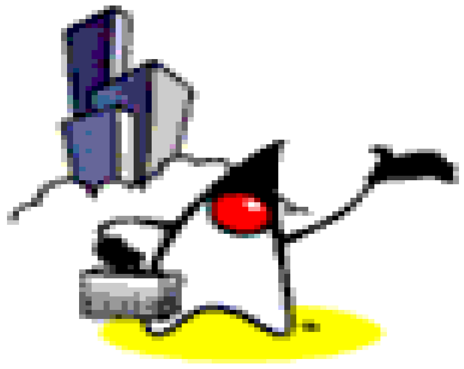
# Command-line Arguments

Sang Shin
JPassion.com
“Code with Passion!”

## Topics

- How to pass Command-line arguments
- How to pass Command-line arguments in NetBeans?
- How to pass Command-line arguments in Eclipse?



## How to pass Command-line Arguments

## Why Command-line Arguments?

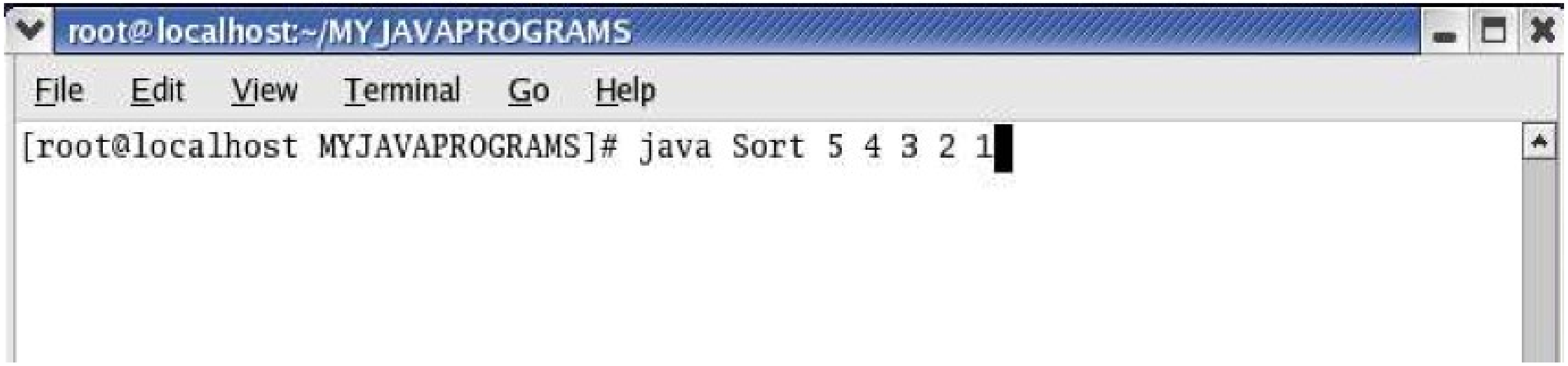
- A Java application can accept any number of arguments from the command-line

- Command-line arguments allow a user to change the behavior of an application
  - Application will perform different logic for different values of the arguments

- The user enters command-line arguments when invoking the application and specifies them after the name of the main class to run

## How to pass Command-line Arguments?

- Java application called “Sort” takes numbers to be sorted as command line arguments
- The arguments are separated by spaces


root@localhost:~/MYJAVAPROGRAMS
File Edit View Terminal Go Help

```
[root@localhost MYJAVAPROGRAMS]# java Sort 5 4 3 2 1
```

## How Java Application Receive Command-line Arguments

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

```
public static void main( String[] args )
```

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

## args[ ] array

- Given the previous example where we run:

  ```
  java Sort  5   4   3   2   1
  ```

  the arguments are stored in the args array of the main method declaration

## Example: Receiving Command-line Arguments

- To print the array of arguments, we write:

```
public class CommandLineSample {
        public static void main( String[] args ){

                for(int i=0; i<args.length; i++){
                        System.out.println( args[i] );
                }

        }
}
```

## Conversion of Command-line Arguments

- 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.
- Here's 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).

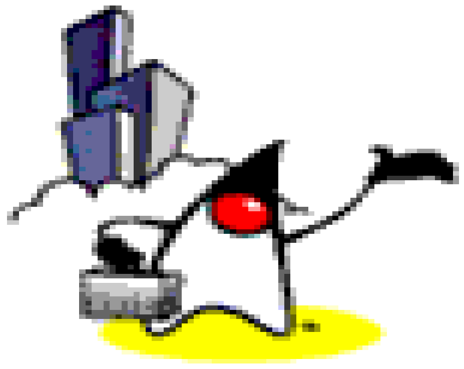
## Command-line Arguments: Coding Guidelines

- Before using command-line arguments, always check the number of arguments 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,

```
if( args.length!= 5 ){
        System.out.println(“Invalid number of arguments”);
        System.out.println(“Please enter 5 arguments”);
}
else{ // Correct number of arguments are passed
        //some statements here
}
```

