JPL :: Command-Line Arguments

TalentSprint

Licensed To Skill

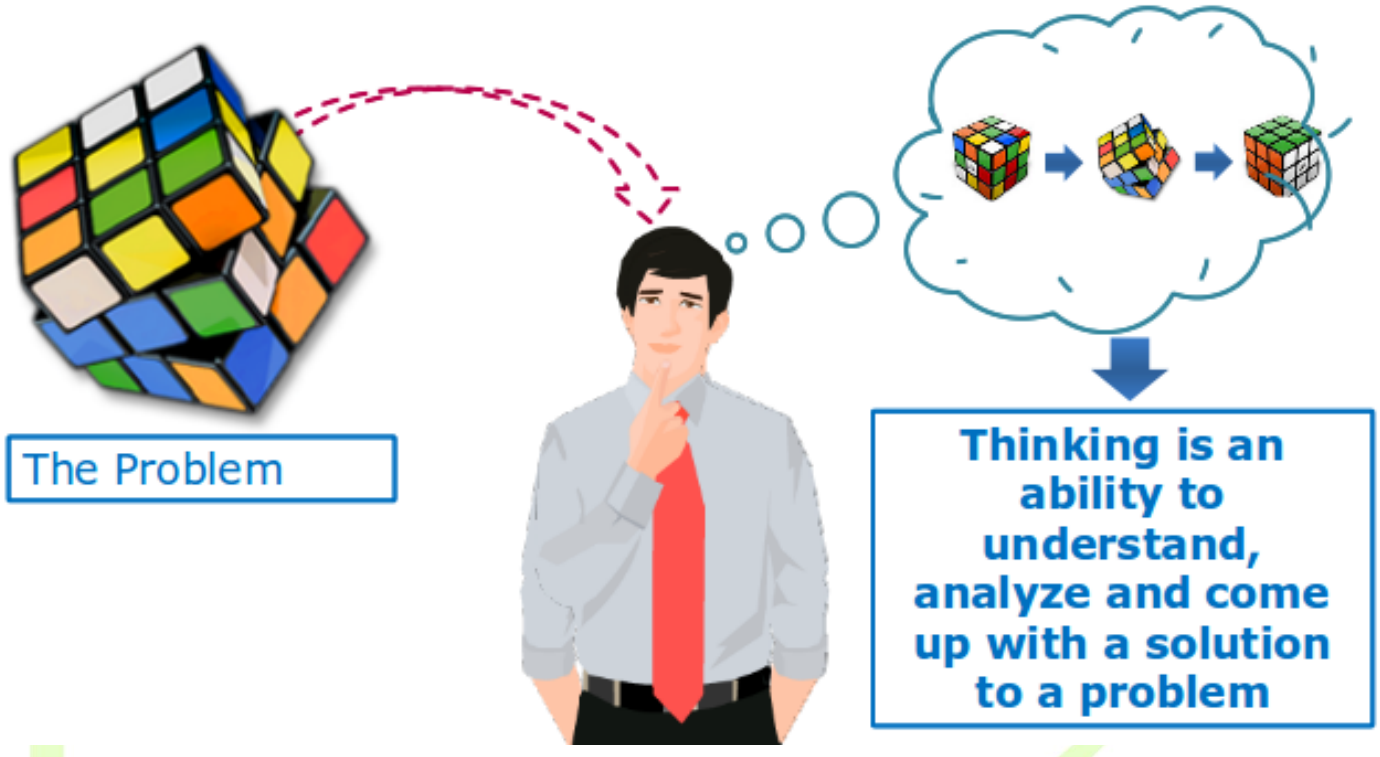
Version 1.0.4

Learning Objectives

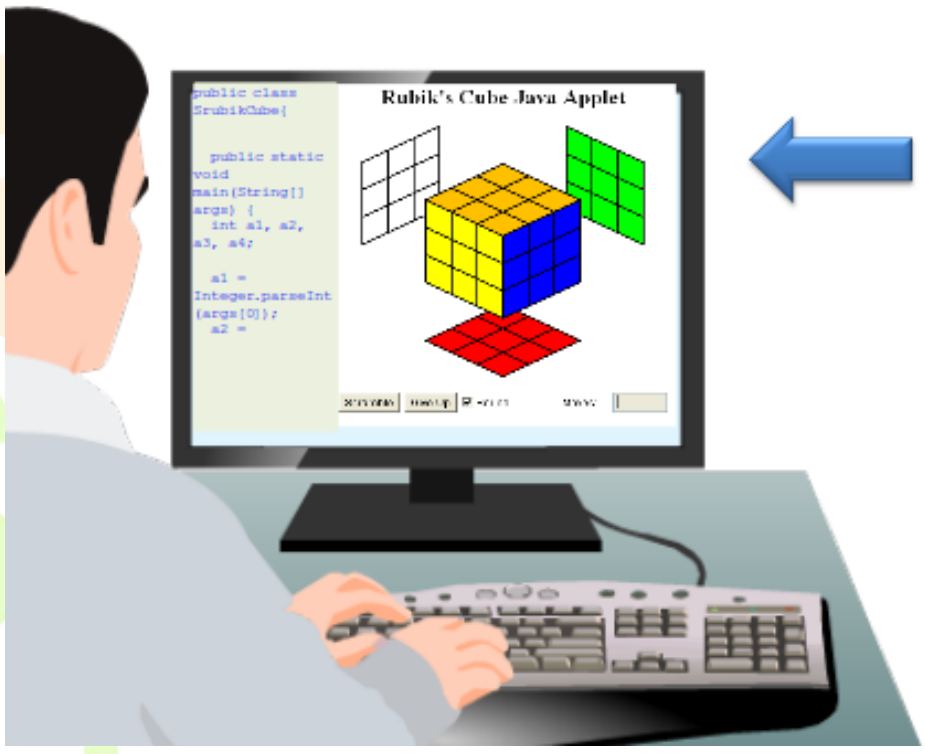
The content in this presentation is aimed at teaching learners to:

- Explain programming thinking
- Explain the fundamental elements of programming
- Use command-line arguments in Java programs
- Explain common errors associated with command-line arguments

Command-Line Arguments



Command-Line Arguments



Programming thinking is to come up with the solutions in a structured manner for the (generally computational) problems.

Command-Line Arguments

Let's think

Problem Find sum of four given numbers.

High Level Solution Find the sum by considering one number at a time and adding it to the sum.

Detailed Solution

- Take the first number and second number.
- Add them and call it sum.
- Take third number and add it to sum.
- Take fourth number and add it to sum.
- Print sum.

Command-Line Arguments

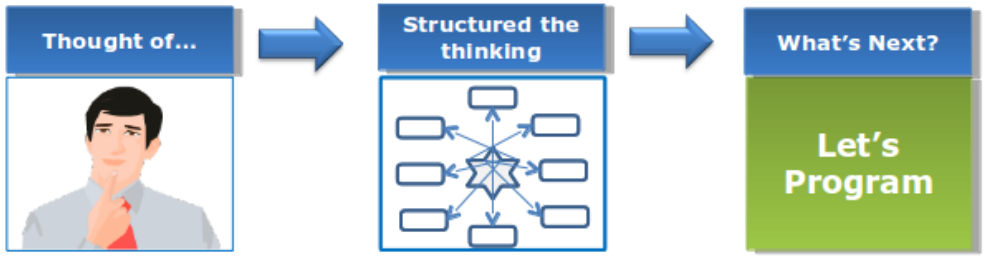
Let's structure our thinking

```
a1      =   First Number
a2      =   Second Number
sum     =   a1 + a2
a3      =   Third Number
sum     =   sum + a3
a4      =   Fourth Number
sum     =   sum + a4
```

print sum

Command-Line Arguments

What's Next...




```
sumSoFar = next;
 next = Integer.parseInt(args[1]);
 sumSoFar += next;
 next = Integer.parseInt(args[2]);
 sumSoFar += next;
 next = Integer.parseInt(args[3]);
```

Command-Line Arguments

Building Blocks of a Program

We will construct the following expressions, which we will use in our FIRST Java program:

```
a1 = Integer.parseInt(args[0]); //Read first number
a2 = Integer.parseInt(args[1]); // Read second number
sum = a1 + a2; // add a1 and a2 and assign to sum
a3 = Integer.parseInt(args[2]); // Read third number
sum = sum + a3; // add a3 to sum
a4 = Integer.parseInt(args[3]); // Read fourth number
sum += a4; // Add a4 to sum
System.out.println(sum); //print sum
```

Command-Line Arguments

- Java application can accept any number of arguments directly from the command-line.
- Users can enter command-line arguments when invoking the application.
- When running the java program with java command, the arguments are provided after the name of the class, separated by space.

