Zoho Schools for Graduate Studies

Notes

PROGRAMMING BASIC PART -1

1) Why the parameter of the main method is a String [], not float []/ int []?

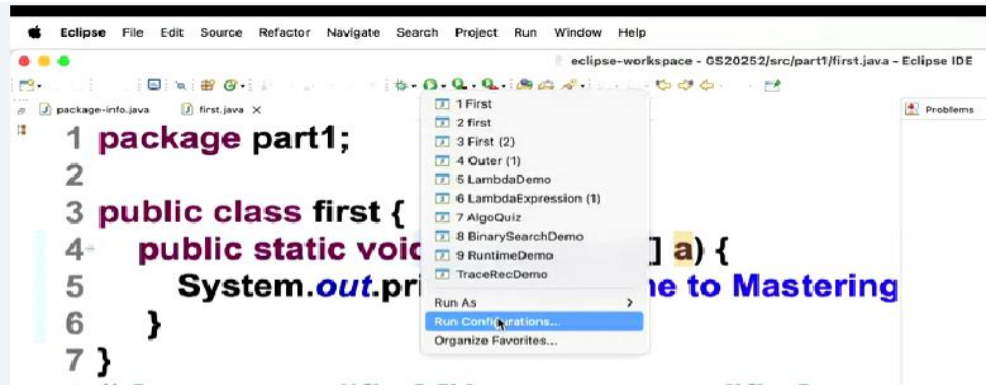
- ⇨ String [] can accommodate any datatype after conversion.
- ⇨ It accepts all 6 numeric datatypes and any object type variables.

2) Why do we use command-line arguments instead of getting inputs from the user?

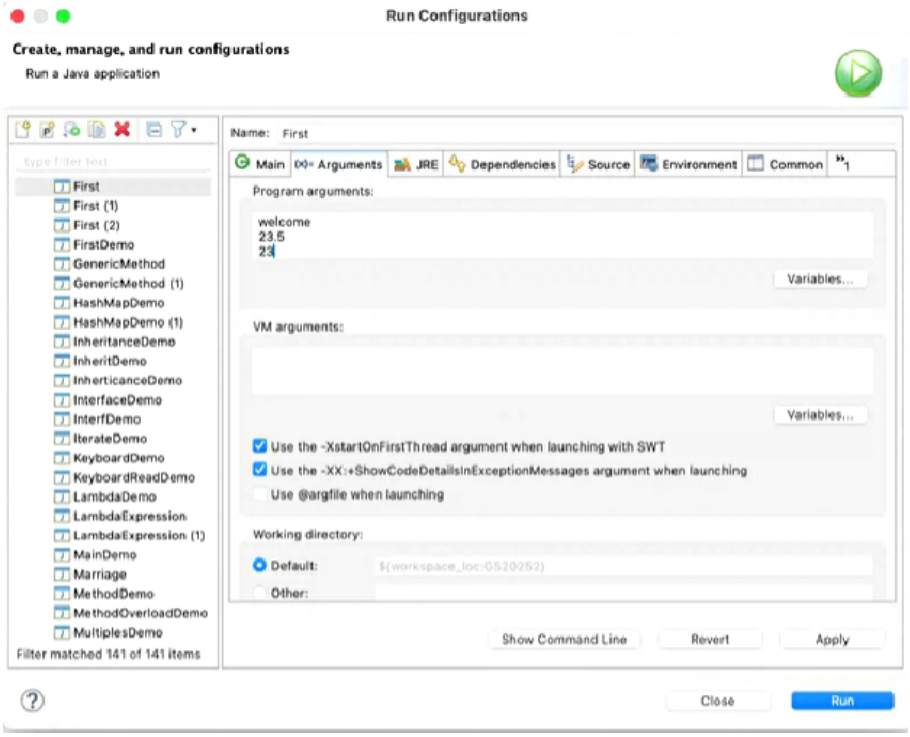
- ⇨ Command-line arguments allow us to supply inputs to the main method without user interaction.
- ⇨ They can also be displayed and used directly in the program.

3) How can we supply the String[] arguments to the main method in Eclipse?

Step1: Open **Run Configurations**.



Step 2: Enter the required arguments under the **Arguments** tab.



⇨ Here space between Strings considered as a separator (new argument).

<u>Example 1:</u>

Welcome

23.5

23

Thamarai Selvam San

⇨ If you want multiple words as a single argument, enclose them in quotes " ".

