The Renegade Coder

CODE FIRST. ASK QUESTIONS LATER

Primitive Types: Solutions

By Jeremy Griffith

The following exercises were created to expose you to Java variable declarations, variable arithmetic, type casting, and strings.

Math with Variables

Now that you have a grasp on arithmetic, you can begin creating some variables and replicating some of the calculations from the last exercise. In the interest of style, make sure to give your variables descriptive names as shown in the tutorial. Try to answer the following before running any code, then verify your answer by using DrJava's interaction pane.

1. Make an integer that holds a value of 5. Make a double that holds a value of 3.5. Add the variables together and store the result in a new integer. What is the result?

```
int length = 5;                                   (5)

double width = 3.5;                               (3.5)

int lengthPlusWidth = length + width;             (8)
```

2. Make an integer that holds a value of 10. Make another integer that holds a value of 7. Using the variables, divide 10 by 7 and store the result in a new double. What is the result?

```
int miles = 10;                        (10)

int hours = 7;                         (7)

double mph = miles / hours;            (1.0)
```

3. Make a boolean that holds a value of true. Make a byte that holds a value of 15. Add the variables together without storing the result. What is the result?

```
boolean hasWon = true;

(true) byte gamesWon = 15;                        (15)

hasWon + gamesWon;                     (syntax error)
```

Once you have a feel for the types of comparisons you can make above, try to quickly answer the following without the interactions pane (*Hint: some of these are not legal*):

1. 1 + 1	(2)
2. 5 / 2.0	(2.5)
3. 5 + 'g'	(108 or 'l')
4. true + true	(syntax error)
5. 12 % 6	(0)
6. 'b' + 7	(105 or 'I')
7. -15 / 3	(-5)
8. Integer.MAX_VALUE + 1	(Integer.MIN_VALUE or -2147483648)
9. 5 == 7	(false)
10. 'w' > 'f'	(true)
11. 5 + 7 < 10	(false)

Type Casting

In the previous tutorial, we briefly discussed type widening when adding a 32-bit number to a 64-bit number. The result is stored in a 64-bit number. Type casting is used when you want to force the result to be a 32-bit number. Compute the following samples:

1.	(int) (5.0 / 2.0)	(2)
2.	(char) 56	(‘8’)
3.	(int) ‘b’	(98)
4.	(byte) 200	(-56)

String Introduction

As we head into a lesson on reference types, let’s get an intro to some of the string operations.

1.	“bat” + “man”	(“batman”)
2.	“bat” – “man”	(Bad type in numeric expression)
3.	“bat” + 5	(“bat5”)
4.	“bat” + 1.0	(“bat1.0”)
5.	“bat”.length()	(3)

Note: Number 5 is not a syntax you should get used to using. It’s just to demonstrate a point about reference types.