Programming Final Exam Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. \_\_\_\_\_Every Java statement must end with a
	1. Period (.)
	2. Comma (,)
	3. Semi-Colon (;)
	4. Colon (:)
2. \_\_\_\_\_Which variable type would be the best choice to store a letter grade?
	1. int
	2. double
	3. char
	4. String
3. \_\_\_\_\_Which of the following is an example of an input device?
	1. CPU
	2. Scanner
	3. Eclipse
	4. Speakers
4. \_\_\_\_\_The expression 3 + 2 \* 5 – 1 yields:
	1. 24
	2. 12
	3. 11
	4. 20
5. \_\_\_\_\_Which of the following identifiers is invalid?
	1. FIRSTNAME
	2. number3
	3. gross\_income
	4. 9innings
6. \_\_\_\_\_Which of the following is a valid statement?
	1. String name(“Doe”);
	2. String (“Doe”);
	3. String name= “Doe”;
	4. Name = new String;
7. \_\_\_\_\_What kind of data will this variable store: *char fName*?
	1. True/false value
	2. A single character
	3. Text string
	4. numeric
8. \_\_\_\_\_Which of the following does NOT contain a syntax error?
	1. double w, x, y and z;
	2. System.out.println(Hello again!”);
	3. value = scan.nextint();
	4. num = a \* (b + c);
9. \_\_\_\_\_Which of the following symbols is used to show program input or output?
	1.
	2.
	3.
	4.
10. \_\_\_\_\_Which of the following symbols is used to show program calculation?
	1.
	2.
	3.
	4.
11. \_\_\_\_\_Of the following types, which one cannot store a numeric value?
	1. Double
	2. Int
	3. Char
	4. All of these can store a numeric value
	5. None of these can store a numeric value
12. \_\_\_\_\_Which of the following operators is used to check the equality of two integer values?
	1. =
	2. ==
	3. int.equals()
	4. Int.equals()
13. \_\_\_\_\_What is the output of the following code?

*if (6 > 8)*

*{*

 *System.out.println(“ \*\* “);*

 *System.out.println(“\*\*\*\*”);*

*}*

*else*

 *if (9 == 4)*

 *System.out.println(“\*\*\*\*”);*

 *else*

 *System.out.println(“\*”);*

* 1. \*
	2. \*\*
	3. \*\*\*
	4. \*\*\*\*
1. \_\_\_\_\_Assume the variable declarations below for this problem:

*int x = 3, y = 2, z = x + y;*

Which of the following output statements correctly displays the sentence below?

*3 + 2 = 5*

* 1. System.out.print(x + “+” + y + “=” + z);
	2. System.out.print(x + “ + “ + y + “ = “ + z);
	3. System.out.print(x + y + “ = “ + z);
	4. System.out.print(x + y = z);
1. \_\_\_\_\_Which of the following terms is NOT a Java reserved word?
	1. int
	2. for
	3. private
	4. my
2. \_\_\_\_\_Which of the following is the logical NOT operator?
	1. !
	2. !=
	3. ||
	4. &
3. \_\_\_\_\_Which one of the following declarations is NOT correct?
	1. double duty;
	2. String loan;
	3. boolean value = 12;
	4. int start = 32, end = 99;
4. \_\_\_\_\_Consider the following code segment:

*int x = 2, y = 0;*

*while (x > y)*

*{*

 *x--;*

 *y++;*

*}*

*System.out.println(x – y);*

What is the output of the *System.out.println* statement?

* 1. -2
	2. -1
	3. 0
	4. 1
1. \_\_\_\_\_Which one of the following expressions is NOT correct?
	1. 5 + 4
	2. (5 \* 4)/2
	3. (-12 + (13 + 7/2)) \*4
	4. -12 + (13 + 7)/2) \* 4
2. \_\_\_\_\_Which of the following is the correct way to invoke (or call) the method length() for an object named str and then store the result in the variable val?
	1. val=str.length();
	2. val=length.str();
	3. val=length().str;
	4. val=length(str);
3. \_\_\_\_\_What does the following code fragment write to the monitor?

*int sum = 14;*

*if (sum < 20)*

 *System.out.print(“Under “);*

*else*

*{*

 *System.out.print(“Over “);*

 *System.out.println(“the limit.”);*

*}*

* 1. Under
	2. Over
	3. Under the limit
	4. Over the limit
1. \_\_\_\_\_What does the following code fragment write to the monitor?

*int sum = 7;*

*if (sum > 20)*

*{*

 *System.out.print(“You win “);*

*}*

*else*

*{*

 *System.out.print(“You lose “);*

*}*

*System.out.println(“the prize.”);*

* 1. You win
	2. You lose
	3. You win the prize.
	4. You lose the prize.
1. What will this line of code produce?

System.out.println(“d:\\”);

* 1. “d:\\”
	2. d:\\
	3. d:\
	4. d:
1. \_\_\_\_\_What does the following code fragment write to the monitor?

*int sum=21;*

*if(sum !=20)*

 *System.out.print(“You win “);*

*else*

 *System.out.print(“You lose “);*

*System.out.println(“the prize.”);*

* 1. You win
	2. You lose
	3. You win the prize.
	4. You lose the prize.

1. \_\_\_\_\_What does the following program output?