Command-Line Arguments

Receive Command-line Arguments - Main

- In Java, when you invoke an application, the runtime system passes the command-line arguments to the application's main method as an array of Strings.

  public static void main(String[] args)

- Each String in the array contains one of the command-line arguments.

Command-Line Arguments

Sample Program

```
class CMDArgs {
    public static void main(String[] args) {
        System.out.println("Hello: " + args[0]);
    }
}
```

Compile javac CMDArgs.java

Run java CMDArgs TalentSprint

Output Hello TalentSprint

Command-Line Arguments

1. Write a program which accepts two names as input values and print “Hello <name1> and <name2>” as output.
   Note: Use command-line arguments
2. Create a file Wish.java and pass two command-line arguments. First argument is AM/PM and second argument is your name. Program should print “Good Morning <your name>”, if first argument is ‘AM’. If the argument is ‘PM’ then it should print “Good Evening <your name>”.

Note

If first argument is not equal to “AM” or “PM” then it should print “First argument should be either ‘AM’ or ‘PM’ ”.

Command-Line Arguments

Conversion of Command-line Argument

- If your program needs to support a numeric command-line argument, it must convert a String argument that represents a number, such as "34", to a number.
- The following is a code snippet that converts a command-line argument to an integer:

```
int firstArg = 0;
if  (args.length > 0)
     firstArg = Integer.parseInt(args[0]);
```

The parseInt() method in the Integer class throws a NumberFormatException(ERROR), if the format of args[0] isn't valid (not a number).

Command-Line Arguments

- Before using command-line arguments, always check the number of arguments, that too, before accessing the array elements, so that there will be no exception generated.
- For example, if your program needs the user to input 5 arguments, then:

```
if (args.length != 5) {
    System.out.println("Invalid number of arguments");
    System.out.println("Please enter 5 arguments");
    return;
}
```

Command-Line Arguments

Adding Two Values Using Command Line Arguments

```
class SumOfTwoNums {
    public static void main(String[] args) {
        // convert first value into integer
        int firstNum = Integer.parseInt(args0]);
        // convert second value into integer
        int secondNum = Integer.parseInt(args[1]);
        int sum = firstNum + secondNum;
        System.out.println("Sum: " + sum);
    }
}
```

Compile javac SumOfTwoNums.java

Run java SumOfTwoNums 10 20

Output Sum: 30

Command-Line Arguments

Execute the previous program by passing different values:

```
1 java SumOfTwoNums 30 40
2 java SumOfTwoNums 20 30 40
3 java SumOfTwoNums 30
4 java SumOfTwoNums 30 A
5 java SumOfTwoNums abc ABC
```

Command-Line Arguments

Try to run the following program and see what happens.

```
class CMDArgs1 {
    public static void main(String[] args) {
        int intAge = Integer.parseInt(args[1]);
        String strName = args[2];
        System.out.println("Age: " + intAge);
        System.out.println("Name: " + strName);
    }
}
```

Command-Line Arguments

Possible Errors With Command Line Arguments

ArrayIndexOutOfBoundsException If user does not pass sufficient number of values (or) array index is beyond its size.

NumberFormatException If user pass alphabets or symbols, where integers are expected. (or) while converting alphabets or symbols to integers.

Command-Line Arguments

Operators

Arithmetic Operators + - * /

Assignment Operators = += -= *= /=

Relational Operators < <= > >= == !=

Can you name these operators and operations they perform on operands?

Command-Line Arguments

Operators

An operator represents an operation to be performed on the given operands.

Expressions

```
int sum1 = a1 + a2;
      or
System.out.println("Sum: " + sum);
```

An expression is a construct made up of variables, operators, and method invocations, which are constructed according to the syntax of the language, that evaluates to a single value.

Command-Line Arguments