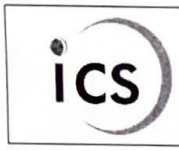

Rochester Institute of Technology
National Technical Institute for the Deaf
Information and Computing Studies Department

Name: Edward Riley

NACA.161 Programming Fundamentals II

In-Class Exercise #13 - Wrapper Classes and Command-Line Arguments

Objectives

Practice:

- Reading command-line input
- Converting strings to various primitive types.
- Use arrays

Overview

All command line arguments are of data type String. Since you cannot perform mathematical calculations directly on a string, you must convert the string to a numeric type. One way to do the conversion is using a "Wrapper class".

The conversion requires a two-step process:

1. Put the String object is inside a Wrapper object using a one-parameter constructor.
2. Call a method of the Wrapper class to convert the string into the desired primitive data type.

1) Create a class called **WrapperFun** that contains a main method.

2) Create code that displays the message "No command-line arguments were entered" if no command-line arguments were entered by the user.

   Write a relational expression that has a value of **true** if there are no command-line arguments. Note a relational expression uses the relational operators `>, <, >=, <=, ==, !=.`

   if (args.length == 0)

   Where are command-line arguments stored in the program?

   At the top of windows program

3) Execute your program with different numbers of arguments and make sure it works properly before continuing.

4) Add additional code to your program so that your code displays the message "nnn command-line arguments were entered" if there is at least one command-line argument. Note: nnn represents the number of arguments that the user entered. **Make sure that your code has only one if-statement.**

5) Execute your program with a various number of arguments and make sure it works properly before continuing.

6) Add additional code to the above to display the value of each command line argument as a **String**, assuming there is at one argument.

What type of loop did you use? For

What was the relational expression to end your loop? <

7) Run your program with the following sets of command line arguments:

```
[no arguments]
10 20 30 40
Jim 26 Bob 32 Leslie 28 Mary 27
```

Note that the number of blanks that you enter between each argument is arbitrary.

The output of your code should look like the following for the third set of data:

```
8 arguments were entered
Argument 0 is Jim
Argument 1 is 26
Argument 2 is Bob
Argument 3 is 32
Argument 4 is Leslie
Argument 5 is 28
Argument 6 is Mary
Argument 7 is 27
```

Remember that all arrays start with index 0.

8) From this point on, we will assume that all of the command-line arguments represent numbers –either integers or doubles.

9) The goal for the next steps will be to compute two sums. The first sum has all the integer values entered in command line. The second sum has all of the double values entered in the command line. For example, if the arguments are

```
2.2  4  1.3  2  -1  5.1  9
```

The sum of the integers is 14 [= 4 + 2 + (-1) + 9]
The sum of the doubles is 8.6 [=2.2 + 1.3 + 5.1]

10) You will need to convert the command-line argument to an integer if it contains an int value and convert the command-line argument to a double value if it contains a double.

What character inside the command-line argument determines if it contains an int or double? Remember a command-line argument is always a string.

indexOf(".")

11) What method in the String class can be used to determine if a string has a decimal point in it?

indexOf

12) Create another loop after the one that displays the command-line arguments as a string. Each time through the loop, create an **Integer** object for each command-line argument containing an int and create a **Double** object for each command-line argument containing a double.

If the argument string contains an int, write the statement to create an **Integer** object using the command-line argument to initialize it.

Integer.parseInt(args[i]);

If the argument string contains a double, write the statement to create a **Double** object using the argument-line argument to initialize it.

Double.parseDouble(args[i]);

13) What method of the **Integer** class can you call to extract the int value from the **Integer** object?

parseInt

14) What method of the **Double** class can you call to extract the double value from the **Double** object?

parseDouble(args[i]);

15) Compile your code and fix any errors that occur. Print the integer and double sums that you have computed as described above. Execute your until your program correctly prints the two sums.

16) Run your program with the command-line arguments of `22 3.4.5 13` which contains an invalid numeric value.

What exception did you get?

Multiple points

Why did you get this exception?

You cannot type in a value 3.4.5

When you complete all of the steps successfully and answer all of the questions, contact your instructor to check if your program(s) executes correctly and to review your code. We will initial the line below.

_____Successful execution of code

If you do not finish the program during the class period, contact your instructor or teaching assistant to initial below so that you can complete it before the next class period.

_____Code not completed during lab time

You may then have your instructor verify your work at the start of work period in the next class. If you do not have a signature, then you can not receive any points for this assignment.