Practice Final Quiz Name ________________________________ Section ___________

1— If you want to store a letter grade (like a course grade) which variable type would you use?

a. int
b. String
c. char
d. boolean

2— If you wanted to divide two numbers precisely, which of these would be best?

a. int
b. long
c. char
d. float

3— Which of these methods return-types mean the method can return multiple values?

a. void
b. String
c. boolean []
d. Strings

4— Which of the following is used in Java to concatenate (put together) two Strings?

a. `==`
b. `+=`
c. `=`
d. `+`

5— Which of the following represents the proper typical structure of a `if-statement`?

a.
```
if ( arguments );
     statement;
     statement;
     . . .;
}
```
b.
```
if ( expression, statement ) {
     statement;
     statement;
     . . .;
}
```
c.
```
if ( expression ) {
     statement;
     statement;
     . . .;
}
```
d.
```
if ( expression ) then
     statement;
     statement;
     . . .;
```

6— Boolean logic, evaluate the following statements to true or false:

```
cat = true;  bird = false;  dog = true;  fish = false;


cat and dog

bird or fish

( cat or fish ) and ( bird or fish )

( cat ) && ( !dog || fish || cat )

( !cat ) || ( !(!dog) || fish || cat )

( cat || dog || fish ) && dog

( ( !cat ) || ( !fish ) ) && ( dog || fish )
```

7— Given a string, return true if the first instance of "x" in the string is immediately followed by *two* exclamation points.

```
doubleX("ax!bb") → false
doubleX("axax!ax") → false
doubleX("!x!!x!x!x!") → true

     boolean doubleX(String str) {
       boolean temp = false;
       int firstX = str.indexOf( 'x' );

       if(   <BLANK>    ) {
         temp = true;
       }

       return temp;
     }
```

What **if-condition** should be placed where <BLANK> so the method works as intended?

8— Why do the following lines of code not compile? (Explain in one sentence each)

```
int null;
```

```
array [] x = { 8, 9, “ten”, “eleven” };
```

```
int x, y, z;
if( x < y < z ) {
    System.out.println( “less” );
}
```

9— A party is “good” if both tea and candy are at least 5. However, if either tea or candy is at least double the amount of the other one, the party is “great.” However, in all cases, if either tea or candy is less than 5, the party is always “bad.”

```
teaParty(1, 20) → bad
teaParty(12, 6) → great
teaParty(6, 10) → good
```

However! It has **three** errors. Correct them!

```
String teaParty(int tea, int candy) {

  if(  tea < 5  &&  candy < 5  ) {
    retVal = bad;
  }
  else if ( tea*2 <= candy  ||  candy*2 <= tea ) {
    retVal = great;
  }
  else {
    retVal = good;
  }

  return retVal;
}
```

10— Write a method that takes one String and one int. We'll say that the *front* of the String is **the first 3 chars, or whatever is there** if the String is less than length 3.
Return a String which contains n copies of the *front*;

```
"Chocolate" , 2      → "ChoCho"
"Chocolate" , 4      → "ChoChoChoCho"
"Ab" , 3             → "AbAbAb"
```

11— The array question.

Given 2 int arrays, each length 2, return a new array length 4 containing all their elements.

```
plusTwo({1, 2}, {3, 4})      → {1, 2, 3, 4}
plusTwo({4, 4}, {2, 2})      → {4, 4, 2, 2}
plusTwo({9, 2}, {3, 4})      → {9, 2, 3, 4}
```

```
int[] plusTwo(  int[] a, int[] b  ) {















}
```

12—
What does the following code fragment (which compiles) print out:

```
String name;
int counter = 0;
boolean startWord;

name = "John A. Boehner";
startWord = true;
while( counter < name.length() ) {
  if (startWord) {
    System.out.print(name.toLowerCase().charAt(counter));
  } else {
    System.out.print(name.toUpperCase().charAt(counter));
  }
  if (name.charAt(counter) == ' ') {
    startWord = true;
  } else {
    startWord = false;
  }
  counter = counter + 1;
}
```

1— The name of the type of method that creates objects of a class is
a. instantiator
b. creator
c. constructor
d. declarator

2— The following code compiles and runs, so which of the following must be true?
```
apple.isItRipe( friday );
```

a. `friday` must be an int or an enum
b. `apple` must be the name of a class
c. `apple` must be a fruit
d. `isItRipe` must be a method

3— Consider the following code segment, which will be used to time a function. It should print out the number of *seconds* that have elapsed so far each time magicFunction finishes. And then return the total time for all of magicFunction's runs.

Note 1: long's are just like ints but bigger.
Note 2: System.currentTimeMillis() returns a long which is the number of milliseconds since
Jan 1, 1970.

```
long timerMethod( int times ) {
     long time1;
     long time2;


     time1 = System.currentTimeMillis();


     for( int i=0; i<times; i++ ) {
          magicFunction( i );
          System.currentTimeMillis()
          System.out.println( time2 = time2/1000 );

     }


     time1 = System.currentTimeMillis();

     return long = time2 - time1;
}
```

There are errors with this code. Can you fix them? (Don't just say "Yes!" actually fix them)

4— Recursion (Bonus?!)
Write a method that given a base and an exponent (both 1 or more) returns the base to the exponent power. so powerN(3, 2) is $3^2 = 9$. Do not use loops. Do not use Math.pow. Write it recursively!

```
int powerN( int base, int exp ) {
```

5— Write a method that **returns** the nth element of the Fibonacci sequence (n should be an input)

The Fibonacci sequence as defined as:

the first element is 0
the second element is 1
all other elements are the addition of the two previous elements.

That is:	0	1	1	2	3	5	8	13	21	34	55	89	144 ...
	0	1	2	3	4	5	6	7	8	9	10	11	12

That is, if your method was given the input 7, your method should return 13.
(This does not need to be recursive!)

6—
Given the tester code below create an Cat class that will work with the tester code, producing the output below.

```
public static void main(String[] args) {
     Cat myCat = new Cat("fluffy");
     System.out.println( myCat.makeSound() );
     System.out.println( myCat.purr() );
     System.out.println( myCat.introduceSelf() );
     Cat myCat = new Cat("Mittens");
     System.out.println( myCat.introduceSelf() );
}
```

```
Output:
     > meow
     > purr
     > My name is fluffy.
     > My name is Mittens.
```