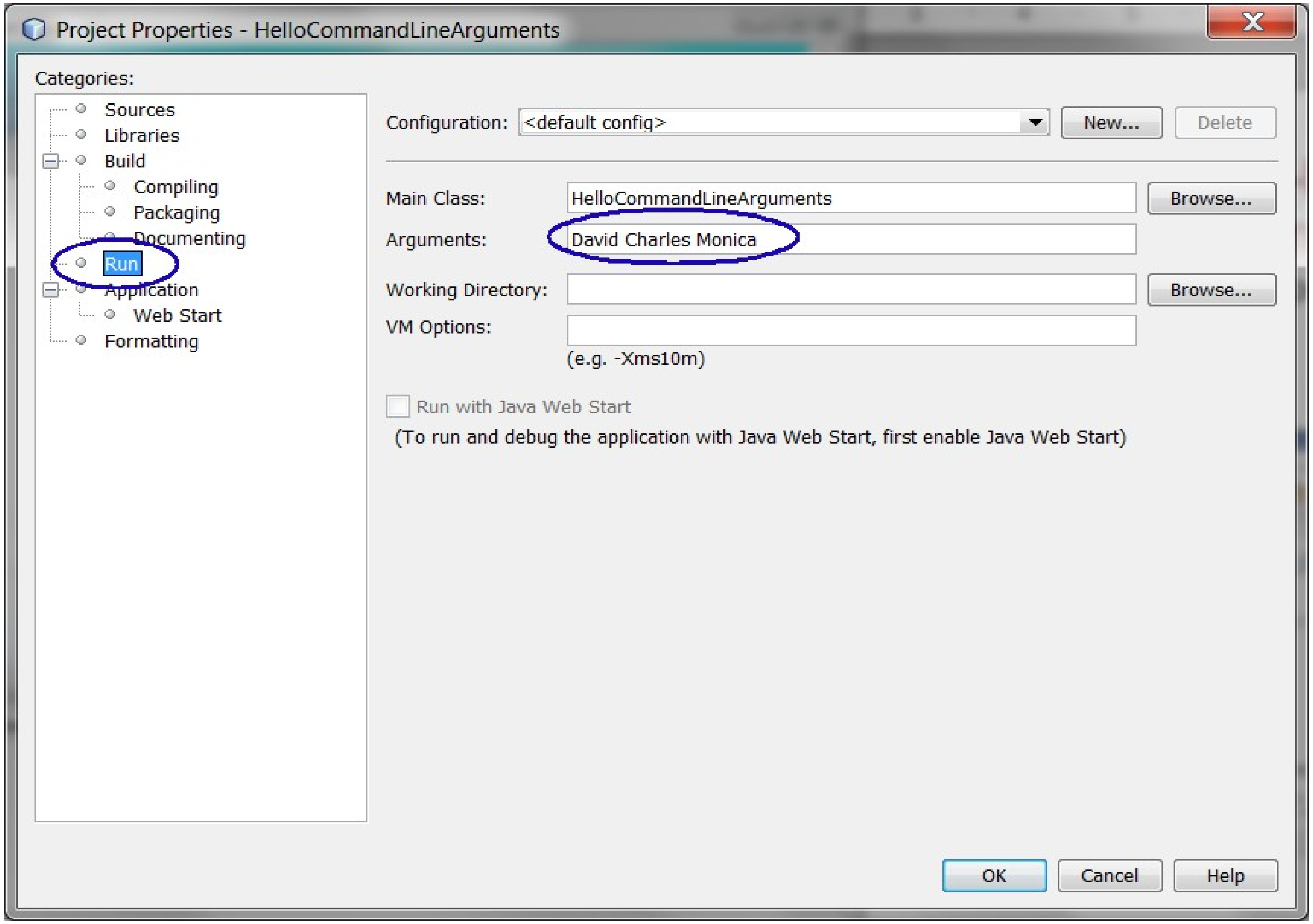
# Lab:

**Exercise 1: Read command-line arguments**
**1038_javase_commandarguments.zip**



## How to pass Command-line Arguments in NetBeans?

# Passing Command-line Arguments in NetBeans


Project Properties - HelloCommandLineArguments

Categories:
- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Application
  - Web Start
- Formatting

Configuration: <default config>

Main Class: HelloCommandLineArguments

Arguments: David Charles Monica

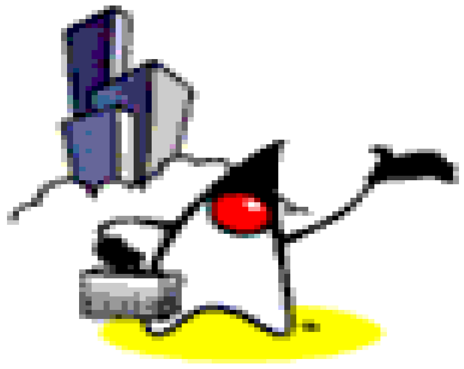
Working Directory:

