OOP with Java

11. Command Line Arguments

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Command Line Arguments

- Usually, a console program takes some arguments

Command Line Arguments

- Usually, a console program takes some arguments
- Under linux, `echo bla` runs program `echo` and passes argument `bla` to it, causing it to print `bla` to its `stdout`

Command Line Arguments

- Usually, a console program takes some arguments
- Under linux, `echo bla` runs program `echo` and passes argument `bla` to it, causing it to print `bla` to its `stdout`
- `java -version` runs the program `java` and passes to it the argument `-version`

Command Line Arguments

- Usually, a console program takes some arguments
- Under linux, `echo bla` runs program `echo` and passes argument `bla` to it, causing it to print `bla` to its `stdout`
- `java -version` runs the program `java` and passes to it the argument `-version`
- `java PrintCommandLineArguments` runs the program `java` and passes to it the argument `PrintCommandLineArguments` (which tells it to start the JVM and executes program `PrintCommandLineArguments`)

Command Line Arguments

- Usually, a console program takes some arguments
- Under linux, `echo bla` runs program `echo` and passes argument `bla` to it, causing it to print `bla` to its `stdout`
- `java -version` runs the program `java` and passes to it the argument `-version`
- `java PrintCommandLineArguments` runs the program `java` and passes to it the argument `PrintCommandLineArguments` (which tells it to start the JVM and executes program `PrintCommandLineArguments`)
- `java PrintCommandLineArguments A B` runs the program `java` and passes to it the arguments `PrintCommandLineArguments`, `A`, and `B` (which it to start the JVM and executes program `PrintCommandLineArguments` with command line arguments `A` and `B`)

Command Line Arguments

- Usually, a console program takes some arguments
- Under linux, `echo bla` runs program `echo` and passes argument `bla` to it, causing it to print `bla` to its `stdout`
- `java -version` runs the program `java` and passes to it the argument `-version`
- `java PrintCommandLineArguments` runs the program `java` and passes to it the argument `PrintCommandLineArguments` (which tells it to start the JVM and executes program `PrintCommandLineArguments`)
- `java PrintCommandLineArguments A B` runs the program `java` and passes to it the arguments `PrintCommandLineArguments`, `A`, and `B` (which it to start the JVM and executes program `PrintCommandLineArguments` with command line arguments `A` and `B`)
- Besides via `stdin`, we now have a second option to pass in data to a program

Command Line Arguments

- Usually, a console program takes some arguments
- Under linux, `echo bla` runs program `echo` and passes argument `bla` to it, causing it to print `bla` to its `stdout`
- `java -version` runs the program `java` and passes to it the argument `-version`
- `java PrintCommandLineArguments` runs the program `java` and passes to it the argument `PrintCommandLineArguments` (which tells it to start the JVM and executes program `PrintCommandLineArguments`)
- `java PrintCommandLineArguments A B` runs the program `java` and passes to it the arguments `PrintCommandLineArguments`, `A`, and `B` (which it to start the JVM and executes program `PrintCommandLineArguments` with command line arguments `A` and `B`)
- Besides via `stdin`, we now have a second option to pass in data to a program
- Command line arguments are used to set parameters and pass in data of small size, `stdin` can be used for arbitrary size unstructured data

Processing Command Line Arguments

- the `main` method of any Java program takes one parameter `args` wich is a String array

Processing Command Line Arguments

- the `main` method of any Java program takes one parameter `args` wich is a String array
- this parameter contains the command line arguments passed to the program, each one as a String

Printing Command Line Arguments

Listing: Printing Command Line Arguments.

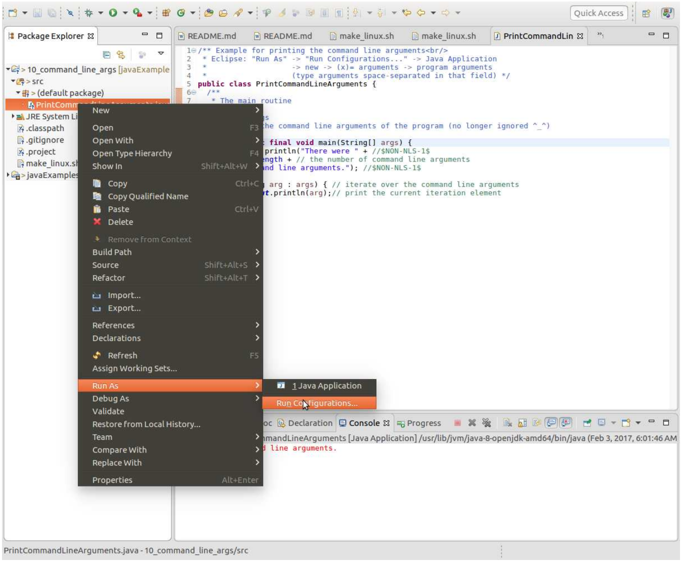
```
/** Example for printing the command line arguments<br/>
 * Eclipse: "Run As" -> "Run Configurations..." -> Java Application
 *                     -> new -> (x)= arguments -> program arguments
 *                     (type arguments space-separated in that field) */
public class PrintCommandLineArguments {

  /**The main routine
   * @param args
   *          the command line arguments of the program (no longer ignored ^_^) */
  public static final void main(String[] args) {
    System.err.println("There␣were␣" + //$NON-NLS-1$
        args.length + // the number of command line arguments
        "␣command␣line␣arguments."); //$NON-NLS-1$

    for (String arg : args) { // iterate over the command line arguments
      System.out.println(arg);// print the current iteration element
    }
  }
}
```

