What questions do you have from last night’s video?

Day 2

method return values

Which of the following is **not** a valid return type?

A. void
B. int
C. double
D. String
E. parameter

Every method must have a return type.

True

False

If a method has a non-void return type, it must return a value.

True

False

Answer

Slide 3: E

Slide 4: True, void is a return type!

Slide 5: True

1) Write the header for a method named `send` that has one parameter of type String, and does not return a value.

2) Write the header for a method named `average` that has two parameters, both of type int, and returns an int value.

3) Write the header for a method `instructions`, which does not take any arguments and does not return a value.

4) Write the header for a method named `method1` that has a String parameter first and a double parameter second, and returns a String.

Answer

1) public static void send(String m)
2) public static int average(int a, int b)

3) public static void instructions()

4) public static String method1(String a, double b)

The method below prints information to the console. A better version of the method would return a value instead. If someone wants to print, they can call the method in a `print` statement. If they don't want to print, they can assign the return value to a variable.

Make it a more flexible method by modifying the method code so that the method returns the value instead of printing it. Cross out code and replace it with other code!

```
public static void main(String[] args) {

        String message = uselessPrintingMethod(1,2, "a");

        System.out.print(uselessPrintingMethod(2,3, "cookies"));

}
public static void uselessPrintingMethod(int a, int b, String c)

{


               System.out.println(a + b + " all of the " + c);
```

Answer:

```
public static void main(String[] args) {

        String message = uselessPrintingMethod(1,2, “a”);

        System.out.print(uselessPrintingMethod(2,3, "cookies"));

}
public static String uselessPrintingMethod(int a, int b, String c){


        return a + b + “ all of the “ + c;


}
```

Write a method `largestAbsVal` that accepts three integers as parameters and returns the largest of their three absolute values.

Precondition: this method cannot be called with arguments with equal absolute value. Write your own postcondition!

For example, a call of `largestAbsVal(7, -2, -11)` would return 11, and a call of `largestAbsVal(-4, 5, 2)` would return 5.

Which methods from the `Math` class do you need to use?

Answers

```
/*Precondition: method called with three int arguments, arguments
cannot be equal
Postcondition: will return the largest value of the three arguments
*/
public static int largestAbsVal(int num1, int num2, int num3) {

    int larger = Math.max(Math.abs(num1), Math.abs(num2));

    return Math.max(larger, Math.abs(num3));

}
```

Scope

Why can a class contain two different variables with the same name??? **SCOPE!**

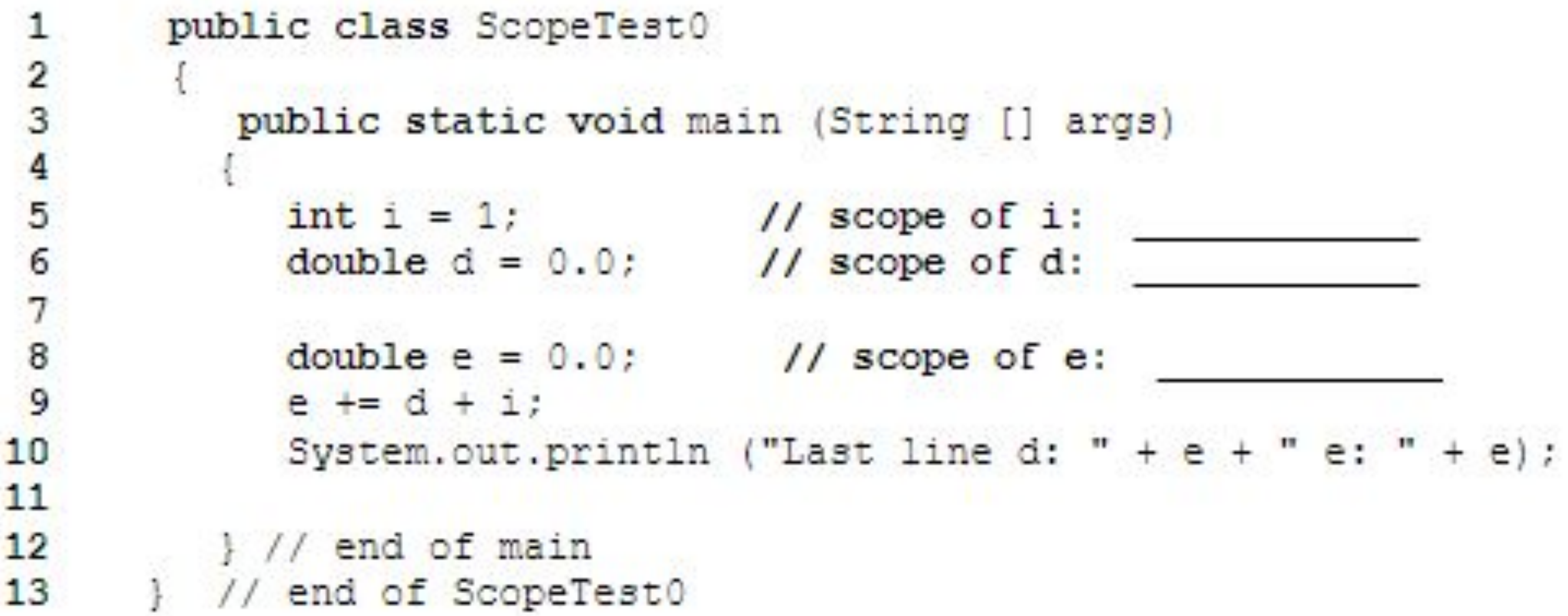
```
public class Swapping
{
    public static void main(String[] args) {
        int  x = 12;
        int  y = 30;
        swap (x, y);
        System.out.println ("X: " + x  + "\nY: " + y);
    }


    public static void swap (int  x, int y)
    {
        int  temp = x;
        x = y;
        y = temp;
    }
}
```

x and y are local to the main method, they cannot be used outside of this block

x and y are local to the swap method, they cannot be used outside of this block

The scopes of x and y in both blocks do not overlap, so both variables can be created in class. The compile will not be confused!

State what the scopes of variables d, e, i, j are:



```
public class ScopeTest0
{
   public static void main (String [] args)
   {
      int i = 1;              // scope of i: ___________
      double d = 0.0;         // scope of d: ___________

      double e = 0.0;          // scope of e: ___________
      e += d + i;
      System.out.println ("Last line d: " + e + " e: " + e);

   } // end of main
}  // end of ScopeTest0
```

```
 1      public class ScopeTest0
 2      {
 3         public static void main (String [] args)
 4         {
 5            int i = 1;             // scope of i: LINES 5-12
 6            double d = 0.0;        // scope of d: LINES 6-12
 7
 8            double e = 0.0;         // scope of e: LINES 8-12
 9            e += d + i;
10            System.out.println ("Last line d: " + e + " e: " + e);
11
12         } // end of main
13     }  // end of ScopeTest0
```

Type the length of the scope (line numbers) for each variable

```
1  public class Demo
2  {
3     public static void main (String [] args)
4     {
5        int x = 27;                    // scope of x ___________
6        methodA (x);
7        int a = 23;                    // scope of a ___________
8        methodA (a);

9     }

10    public void methodA (int  a)  // scope of parameter a __________
11    {
12            System.out.println("Is this the same a as the one declared" +
                      " in the main method?");
13    }


14    public void methodB()
15    {
16       int  x = a + 12;   // cannot access "a" in methodA from
                            // here
                            // scope of x:  __________
17    }
18 }
```

```
1  public class Demo
2  {
3     public static void main (String [] args)
4     {
5        int x = 27;                         // scope of x  MAIN METHOD, LINES 5-9
6        methodA (x);
7        int a = 23;                         // scope of a MAIN METHOD, LINES 7-9
8        method (a);

9     }

10     public void methodA (int  a)  // scope of parameter a LOCAL, LINE 10-13
11     {
12         System.out.println("is this the same a as the one in the main method?");
13     }


14     public void methodB()
15     {
16        int  x = a + 12;   // cannot access "a" in methodA from
                             // here
                             // scope of x:  LOCAL VARIABLE, LINE 16-17
17     }
18  }
```

Write the method `hypotenuse`, which takes two double arguments `side1` and `side2` and returns a double result that represents the length of the hypotenuse of a right triangle with side lengths equal to the length of the arguments. Don’t forget pre and postcondition.

precondition: user will only call method with positive values

Answers

```
/*precondition: called with two doubles representing the lengths of legs in a
triangle, must be > 0
postcondition: will return the length of the hypotenuse of the triangle with given
leg lengths */

public static double hypotenuse (double side1, double side2) {

     //two ways to write code to square a value!

     return Math.sqrt(Math.pow(side1,2) + side2*side2);

}
```

Write a method called `countQuarters` that takes an `int` representing a number of cents as a parameter and returns the number of quarter coins represented by that many cents.

Disregard any whole dollars because those would be dispensed as dollar bills.

For example, `countQuarters(64)` would return `2`, because 2 quarters make 50 cents, with 14 extra left over.

A call of `countQuarters(1278)` would return `3`, because after the 12 dollars are taken out, 3 quarters remain in the 78 cents left.

Answers

```
/* precondition: cents >= 0

postcondition: returns the largest number of quarters in the
non-whole dollar value of cents */

public static int countQuarters(int cents) {

   cents %= 100; //eliminates whole dollars

   return cents/25;

}
```

Write the method `newGrade`. The method returns a `double` representing the student's new average grade after the lowest grade is dropped. The method accepts one argument representing the number of assignments entered. Don't forget pre and postcondition.

BE CAREFUL- trace your code to make sure it returns the correct value and type!!!

In the same class, there are two static methods:

1) `sum()` which asks a teacher to enter the scores and then returns an `int` representing the sum.
2) `min()` which asks a teacher to enter the scores and then returns an `int` representing the lowest score

```
public static double newGrade(int numScores)
```

Answer

```
/*Precondition: method called with int argument representing number
of scores > 1
Postcondition: will return the average of all test scores, dropping
the lowest score*/
public static double newGrade(int numScores) {
    int sum=sum();
    int min=min();
    return (double)(sum-min)/(numScores-1);
}

OR {
    return (sum()-min())/(double)(numScores-1);
    }
```

Write the method `greetingMessage`, which takes two String arguments (`name` and `month`) and two int arguments (`birthday` and `year)` and returns a String. Use the method call and return value below to help you write your method. Don't forget pre and postconditions.

precondition: user will only call method with an appropriate `month` name, `birthday` value between 1 and 31 and appropriate `year` value.

```
greetingMessage("Mia", "March", 24, 2009);
```

returns:
Happy Birthday Mia!
March 24, 2009 is the best day of the year!

Answers

```
/* precondition: user will only call method with an appropriate String representing a month
and a String representing a name, and int birthday value between 1 and 31 and int
representing an appropriate year value.

postcondition: will return a String with birthday greeting */

public static String greetingMessage(String name, String month, int birthday, int year) {

     return “Happy Birthday ” + name + “!\n” + month + “ “ + birthday + “, “ +

     year + “ is the best day of the year!”;

}
```