Practice with variables and types

1. Types. For each literal or expression, state its type (String, int, double, or boolean).

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Expression</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>387</td>
<td>int</td>
<td>&quot;pancakes&quot;</td>
<td>String</td>
</tr>
<tr>
<td>true</td>
<td>boolean</td>
<td>45.0</td>
<td>double</td>
</tr>
<tr>
<td>&quot;14&quot;</td>
<td>String</td>
<td>87.98515</td>
<td>double</td>
</tr>
<tr>
<td>&quot;false&quot;</td>
<td>String</td>
<td>15 &gt;= 71</td>
<td>boolean</td>
</tr>
<tr>
<td>31.6 + 7</td>
<td>double</td>
<td>(double)(int)93.2</td>
<td>double</td>
</tr>
</tbody>
</table>

2. Declaring and using variables. Only one of the following code snippets is valid (i.e., will compile without errors). Which is it, and what’s wrong with each of the others?

**Code snippet A:**

```java
int x = 3;
int y = 17;
int x = x + y;
```

Variable x is declared twice.

**Code snippet B:**

```java
int num = 42;
double anotherNum = 81;
num = anotherNum - num;
```

anotherNum - num has type double, which cannot be stored in num, an int

**Code snippet C:**

```java
int years = 18;
int months = 7;
double totalAge = years + months/12.0;
```

This one works!

**Code snippet D:**

```java
int p = 5;
int q = 43.7;
p = q;
```

Can’t assign a double (q) to an int (p).
3. **Casting.** For each of the following, add a cast to fix the type error.

```java
int i = 5;
double j = 21.3;
i = i + (int)j;
```

```java
int totalLabScore = 84;
int numLabs = 10;
double averageScore = (double)totalLabScore/numLabs;
```

Note: this doesn’t cause a compiler error, but it didn’t achieve the intended behavior of computing the average score, which should include the decimal places.

4. **Using variables.** Write a piece of code that asks the user to enter their height (as a number of feet and a number of inches, i.e., 5 7) and tells them their height in meters (i.e., 1.7018). (Note: there are 12 inches in a foot, and there are 3.28 feet in a meter.)

```java
System.out.println("Enter the number of feet");
int feet = keyboard.nextInt();
System.out.println("Enter the number of inches");
int inches = keyboard.nextInt();
System.out.println("Your height is: " + (feet + inches/12.0)/3.28);
```
Practice with if statements

1. Are they equivalent? Which of the following snippets of code do the same thing? That is, which print the same message(s) on every single input value for num?

**Code snippet A:**
```java
int num = keyboard.nextInt();
if(num > 54) {
    if(num > 82) {
        System.out.println("one");
    } else {
        System.out.println("two");
    }
} else {
    System.out.println("three");
}
```
- When num > 82, prints “one”
- When 82 ≥ num > 54, prints “two”
- When num ≤ 54, prints “three”

**Code snippet B:**
```java
int num = keyboard.nextInt();
if(num > 82) {
    System.out.println("one");
} else if(num > 54) {
    System.out.println("two");
} else {
    System.out.println("three");
}
```
- When num > 82, prints “one”
- When 82 ≥ num > 54, prints “two”
- When num ≤ 54, prints “three”

A and B do the same thing.

**Code snippet C:**
```java
int num = keyboard.nextInt();
if(num < 54) {
    System.out.println("three");
} else if(num > 82) {
    System.out.println("one");
} else {
    System.out.println("two");
}
```
- When num > 82, prints “one”
- When 82 ≥ num ≥ 54, prints “two”
- When num < 54, prints “three” and “two”

**Code snippet D:**
```java
int num = keyboard.nextInt();
if(num > 54) {
    if(num < 82) {
        System.out.println("two");
    }
} else if(num > 82) {
    System.out.println("one");
} else {
    System.out.println("three");
}
```
- When num > 82, prints nothing
- When 82 > num > 54, prints “two”
- When num ≤ 54, prints “three”
2. Old enough? Write some code that asks the user for their age and then prints out whether they are old enough to:

1. Vote (age 18)
2. Get a driver’s license in MA (age 16)
3. Rent a car (age 25)
4. Drink legally (age 21)

```java
System.out.println("How old are you?");
int age = keyboard.nextInt();

if(age >= 16) {
    System.out.println("You can drive");
    if(age >= 18) {
        System.out.println("You can vote");
        if(age >= 21) {
            System.out.println("You can drink");
            if(age >= 25) {
                System.out.println("You can rent a car");
            }
        }
    }
}

Other solutions are also possible.
3. **Scope.** Determine whether each of the following code snippets will compile successfully. If not, correct the error. Then determine what prints.

**Code snippet A:**

```java
int i = 5;
if(i > 2) {
    i = i * 7;
}
System.out.println(i);
```

Compiles and prints 35

**Code snippet B:**

```java
int i = 8;
int j = 0;
if(i % 2 == 0) {
    j = 4;
}
System.out.println(i + j);
The scope of j is only inside the if statement. Can fix this by declaring and initializing j before the if statement. With the above correction, 12 will print.
```

**Code snippet C:**

```java
int x = -3;
int y = -2;
if(x * y > 0) {
    int z = x + y;
    y = z * 2;
}
System.out.println(x + " " + y);
```

Compiles and prints “-3 -10”

**Code snippet D:**

```java
int num1 = 42;
int num2 = 0;
if(num1 < 10) {
    num2 = 3;
}
System.out.println(num2);
The compiler doesn’t know whether num2 will be initialized before the print statement. We can fix this by initializing num2 before the if statement. This prints 0.
```

4. **Seasons.** Write some code that asks the user to enter the current month (as an int, 1=January and 12=December) and then prints the season (Winter for Dec-Feb, Spring for Mar-May, Summer for June-Aug, Fall for Sep-Nov).