Greetings Printer Revised: Now with Args

Listing: Greetings Printer Revised: Now with Command Line Arguments.

```java
/** Examples for using command line arguments (and if-then-else) */
public class HelloSwitchCaseArgs {

  /** The main routine
   * @param args
   *          the command line arguments of the program (no longer ignored ^_^) */
  public static final void main(String[] args) {

    if (args.length < 2) { // check if there are at least two arguments, if not, print help and exit
      System.out.println("Please␣specify␣two␣arguments:␣your␣family␣name␣and␣your␣gender."); //$NON-NLS-1$
    } else { // OK, we have at least two command line arguments
      System.out.print("Hello␣"); // just print hello //$NON-NLS-1$
      switch (args[1]) { // choose what to do based on gender provided as second command line argument
        case "f": //$NON-NLS-1$
        case "F": {// we will get here if gender is either "f" or "F" //$NON-NLS-1$
          System.out.print("Mrs.␣"); //$NON-NLS-1$
          break;
        }

        case "m": //$NON-NLS-1$
        case "M": {// we will get here if gender is either "m" or "M" //$NON-NLS-1$
          System.out.print("Mr.␣"); //$NON-NLS-1$
          break;
        }

        default: { // we will get here if the gender is neither "f", "F", "m", "M"
          System.out.print(args[1]);
          System.out.print('␣');
          break;
        }
      }
      System.out.println(args[0]); // print family name, the first command line arg
    }
  }
}
```

How to Specify Command Line Args in Eclipse

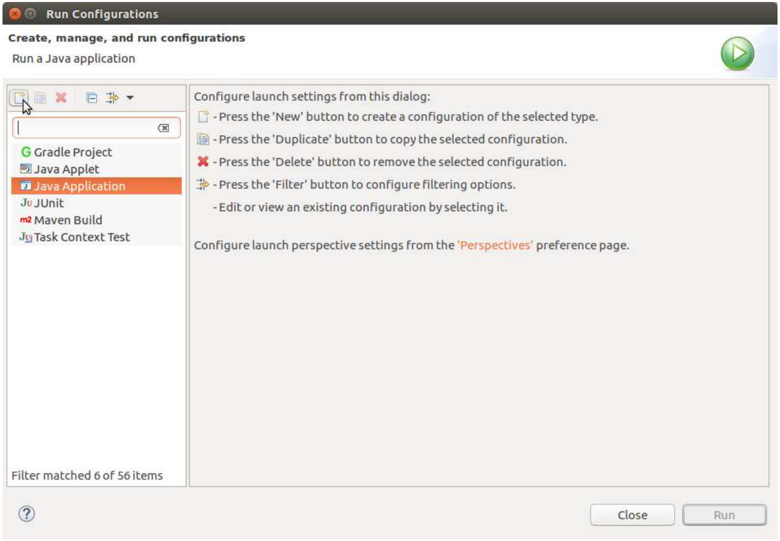


```
/** Example for printing the command line arguments<br/>
 * Eclipse: "Run As" -> "Run Configurations..." -> Java Application
 *                     -> new -> (x)= arguments -> program arguments
 *                     (type arguments space-separated in that field) */
public class PrintCommandLineArguments {
  /**
   * The main routine
          s
         the command line arguments of the program (no longer ignored ^_^)

         final void main(String[] args) {
         println("There were " + //$NON-NLS-1$
         ength + // the number of command line arguments
         nd line arguments."); //$NON-NLS-1$

         arg : args) { // iterate over the command line arguments
         t.println(arg);// print the current iteration element
```

```
mandLineArguments [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 3, 2017, 6:01:46 AM
d line arguments.
```

