Mixing Types & Casting

**Content**

**Management of Complexity**

This activity will provide you with some experience with mixing variables and casting. As you create larger programs for other people, you will be manipulating different kinds of data and information. This will add complexity to your program, which is why it’s important to learn how to handle the variety of data types used in your program.

Imagine you ask a user to enter their age, you would expect them to enter an integer such as “34” or “14”. But it might surprise you when a user types “Fifteen”, “Sixty-seven” or “9.5”. In order to avoid your program from crashing, you want to be able to add complexity to the program so that you can handle these errors effectively.

*NOTE: even though we always write programs with a class header and main method, today we’re only going to focus on declaring variables, so we won’t be asking you to write those other segments of code. In real life, you would never evaluate statements standing alone! These expressions would normally be found in the context of code that looks similar to this:*

public class Example{

public static void main(String[] args){

System.out.prinln(1 + 2 – 3);

}

}

Evaluate the expressions below.  Make sure to use the rules of precedence (PEMDAS), and the rules about converting mixed types. Remember that werewolf double types “infect” int types, and zombie Strings “infect” double and int types.

Exercise 1

4 / 3

3 / 4

4.0 / 3

3.0 / 4

4.0 / 3.0

1.0 + 3 / 4

1 + 3 / (2 + 2.0)

9 / 10

9.0 / 10

Exercise 2

(1 + 1 + 1 + 1 + 1 + 1 + 1) / 8

(1 + 1 + 1 + 1 + 1 + 1.0 + 1) / 8

4 % 3

(100 / 27) \* 27 + 100 % 27

Exercise 3

"I like " + "zombies!"

"I have eaten " + 5 + " brains today."

"I have eaten " + 10.0 / 2 + " brains today."

"I have eaten " + 10 + 2 + " brains today."

1 + " teacher, " + 1.0 + " werewolf, and one zombie walk into a bar. Three zombies walk out!"

"What do vegan zombies eat? " + "GRAAAAAAAAAAINS!!!"

1 + 1 + 1 + 1 + 1 + "1" + 1 + 1 + 1 + 1

(1 + 1 + 1 + 1 + 1 + 1 + 1) / 8 + "oops"

(1 + 1 + 1 + 1 + 1 + 1.0 + 1) / 8 + "oops"

"What weird expression " + 3 \* 2 % 5 + 1 + " is this?"

"Halloween is on October 31; the next day is November " + (31 + 1) % 31

"I didn't expect the answer to be " + ((100 / 27) \* 27 + (100 % 27))

5 + "4" + 3 + 2 + 1 + " blastoff!"

" " + 0 + 0 + 0 + 0 + 0 + " SCARY"

Exercise 4

*When evaluating the expressions below, remember that you can cure an infection (double -> int) or cause infection (int -> double) by asking Java to convert to the type you want. This is called “casting” (like a Unicorn casts a magical spell to reverse the infection).*

*Examples: (int) 1.98 evaluates to 1*

*(double) 3 / 4 evaluates to 0.75*

(int) (1.8 \* 2.0)

(int) 1.8 \* 2.0

(double) 3 / 4

3 / (double) 4

(1 + 1 + 1 + 1 + (double)1 + 1 + 1) / 8

(int) (1 + 1 + 1 + 1 + 1.0 + 1 + 1) / 8

"I give this worksheet " + (int) 5.1 + " out of five stars."

"integer 3 / 4 is " + (3 / 4) + " but double 3 / 4 is " + (double)3 / 4

Exercise 5

*Add a cast in the blanks \_\_\_\_\_\_ to the following expressions to produce the desired result.*

(\_\_\_\_\_\_\_\_\_) 3 / 4 evaluates to : 0.75

"I got a " + (\_\_\_\_\_\_\_\_\_) 100.99 + " on my test!"  
  
 evaluates to : "I got a 100 on my test!"

(\_\_\_\_\_\_\_\_) 1 /  (\_\_\_\_\_\_\_\_) 2.1

evaluates to : 0.5

(\_\_\_\_\_\_\_\_) 1.5 + (\_\_\_\_\_\_\_\_\_) 3.4 / (\_\_\_\_\_\_\_\_\_) 4.1

evaluates to : 1.5

(\_\_\_\_\_\_\_\_\_) 0.85 + (\_\_\_\_\_\_\_\_\_) 0.75 + (\_\_\_\_\_\_\_\_) 0.99  
  
 evaluates to : 0