;
            case '-':
                int sub=Math.subtractExact(num1, num2);
                System.out.println("Sub--> "+sub);
            break;
            case 'M':
                int max=Math.max(num1, Math.max(num2, num3));
                System.out.println("Max is--> "+max);
            break;
        }
```

Try the other option and test the output

Examples

1. Write an application to count the number of non-vowels in a given string using toUpperCase().
2. Write a program to find the shortest and the longest word in a sentence and print them along with their length.
3. Write an application that inputs a line of text and outputs the text twice—once in all uppercase letters and once in all lowercase letters.
4. Extract the words in a given sentence and print them, using one line per word. Don't use split() method
5. Write an application to ask the user to enter username and password, and compare the input values to the specified values by the programmer, if so, print "**welcome**" else print "**username or password is incorrect**". Let username be "**programmer**" and password be "**user12345**" and use equal() method to test the two strings