PrintCommandLineArguments.java - 10_command_line_args/src

How to Specify Command Line Args in Eclipse



Create, manage, and run configurations

Run a Java application

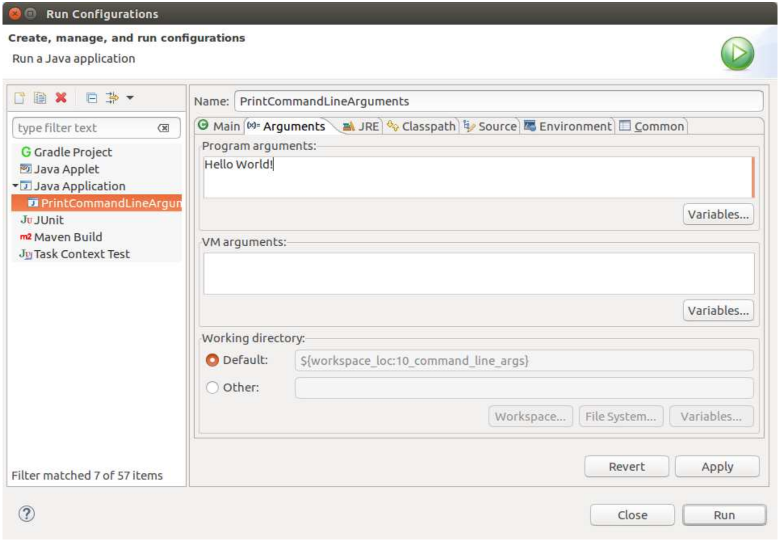
Gradle Project
Java Applet
Java Application
JUnit
Maven Build
Task Context Test

Filter matched 6 of 56 items

Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

How to Specify Command Line Args in Eclipse



Name: PrintCommandLineArguments

Program arguments:

```
Hello World!
```

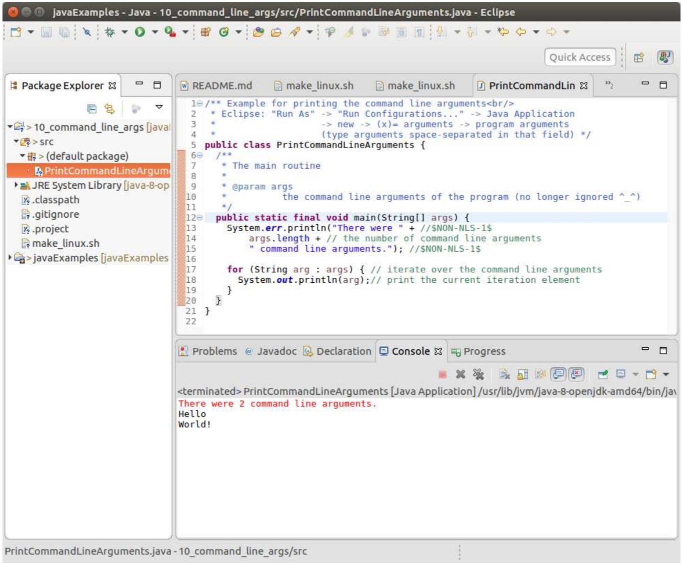
VM arguments:

Working directory:

[✓ Default:] ${workspace_loc:10_command_line_args}

Other:

How to Specify Command Line Args in Eclipse


javaExamples - Java - 10_command_line_args/src/PrintCommandLineArguments.java - Eclipse
Quick Access
Package Explorer
> 10_command_line_args [java
> src
> (default package)
PrintCommandLineArgum
JRE System Library [java-8-op
.classpath
.gitignore
.project
make_linux.sh
> javaExamples [javaExamples
README.md
make_linux.sh
make_linux.sh
PrintCommandLin
Problems
Javadoc
Declaration
Console
Progress
PrintCommandLineArguments.java - 10_command_line_args/src

```java
/** Example for printing the command line arguments<br/>
 * Eclipse: "Run As" -> "Run Configurations..." -> Java Application
 *                     -> new -> (x)= arguments -> program arguments
 *                     (type arguments space-separated in that field) */
public class PrintCommandLineArguments {
  /**
   * The main routine
   *
   * @param args
   *          the command line arguments of the program (no longer ignored ^_^)
   */
  public static final void main(String[] args) {
    System.err.println("There were " + //$NON-NLS-1$
        args.length + // the number of command line arguments
        " command line arguments."); //$NON-NLS-1$

    for (String arg : args) { // iterate over the command line arguments
      System.out.println(arg);// print the current iteration element
    }
  }
}
```

```
<terminated> PrintCommandLineArguments [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/ja
There were 2 command line arguments.
Hello
World!
```

Summary

- We have learned what command line arguments are: A second way to pass data to a program (the first is `stdin`)
- We have learned how to receive them in a Java program: via `String[]` parameter of `main` method
- We have seen how to specify them in Eclipse if we want to run a program

谢谢
Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, "Der Wanderer über dem Nebelmeer", 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog