Command line argument

Prepared by

Dr. Rajesh Kumar Ojha
Asst. Prof., CSE, Silicon University

Options for User Input

- Options for getting information from the user
  - Write event-driven code
    - Con: requires a significant amount of new code to set-up
    - Pro: the most versatile.
  - Use System.in
    - Con: less versatile then event-driven
    - Pro: requires less new code
  - Use the command line (String[ ] args)
    - Con: very limited in its use
    - Pro: the simplest to set up

Using the command line

- Remember what causes the "big bang" in our programs?

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

Using the command line

- Remember what causes the "big bang" in our programs?

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

- `main` expects an array of strings as a parameter.
- The fact of the matter is that this array has been a null array in each of our programs so far.

Using the command line

- However, we can give this array values by providing command line arguments when we ***start*** a program running.

Using the command line

- However, we can give this array values by providing command line arguments when we start a program running.

```
java MyProgram String1 String2 String3
```

Using the command line

- However, we can give this array values by providing command line arguments when we start a program running.

```
$ java MyProgram String1 String2 String3

                  args[0] args[1] args[2]
                     ↑       ↑       ↑
```

Using the command line

- We can use this to get information from the user when the program is started:

```
public class Echo {
  public static void main(String [] args) {
      System.out.println("args[0] is " + args[0]);
      System.out.println("args[1] is " + args[1]);
  } // end main
} // end Echo class


$ javac Echo.java
$ java Echo Mark Fienup
args[0] is Mark
args[1] is Fienup
```

What are some of the problems with this solution

- This works great if we "behave" and enter two arguments. But what if we don't?

```
$ java Echo Mark Alan Fienup
args[0] is Mark
args[1] is Alan
```

(no problem, but Fienup gets ignored)

```
$ java Echo Mark
args[0] is Mark
Exception in thread "main"
          java.lang.ArrayIndexOutOfBoundsException:
```

(Big problem!)

Fixing this problem

- There are several ways to work around this problem
  - Use Java's exception handling mechanism (not ready to talk about this yet)
  - Write your own simple check and handle it yourself

```
public class MyEcho2 {
    public static void main( String[] args ) {
       if (args.length == 2) {
         System.out.println("args[0] is ” + args[0]);
         System.out.println("args[1] is " + args[1]);
       } else {
         System.out.println( "Usage:  java MyEcho2 "
                                  + "string1 string2");
       } // end if
     } // end main
} // end MyEcho2
```

Fixing this problem

```
public class MyEcho2 {
   public static void main( String[] args ) {
      if (args.length == 2) {
        System.out.println("args[0] is ” + args[0]);
        System.out.println("args[1] is " + args[1]);
      } else {
        System.out.println( "Usage:  java MyEcho2 "
                                  + "string1 string2");
      } // end if
    } // end main
} // end MyEcho2
```

Output:

```
$ java MyEcho2 Mark
Usage:  java MyEcho2 string1 string2
$ java MyEcho2 Mark Alan Fienup
Usage:  java MyEcho2 string1 string2
```

I learned something new!

- Will this code work if the user types NO arguments:
  "java MyEcho2"?

```
public class MyEcho2 {
    public static void main( String[] args ) {
       if (args.length == 2) {
         System.out.println("args[0] is " + args[0]);
         System.out.println("args[1] is " + args[1]);
       } else {
         System.out.println( "Usage:  java MyEcho2 "
                                  + "string1 string2");
       } // end if
     } // end main
} // end MyEcho2
```

Yes!

- The args array reference could be “null”, so doing args.length would cause an error!
- But it is not, since args is an array reference to an actual array with zero elements.

What about?

```
public class TestArray {
   public static void main( String[] args ) {
      System.out.println("args.length = " + args.length );
      String[] temp0 = {};
      System.out.println( "temp0.length = " + temp0.length );
      String[] tempNull;
      System.out.println("tempNull.length=" +
  tempNull.length);
    } // end main
} // end TestArray class


$ javac TestArray.java
TestArray.java:7: variable tempNull might not have been
  initialized
        System.out.println( "tempNull.length = " +
  tempNull.length );
                                                    ^
error
```

What about?

```
public class TestArray {
   public static void main( String[] args ) {
      System.out.println("args.length = " + args.length );
      String[] temp0 = {};
      System.out.println( "temp0.length = " + temp0.length );
      String[] tempNull = null;
      System.out.println("tempNull.length=" + tempNull.length);
    } // end main
} // end TestArray class
```

Output:

```
$ java TestArray
args.length = 0
temp0.length = 0
Exception in thread "main" java.lang.NullPointerException
        at TestArray.main(TestArray.java:7)
```

Your turn to write a simple program using the command line

- Write a program to echo all command-line arguments to the System.out

```
$ java EchoAll This is a long line
args[0] is This
args[1] is is
args[2] is a
args[3] is long
args[4] is line
```

Simple Calculations at Command Line

```
$ java Calculate 6 + 4
10
$ java Calculate 8 - 5
3
```

First Attempt - What's wrong?

```
public class Calculate {
    public static void main( String[] args ) {
        int    operand1 = args[0];
        String operator = args[1];
        int    operand2 = args[2];
        if ( operator.equals("+") ) {
           System.out.println( operand1 + operand2 );
        } else if ( operator.equals("-") ) {
           System.out.println( operand1 - operand2 );
        } else {
           System.out.println("Invalid operator: " + operator);
        } // end if
    } // end main
} // end Calculate
$ javac Calculate.java
Calculate.java:3: incompatible types
found   : java.lang.String
required: int
        int    operand1 = args[0];
                          ^
```

Correct Calculate

```
public class Calculate {
    public static void main( String[] args ) {
        int    operand1 = Integer.parseInt( args[0] );
        String operator = args[1];
        int    operand2 = Integer.parseInt( args[2] );
        if ( operator.equals("+") ) {
            System.out.println( operand1 + operand2 );
        } else if ( operator.equals("-") ) {
            System.out.println( operand1 - operand2 );
        } else {
            System.out.println( "Invalid operator: "
                                + operator );
        } // end if
    } // end main
} // end Calculate
```

Thank you