*class question*

*{*

 *public static void main (String[] args)*

*{*

 *int unitCost = 8;*

 *int items = 5;*

 *System.out.println(“total cost: “ + (unitCost \* items));*

 *}*

 *}*

* 1. total cost: + 40
	2. total cost: 8 \* 5
	3. total cost: 40
	4. “total cost: “ 40
1. \_\_\_\_\_Which of the following is correct?
	1. An OBJECT is like the blueprint, and we can have many instances of it, which are called CLASSES.
	2. A CLASS is like the blueprint, and we can have many instances of it, which are called OBJECTS.
	3. There is no difference between a CLASS and an OBJECT.
	4. An OBJECT sits in CLASS and learns about programming.
2. \_\_\_\_\_Assume that x, y, and z are all int variables. Consider the following code segment:

*if (x == 0)*

*{*

 *if (y == 1)*

 *z += 2;*

*}*

*else {*

 *z += 4;*

*}*

*System.out.print(z);*

What is printed if x, y, and z are all equal to zero before the code segment runs?

1. 0
2. 1
3. 2
4. 4
5. 6
6. \_\_\_\_\_What is the output produced by the following code?

*int code=1;*

*code++;*

*if (code >0)*

 *System.out.print(“Hello”);*

*Else*

 *System.out.print(“Good-bye”);*

* 1. Hello
	2. Good-bye
	3. Hello Good-bye
	4. NullPointerException
1. \_\_\_\_\_Examine the following code:

*int count = 1;*

*while (count < 5)*

*{*

 *System.out.print(count + “ “ );*

*}*

*System.out.println( );*

What does this code print on the monitor?

* 1. 1 2 3 4
	2. 1 2 3 4 5
	3. 2 3 4
	4. 1 1 1 1 1 1 1 1 1 ….
1. \_\_\_\_\_What must the test be so that the following fragment prints out the integers 5 through and including 15?

*int j;*

*for (j = 5; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ; j ++)*

*{*

 *System.out.print(j + “ “);*

*}*

*System.out.println();*

* 1. j < 15
	2. j < 16
	3. j <= 16
	4. j == 15
1. \_\_\_\_\_What must the change be so that the following fragment prints out the even integers 0 2 4 6 8 10?

*int j;*

*for (j = 0; j<=10; )*

 *System.out.print(j + “ “ );*

*System.out.println();*

* 1. j + 2
	2. j = j+2
	3. j++++
	4. ++j++
1. \_\_\_\_\_What must the initialization be so that the following fragment prints out the integers -3 -2 -1 ?

*int j;*

*for ( ; j < 0; j++ )*

 *System.out.print( j+ “ “);*

*System.out.println();*

* 1. j = 0
	2. j < 0
	3. j = -3
	4. j = -4
1. \_\_\_\_\_Consider the following fragment of code:

int field1 = 100;

int field2 = 200;

field2 = field1;

field1 = field2; // line 4

When the code has executed through the line with the comment “line 4”, what are the values of field1 and field2?

* 1. field1 is 100 and field2 is 200
	2. field1 is 200 and field2 is 100
	3. field1 is 100 and field2 is 100
	4. field1 is 200 and field2 is 200
	5. none of the above
1. Suppose that *takeFinal* is a method of a class named *Student*. The *takeFinal* method takes no arguments (variables). Write an example of calling the method *takeFinal* for an object instance of Student named *aStudent*.
2. Give the declaration for a variable called *amount* of type *double*. The variable should be initialized to 20 in the declaration.
3. What is the value of a in this statement: boolean a = (3 == (1 + 2 \* 11 % 4));
4. 1 + 2 > 4-2 && 12 > 23 evaluates to:
5. The following program is supposed to write out the integers 2, 1, 0, -1. Decide what should go in the blanks.

*class Counter3*

*{*

 *Public static void main (String[] args)*

 *{*

 *int count;*

 *count = ;*

 *while (count >= )*

 *{*

 *System.out.println(count);*

 *count = count - ;*

 *}*

 *}*

*}*

1. **Complete** the following class called *MyClass*. This class consists of a single main method that asks the user to input an integer. The main method tells whether the number is even or odd.

A sample output for your class would be:

 **Enter an integer: 12**

 **12 is an even number**

Finish writing the class.

*import java.util.Scanner;*

Your

Answer

*public class MyClass{*

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

 *Scanner scan = new Scanner(System.in);*

 *System.out.print(“Enter an integer: “);*

 *Int num = input.nextInt();*

 }

}

1. Write the value of the expression

 **x <=y || y > 100**

given that x = 150 and y = 125.

Answer:

1. The following program has several errors in it. Identify at least 5.
2. */\**
3. *List Even Numbers Java Example*
4. *This List Even Numbers Java Example shows how to find and list even*
5. *numbers between 1 and any given number.*
6.
7. **public** **class** ListEvenNumbers {
8.
9. **public** **static** **void** Main(String[] args) {
10.
11. *//define limit*
12. **int** 1stLimit = 50
13.
14. System.out.println("Printing Even numbers between 1 and  + 1stLimit);
15.
16. **for**(i=1; i <= 1stLimit; i++){
17.
18. */ if the number is divisible by 2 then it is even*
19. **if**( i % 2 = 0){
20. system.out.print(i and " ");
21. }
22. }
23.
24. }

Error 1:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error 2:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error 3:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error 4:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Error 5:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. Define any **9** of the following programming terms. You must EXPLAIN, not just give an example!

GUI\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Debugging\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Software\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hardware\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Parameter\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Object\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Variable\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

String\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Boolean\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Double\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Int\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Concatination\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operator\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Operand\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Initialization\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instantiation\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Abstraction\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Encapsulation\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pseudocode\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_