Java Language Fundamentals

AP CS Review #1

April 10, 2017

Question 1

Which of the following does NOT evaluate to 0.4?

A. (int) 4.5 / (double) 10;
B. (double) (4 / 10);
C. 4.0 / 10;
D. 4 / 10.0;
E. (double) 4 / (double) 10;

Question 1 - Answer

Which of the following does NOT evaluate to 0.4?

A. (int) 4.5 / (double) 10;
B. **(double) (4 / 10);**
C. 4.0 / 10;
D. 4 / 10.0;
E. (double) 4 / (double) 10;

Concepts

Casting happens before math operations.

(*unless otherwise noted by parenthesis)*

Questions:

What comes first: casting or the dot operator?

(dot operator)

When do we particularly need to pay attention to this?

(class casting)

Concepts

Types & Arithmetic.

(result of an operation will default to the more precise type)

int + int = ?
int + double = ?
double + int = ?
double + double = ?

Concepts

Types & Arithmetic.

(result of an operation will default to the more precise type)

int + int = int
int + double = double
double + int = double
double + double = double

For what math operation could the value of the result actually differ?

division

Exercise

Name all the *math operators.*
Name all the *logical operators*.
Name all the *relational operators.*
Name all the *assignment operators.*
Name all the *increment/decrement operators.*

Create a precedence table that includes all of the above operators.

While on this topic…

DeMorgan's Law

(distribute the negative, flip the logical operator)

Example:

!(y > 3 && x != y || x <= 20)

y <= 3 || x == y && x > 20

Can you “math”?

```
int result = 13 – 3 * 6 / 4 % 3;
```

What value does result contain?

//I won’t even grace you with options… this

//question should never be missed!

12

Question 2

What are the results of the following?

```
int x = 0;
int y = 50;
if (x / y > 5 && y != 0){
    System.out.print(“Success”);
}else{
    System.out.println(“Failure”);
}
```

A. Success B. Failure C. No Output
D. Arithmetic Exception E. Success Failure

Question 2 - Answer

What are the results of the following?

```
int x = 0;
int y = 50;
if (x / y > 5 && y != 0){
    System.out.print(“Success”);
}else{
    System.out.println(“Failure”);
}
```

A. Success **B. Failure** C. No Output
D. Arithmetic Exception E. Success Failure

Question 3

What are the results of the following?

```
int x = 50;
int y = 0;
if (x / y > 5 && y != 0){
    System.out.print(“Success”);
}else{
    System.out.println(“Failure”);
}
```

A. Success B. Failure C. No Output
D. Arithmetic Exception E. Success Failure

Question 3 - Answer

What are the results of the following?

```
int x = 50;
int y = 0;
if (x / y > 5 && y != 0){
    System.out.print(“Success”);
}else{
    System.out.println(“Failure”);
}
```

A. Success B. Failure C. No Output
D. Arithmetic Exception E. Success Failure

Question 4

What are the results of the following?

```
int x = 50;
int y = 100;
if (x / y > 5 && y != 0);
    System.out.print("Success");
    System.out.println("Failure");
```

A. Success B. Failure C. No Output
D. Arithmetic Exception E. Success Failure

Question 4 - Answer

What are the results of the following?

```
int x = 50;
int y = 100;
if (x / y > 5 && y != 0);
    System.out.print(“Success”);
    System.out.println(“Failure”);
```

A. Success B. Failure C. No Output
D. Arithmetic Exception **E. Success Failure**

Concepts

Branching Structure

(Pay attention to if, if-else, and if-else if structures. Make sure you know whether multiple statements, only one, or none will execute).

Logical Operators

(What do &&, ||, and ! all do?)

Concepts

Short-circuiting

(&& and || will only evaluate as far as they need to – if 1st of && is false, skips second; if 1st of || is true, skips second)

One-line rules

(only the first line is included in the body of an if, else, or loop if brackets are neglected)

Concepts

Empty Statements – “Do Nothing”

(; is an empty statement, be careful!)

On that note:

Is it possible to have the word “return” used in a method whose return type is void?

(Yes, you just need to return an empty statement)

Question 5

What is the output of the following?

```
String s;   //class scope
if(s.equals(“”)){ //in method
        System.out.print(“true”);
}else{
        System.out.println(“false”);
}
```

A. true B. false C. No output
D. an exception occurs E. true false

Question 5 - Answer

What is the output of the following?

```
String s;   // class scope
if(s.equals(“”)){  //in method
        System.out.print(“true”);
}else{
        System.out.println(“false”);
}
```

A. true B. false C. No output
D. an exception occurs E. true false

Question 6

What is the output of the following?

```
String s;  //class scope
if(s.equals(null)){
        System.out.print(“true”);
}else{
        System.out.println(“false”);
}
```

A. true B. false C. No output
D. an exception occurs E. true false

Question 6 - Answer

What is the output of the following?

```
String s;  //class scope
if(s.equals(null)){
        System.out.print(“true”);
}else{
        System.out.println(“false”);
}
```

A. true B. false C. No output
D. an exception occurs E. true false

Concepts

How do we check if something is null?

