2.08 Parsing Notes

**Do You Parse What I Am Asking?**

It’s surprisingly easy to crash a program by entering the wrong input. Have you ever mistyped your user name or password when logging into a website? Have you ever typed a word when a number was expected? Such mistakes are usually patiently (and politely) handled by the program, and you’re allowed to re-enter your information. Software developers devote a great deal of attention to catching and compensating for user errors to prevent crashes. The good news is that all input can be read as a String and then converted as necessary. This is a simple way to prevent program crashes, and it’s good programming practice.

### Part 1

Can you do math with Strings? Shakespeare might pose the problem thusly.

Public Domain

To use ***next()***, or not to use ***next()***: that is the question. Whether 'tis nobler in code to suffer the crashes and errors of outrageous user input, or to take precautions against legions of accidental or intentional user entry mistakes.

Had Shakespeare been a programmer he probably would have chosen to use the **next()** method as the lesser of two evils!

* The **nextInt()** and **nextDouble()** methods handle numbers correctly (unless a decimal is entered when an integer is expected); however, entering a **String** causes both methods to crash.
* The **next()** method handles both numeric and alphanumeric data by assigning all input to a **String** object; however, you can't do math with Strings.

If only there was a way for Java to do math with **Strings,** the problem would be solved. Alas, there isn't a Java method to do alphanumeric math, but somewhere in Java's vast API there must be a solution to convert a **String** to a number. Stay tuned for Act 2.

### Part 2

Are you familiar with the word **parse (to resolve into component parts)**? Generally it means to understand as in, "I don't parse the meaning of your statement to the committee, Mr. Undersecretary." Or you may have heard it in your English class in relation to analyzing the grammatical form, function, and interrelationship of each word in a sentence (i.e. diagramming a sentence).

In computer science, it means to separate into more easily processed parts.

When a number is read by the **next()** method, it is wrapped up as a String, and consequently no longer has any numerical value. However, Java has two methods for removing, or parsing, a number from its String wrapper so it can be used once again in an arithmetic expression to perform calculations.

### **TRY IT**

### Part 1

Computers are number-crunching, information-processing, analytical machines controlled by programmers just like you. However, sometimes data are not initially in the correct format to be crunched, processed, or analyzed and must be converted into a useable type. Java provides several useful techniques to convert back and forth between numeric and alphanumeric data types, which will make your programming tasks much easier to complete.

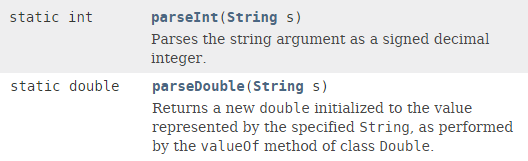
[**Complete the following eIMACS lab.**](https://l.flvsgl.com/GSL008bdee8c5e612ebebeb927fdd9b0ceef)

* Java Basics
  + Variables and Expressions
    - Converting between Numbers and Strings

For other practice options contact your instructor.

### Part 2

The **parseInt()** and **parseDouble()** methods are members of the **Integer** and **Double** classes, respectively. The Method Summaries from the Java API are shown below.



Converting numbers entered as **Strings** back to numeric values is a very simple process, as illustrated in the following demo program:

* Create a new project called 02.08 Parsing Examples in the Mod02 Lessons folder.
* Download the [**ParsePractice.java**](https://lti.flvsgl.com/flvs-cat-content/r7tmpltcc9i2gjfmjkqtf5ch4h/flvs-cat-session/apcomputersciencea_v20/module02/lesson08/docs/03_08b/parsepractice.java) file to the newly-created folder.
* Open the program and study the source code. You should conduct your own line-by-line desk check analysis of the program.
* After you understand the program, run it and observe the output.

Parsing **Strings** to numbers is just this easy. However, what happens if you enter thirty-two instead of 32, or if you include a non-numeric character like a comma, a dollar sign, or a negative sign as part of the input? Sometimes you solve one problem, and then you have to deal with new ones.

Please enter your name (first last): Cole Woodyard

Please your age in years: 18

Name: C. Woodyard Age in Minutes: 9466560

#### Answer Key

Top of Form

#### Question 1 **(Worth 2 points)**

Errors can be syntax errors or logic errors (the code works, but not as intended).  
  
Which of the following statements contains an error?

1. Scanner in = new Scanner(System.in);
2. System.out.print("Please enter your name (first last): ");
3. String firstName = in.Next();

 I only

 II only

 III only

 I and II only

 II and III only

Points earned on this question: **2**

#### Question 2 **(Worth 2 points)**

Errors can be syntax errors or logic errors (the code works, but not as intended).  
  
Which of the following statements does NOT contain an error?

1. String firstName = in.next();
2. String lastName = in.nextLine();
3. System.out.print("/n");

 I only

 II only

 III only

 I and II only

 II and III only

Points earned on this question: **0**

#### Question 3 **(Worth 2 points)**

Errors can be syntax errors or logic errors (the code works, but not as intended).  
  
Which of the following statements contains an error?

1. String userAge = in.next();
2. String ageInYears = Integer.parseInt(userAge);
3. double approxAgeInDays = ageInYears \* 365.25;

 I only

 II only

 III only

 I and II only

 II and III only

Points earned on this question: **0**

#### Question 4 **(Worth 2 points)**

Errors can be syntax errors or logic errors (the code works, but not as intended).  
  
Which of the following statements contains an error?

1. String firstInitial = firstName.substring(0,1);
2. String lastName = in.NextLine();
3. String name = firstInitial + ". " + lastName;

 I only

 II only

 III only

 I and II only

 II and III only

Points earned on this question: **0**

#### Question 5 **(Worth 2 points)**

Errors can be syntax errors or logic errors (the code works, but not as intended).  
  
Which of the following statements does NOT contain an error?

1. System.out.print("/tab Age in Minutes: " + ageInMinutes + "/n");
2. System.out.print("/t Age in Minutes: " + ageInMinutes + "/n");
3. System.out.print("\t Age in Minutes: " + ageInMinutes + "\n");

 I only

 II only

 III only

 I and II only

 II and III only

Points earned on this question: **2**

Bottom of Form