<u>Example 2:</u>

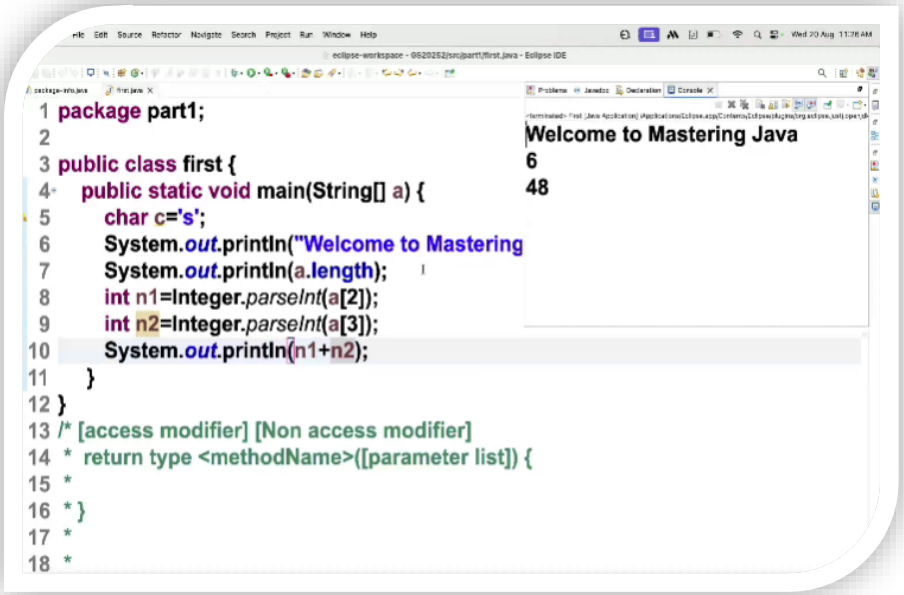
Welcome

23.5

23

"Thamarai Selvam San"

4) How can we use it in our program?



I. Length of the arguments (Example 1):

a.length => output: 6

II. Length of the arguments (Example 2):

a.length => output: 4

III. System.out.println(a[2]+a[2]) => output: 2323 *// here "+" act as concatenation*

IV. System.out.println(a[2]+'s') => output: 138 *// here 's' changed as ASCII value 115*

V. int n1 = Integer.parseInt(a[2]); *// Converts 3rd command-line argument to integer. (Example:23)*

int n2 = Integer.parseInt(a[3]); *// Converts 4th command-line argument to integer. (Example:11)*

System.out.println(n1 + n2); *// Prints the sum of the two integers. => output 34*

5) java.lang => Integer => parseInt

public static int parseInt(String s) throws NumberFormatException

Parses the string argument as a signed decimal integer.

Parameters:

s - a String containing the int representation to be parsed

Returns:

the integer value represented by the argument in decimal.

Throws:

NumberFormatException - if the string does not contain a parsable integer.

6) Plus Operator (+):

"+" Acts as **concatenation** when at least one operand is a **Non-numeric**.

"+" Acts as **addition** when both operands are **numeric**.

"+" binary operator / implicitly overloaded operator

7) Wrapper Classes:

- ⇨ Wrapper classes are the object representation of primitive data types.
- ⇨ Classes: Byte, Short, Integer, Long, Float, Double, Character, Boolean.

8) Naming Convention:

1. Class Names

- Use UpperCamelCase (PascalCase).
- Each word starts with an uppercase letter.
- Example: StudentDetails, BankAccount.
- Using lowercase is legal, but not good practice.

2. Method Names

- Use lowerCamelCase.
- The first word starts with a lowercase letter.
- Each subsequent word starts with an uppercase letter.
- Example: calculateSum(), getStudentName().

3. Variable Names

- Same as method names → lowerCamelCase.
- Example: totalMarks, studentAge.

4. Constants / final variables

- Use uppercase letters with underscores.
- Example: PI, MAX_SPEED.

5. Packages

- Always lowercase.
- Example: java.util, part1.

6. Keywords

- Always lowercase (defined by Java).
- Example: class, public, static.

Naming Conventions:

1. If several words are linked together to form a name for an identifier, then the first letter of the inner words should be capitalized (upper case)

2. Class names and interface names must begin with an uppercase
   a) class names should be typically be nouns
   b) interface names should be typically be adjectives

3. Methods and variables must begin with a lower case
   a) method names should be typically be verb-noun pairs
   b) variable names should be short and meaningful

4. Constants must be fully capitalized with underscore connecting multiple words.

```
// onetwothree -> OneTwoThree
// class OneTwoThree{   }
// interface OneTwoThree{   }
// void oneTwoThree{   }
// int  oneTwoThree=5;
// package oneTwoThree;
```

Question:

How many implicitly overloaded operators are there in Java?