(using the == operator)

What is the difference between null and an empty string?

(null means no object; an empty string is an object, but with no characters)

Concepts

Where are variables accessible?

(from the line on which they are declared to the end of that block of code)

What is a block of code?

(typically any thing between two curly braces – one line rule is the only exception, but insert the braces where they would be to see the "block")

Concepts

Which variables are automatically assigned?

(Class variables only.)

Any such thing as a global variable in Java?

(Technically speaking? NOPE.)

What are the default values for class variables?

(0 for numbers, false for booleans, null for objects)

LECTURE BREAK

TRACE MANIA!

(8 minutes, 5 questions)

ANSWERS

1. B
2. B
3. B
4. C
5. C

Primitives vs. Objects

What are the primitives in Java?

(byte, short, int, long, float, double, boolean, char)

Do both have references?

(No, only objects)

Do both have methods?

(No, only objects)

Primitives vs. Objects

They are actually stored in different “parts” of the environment:

Primitives & reference variables – stack

Actual objects – heap

Primitives vs. Objects

So what?

Objects must be accessed via an address. A reference variable carries the address to the object in the heap.

Moreover…

EVERY variable declaration is putting another variable on the stack of the environment. (This actually has implications for scope)

For example…

```
public class SomeClass{
        private int x;
        public SomeClass(int x, int y){
                System.out.println(x); //which x?
        }
        //other methods not shown
}
```

(the parameter x)

For example…

```
public class SomeClass{
        private int x;
        public SomeClass(int x, int y){
                System.out.println(x);
        }
        //other methods not shown
}
//What if I wanted the member variable x?
```
(either: rename one of them OR use “this”)

Question 6

```
public void fun(int a, int b){
    a += b;
    b += a;
}
```

Find the output.

```
int x = 3, y = 5;
fun(x, y);
System.out.println(x + " " + y);
```

Question 6 - Answer

```
public void fun(int a, int b){
    a += b;
    b += a;
}
```

3 5

Find the output.

```
int x = 3, y = 5;
fun(x, y);
System.out.println(x + " " + y);
```

Question 7

```
public void fun(int a, int b){
    a += b;
    b += a;
}
```

Find the output.

```
int a = 3, b = 5;
fun(a, b);
System.out.println(a + " " + b);
```

Question 7 - Answer

```
public void fun(int a, int b){
    a += b;
    b += a;
}
```

3 5

Find the output.

```
int a = 3, b = 5;
fun(a, b);
System.out.println(a + " " + b);
```

Concepts

Variables are passed by value.

(Parameters get a copy of the value passed to it when the method is called.)

Variable's scope.

(Begins at the declaration, ends at the end of the block; no two variables of the same name may have the same scope; the variable name refers to the "closest" one, if more than one of same name)

Question 8

```
public void accumulate(int[] a, int n){
    while(n < a.length){
        a[n] += a[n-1];
        n++;
    }
}
```

What is the output of the following?

```
int[] a = {1, 2, 3, 4, 5};
int n = 1;
accumulate(a, n);
for(int k = 0; k < a.length; k++)
    System.out.print(a[k] + " ");
System.out.println(n);
```

Question 8 - Answer

```
public void accumulate(int[] a, int n){
    while(n < a.length){
        a[n] += a[n-1];
        n++;
    }
}
```

1 3 6 10 15 1

What is the output of the following?

```
int[] a = {1, 2, 3, 4, 5};
int n = 1;
accumulate(a, n);
for(int k = 0; k < a.length; k++)
    System.out.print(a[k] + " ");
System.out.println(n);
```

Concepts

Object types *can* change after a method call.

(When objects are passed, the address is what is really passed. The variable becomes an alias to the object, which could change the object's state.)

Can ALL types of objects be changed?

(No! Only classes with mutator methods!)

Mutable vs. Immutable Classes

- Mutable:
  Once the object is made, it may change state (the class variables can change in one of the methods)

- Immutable:
  Once the object is made, it may NOT change state (there are no mutator methods)

Question 10

What is the output?

```
String s1 = “Penguins ”;
String s2 = s1;
s2 += “ROCK! ”;
s2.substring(9);
System.out.println(s1 + s2);
```

A. Penguins ROCK!
B. Penguins Penguins ROCK!
C. Penguins ROCK! ROCK!
D. Penguins ROCK! Penguins ROCK!
E. Penguins ROCK! Penguins

Question 10 - Answer

What is the output?

```
String s1 = “Penguins ”;
String s2 = s1;
s2 += “ROCK! ”;
s2.substring(9);
System.out.println(s1 + s2);
```

A. Penguins ROCK!
B. Penguins Penguins ROCK!
C. Penguins ROCK! ROCK!
D. Penguins ROCK! Penguins ROCK!
E. Penguins ROCK! Penguins

Other immutable classes?

String, Integer, Boolean, Double…

//all wrapper classes

What does that really mean?

Original not changed; if need a result, it will be a new object.

Recursion Wrap-Up!

Complete the next 5 problems.

8 minutes.

Recursion Wrap-up

1. D
2. A
3. E
4. C
5. A