```java
System.out.println("What month is it?");
int month = keyboard.nextInt();

if(month == 12 || month <= 2) {
    System.out.println("Winter");
} else if(month <= 5) {
    System.out.println("Spring");
} else if(month <= 8) {
    System.out.println("Summer");
} else {
    System.out.println("Fall");
}
```
Practice with boolean expressions and order of operations

1. true or false? Evaluate each of the following boolean expressions when \(\text{int } x = 4\) and \(\text{int } y = 6\).

\[x \leq 5 \text{ || } y + x > 12 \text{ && } !(x \% 3 == 1)\]

true. We have \(x \leq 5\) is true, \(y + x > 12\) is false, and \(x \% 3 == 1\) is true, so \(!(x \% 3 == 1)\) is false. So our expression becomes \(true \text{ || } false \text{ && } false\), and the AND gets evaluated first, so \(true \text{ || } false\), which is true.

\[y/x > 1 \text{ && } x != 17\]

false. Since \(x\) and \(y\) are both \text{int}s, the quotient \(y/x\) is also an \text{int}, so it has value 1. We then have \(false \text{ && } true\), which is false.

\[!(y \% 4 \% 2 == 0 \text{ || } !(x + y / 3) >= y)\]

false. Simplifying this one step at a time, we have:

\[!(6 \% 4 \% 2 == 0 \text{ || } !((4 + 6 / 3) <= 6))\]

\[!(2 \% 2 == 0 \text{ || } !(4 + 2 <= 6))\]

\[!(true \text{ || } !true)\]

\[!true\]

false

2. What prints? What prints when each of the following pieces of code runs?

```java
int month = 2;
int day = 20;
System.out.println("Tomorrow is "+ month/day);
```

Tomorrow is 0

```java
int month = 2;
int day = 20;
System.out.println("Tomorrow is " + month + "/" + day);
```

Tomorrow is 2/20

```java
int age = 19;
System.out.println("In three years your age will be: " + age + 3);
System.out.println("Your age in three years is: " + (age + 3));
System.out.println(age + 3 + " is your age in three years");
```

In three years your age will be 193
Your age in three years is: 22
22 is your age in three years
3. Broken code. Assume that the declaration and initialization `int x = 7;` appears somewhere earlier in the code. None of the following pieces of code will compile without error. Make a small change to fix the error without changing the intended meaning of the code.

```java
if(!x < 17) {
    System.out.println("yes");
}

if(!(x < 17)) {
    System.out.println("yes");
}

Needs parenthesis around `x < 17` because `!` has higher precedence than `<.`

```java
int y = 4;
if(x < -1 || < y) {
    System.out.println("yes");
}

int y = 4;
if(x < -1 || x < y) {
    System.out.println("yes");
}

Needs a complete boolean expression on either side of the OR.

```java
if(10 >= x > 2) {
    System.out.println("yes");
}

if(10 >= x && x > 2) {
    System.out.println("yes");
}

Can’t chain inequalities.
Practice with loops

1. What prints? Consider the following while loop. What is the output?

```java
int i = 0;
while(i < 5) {
    int j = 0;
    while(j < 3) {
        System.out.print(i + j);
        j++;
    }
    System.out.println();
    i++;
}
```

```
012
123
234
345
456
```

2. while and for. Translate the following while loop into a for loop that does the same thing.

```java
int i = 0;
while(i < 100) {
    System.out.println(i * 7);
    i++;
}
```

```java
for(int i = 0; i < 100; i++) {
    System.out.println(i * 7);
}
```

3. Pretty patterns. Write some nested while loops that print the following pattern:

```
******
*    *
*    *
*    *
*    *
*    *
******
```
```java
int i = 0;
while(i < 6) {
    if(i == 0 || i == 5) {
        System.out.println("****");
    } else {
        System.out.println("*");
    }
    i++;
}

Another solution that prints an n by n border for any int n:

int i = 0;
while(i < n) {
    if(i == 0 || i == n - 1) {
        int j = 0;
        while(j < n) {
            System.out.print("*");
            j++;
        }
        System.out.println();
    } else {
        System.out.print("*");
        int j = 0;
        while(j < n - 2) {
            System.out.print(" ");
            j++;
        }
        System.out.println("*");
    }
    i++;
}
```

4. **Comparing code.** Do the following two pieces of code do the same thing? If so, what do they both do? If not, change the second in some small way so that they do the same thing.

**Code snippet A:**

```java
for(int i = 1; i <= 10; i++) {  
    System.out.println(i);
}
```

**Prints** 1 2 3 4 5 6 7 8 9 10

**Code snippet B:**
for(int i = 10; i > 0; i--) {
    System.out.println(10 - i);
}

Prints 0 1 2 3 4 5 6 7 8 9

We can get them to do the same thing by making the second loop counter i start at 11.

5. Improving code that already works. What is stylistically not so great about the following piece of code? Fix it to improve the code style without changing what it does.

```java
int i = 0;
while(i < 1) {
    System.out.print(i + " ");
    i++;
}
System.out.println();
i = 0;
while(i < 2) {
    System.out.print(i + " ");
    i++;
}
System.out.println();
i = 0;
while(i < 3) {
    System.out.print(i + " ");
    i++;
}
System.out.println();
i = 0;
while(i < 4) {
    System.out.print(i + " ");
    i++;
}
System.out.println();
```

Since we’re repeating what is essentially the same code four times in a row, it would be better to put it in some sort of loop to make the code more concise. Here’s one way to do it:

```java
for(int j = 1; j <= 4; j++) {
    int i = 0;
    while(i < j) {
        System.out.print(i + " ");
        i++;
    }
    System.out.println();
}
```