VM Options:

(e.g. -Xms10m)

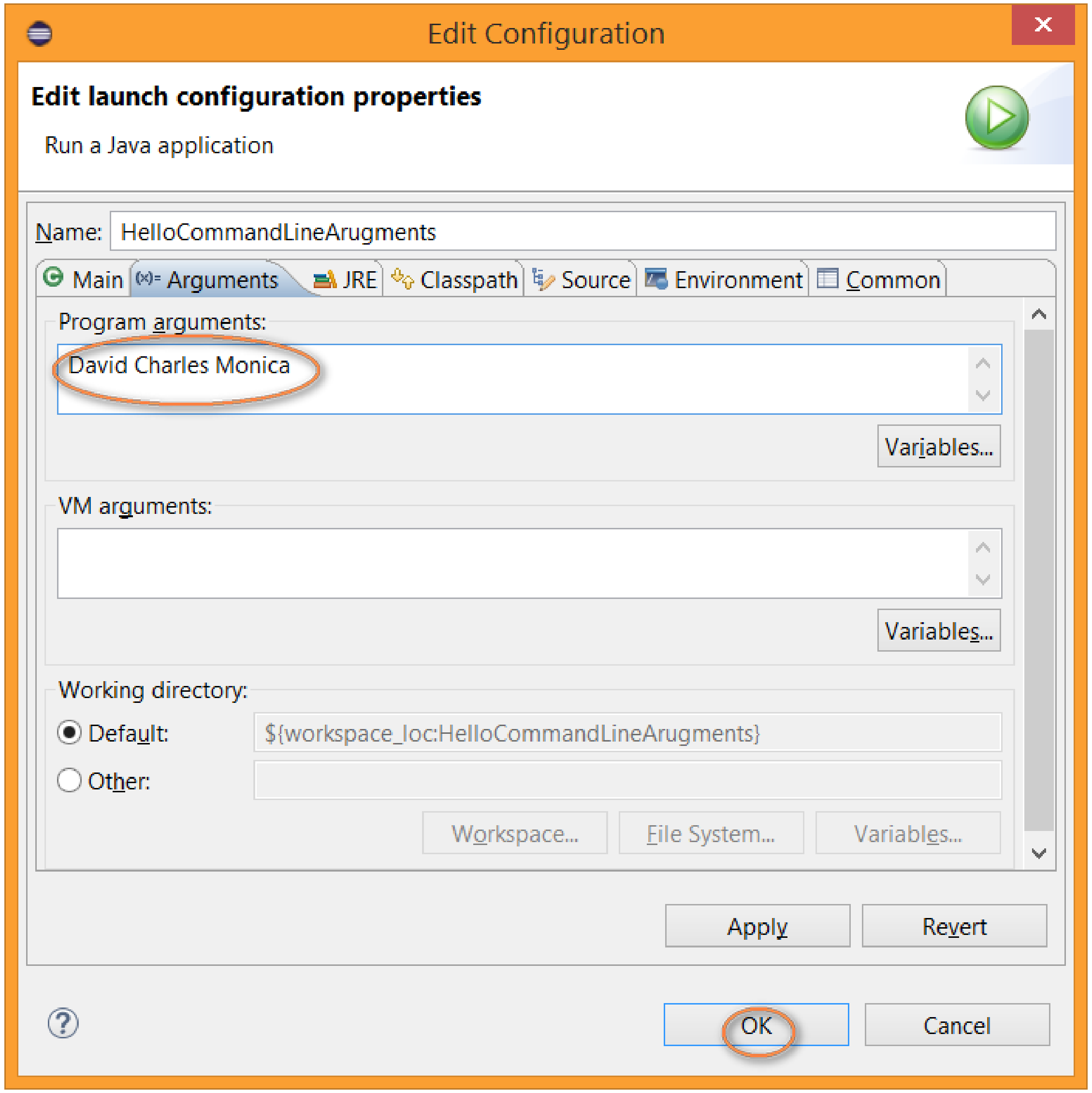
Run with Java Web Start

(To run and debug the application with Java Web Start, first enable Java Web Start)



## How to pass Command-line Arguments in Eclipse?

# Passing Command-line Arguments in Eclipse



Edit Configuration

**Edit launch configuration properties**

Run a Java application

Name: HelloCommandLineArugments

Program arguments:

David Charles Monica

VM arguments:

Working directory:

[✓ Default:] ${workspace_loc:HelloCommandLineArugments}

Other:

# Lab:

**Exercise 2: Read command-line arguments in NetBeans/Eclipse 1038_javase_commandarguments.zip**

Code with Passion!
JPassion.com