

CS302: Self Check Quiz 2

name: _____

Part I True or False

For these questions, is the statement true or false?

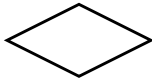
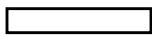

Assume the statements are about the Java programming language.

- 1.) The result of an expression with a single *logical* operator `&&` is a *boolean* value.
- 2.) A syntax error occurs if `true` is used as the condition (i.e. boolean expression) of a `while` loop.
- 3.) *Local variables* only exist while their method executes.
- 4.) Class methods are declared without the *static* modifier.
- 5.) A *compound statement* is a sequence of statements surrounded by parentheses.
- 6.) Variables declared inside a method's body are called *data members*.
- 7.) There is no limit to the number of *levels of nesting* in Java.
- 8.) The result of: `true && !(true && true)`
- 9.) Java programs are terminated if an *overflow error* occurs.
- 10.) *Short-circuit evaluation* only occurs when a boolean variable is assigned a boolean value.
- 11.) A *count-controlled* loop repeats until a *sentinel* value is reached.
- 12.) It is a good programming practice to design an object to handle many tasks.
- 13.) The code: `i = d;` is legal if `i` is a variable of type `int` and `d` is a variable of type `double`.
- 14.) The data type `MainWindow` is a *reference* data type.
- 15.) *Nesting* is when a statement is placed inside the body of another statement.
- 16.) The logical operator `!` is a *binary operator*.
- 17.) It is a good programming practice to cover all possible cases when using a *selection control* statement.
- 18.) If a *then block* or an *else block* is just one statement, braces (i.e. `{ }`) aren't needed.
- 19.) *Pseudocode* is an informal description of code that is useful for describing complex algorithms.
- 20.) A *recursive* method is one that calls itself.
- 21.) `for` loops are *posttest* loops since they have an *increment* part.
- 22.) A *boolean expression* must have either a `true` or `false` result.
- 23.) The code, `new Exam()`, can be used as the argument for a parameter that specifies the type `Exam`.
- 24.) You cannot put a `while` loop inside an `if`'s *then block*.
- 25.) *Arguments* are passed to methods using the *pass-by-value* scheme.

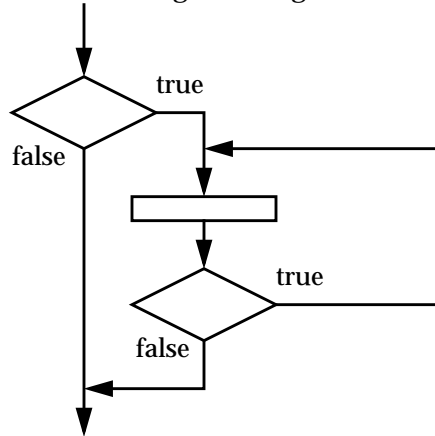
Part II Multiple Choice

For these questions, choose the one best answer after reading all of the choices.

26.) The following symbols are used to indicate the parts of a flow diagram:

symbol	meaning
	a condition (i.e. a boolean expression)
	a single statement or compound statement
	a path in the flow of execution

Consider the following flow diagram:



This flow diagram represents which one of the following?

- A. an if statement with a while loop nested inside
- B. a while loop with an if statement nested inside
- C. an if statement with an if statement nested inside
- D. an if statement with a do-while loop nested inside
- E. a while loop with an do-while loop nested inside

27.) Assume the variables below represent hours and minutes, and are assigned values within the valid ranges:

```
int hour;      // valid range: 0 - 23
int minute;   // valid range: 0 - 59
```

Which one of the following expresses the interval from 7:15 to 9:15?

- A. `hour == 7 || hour == 8 || hour == 9 && minute <= 15`
- B. `hour == 7 && minute >= 15 || hour == 9 && minute <= 15`
- C. `hour >= 7 && minute >= 15 || hour <= 9 && minute <= 15`
- D. `hour == 7 && minute >= 15 || hour == 8 || hour == 9 && minute <= 15`
- E. `hour == 7 || minute >= 15 && hour == 8 && hour == 9 || minute <= 15`

28.) Which one of the following code fragments computes the factorial of 5? The factorial of 5 is computed by multiplying all of the numbers between 1 and 5 inclusive (i.e. including 1 and 5).

- A.

```
int factorial = 1;
for (int n = 5; n >= 0; n--) {
    factorial = factorial * n;
}
```
- B.

```
int factorial = 1;
for (int n = 1; n <= 5; n++) {
    factorial *= factorial;
}
```
- C.

```
int n = 1, factorial = 1;
while (n < 5) {
    factorial = factorial * n;
    n = n + 1;
}
```
- D.

```
int n = 1, factorial = 0;
while (n <= 5) {
    factorial *= n;
    n += 1;
}
```
- E.

```
int n = 5, factorial = 1;
do {
    factorial = factorial * n;
    n = n - 1;
} while (n > 0);
```

29.) Consider the following expression:

```
!(n > 11 && n <= 22)
```

The expression above is equivalent to which one of the following?

- A. $n < 11 \ || \ n \geq 22$
 B. $n \leq 11 \ || \ n > 22$
 C. $n > 11 \ || \ n \leq 22$
 D. $n < 11 \ \&\& \ n \geq 22$
 E. $n \leq 11 \ \&\& \ n > 22$

30.) Assume `n` is an integer variable that has been initialized to some positive number and `odd` is a boolean variable that has been initialized to `true`. Consider the following three code fragments:

fragment 1

```
if (n%2 == 0)
    odd = false;
else
    odd = true;
```

fragment 2

```
if (n%2 == 1)
    odd = true;
```

fragment 3

```
odd = n%2 == 1;
```

Which of the fragments results in `odd` being `true` if `n` is an odd number and otherwise `false`?

- A. fragment 2 only
 B. fragment 1 and fragment 2 only
 C. fragment 1 and fragment 3 only
 D. fragment 2 and fragment 3 only
 E. all three fragments

31.) Consider the following code fragment:

```
switch (i) {
    case 10:
    case 15: d = 2.2; break;
    case 25: d = 7.7; break;
    default: d = 1.1;
}
```

This fragment is equivalent to which one of the following?

- A. `if (i == 10 || i == 15) d = 2.2;`
`if (i == 25) d = 7.7;`
`d = 1.1;`
- B. `if (i >= 10 && i <= 15) d = 2.2;`
`if (i == 25) d = 7.7;`
`d = 1.1;`
- C. `if (i >= 10 && i <= 15) d = 2.2;`
`else if (i == 25) d = 7.7;`
`else d = 1.1;`
- D. `if (i >= 10 || i <= 15) d = 2.2;`
`else if (i == 25) d = 7.7;`
`else d = 1.1;`
- E. `if (i == 10 || i == 15) d = 2.2;`
`else if (i == 25) d = 7.7;`
`else d = 1.1;`

32.) Consider the following boolean expression intended to test if `count` is a prime number between 1 and 10:

```
count == 1 && count == 2 && count == 3 && count == 5 && count == 7
```

Which one of the following best describes this code?

- A. The result is always false, and the `&&`'s should be changed to `||`'s.
- B. The result is always true, and the `&&`'s should be changed to `||`'s.
- C. The result is true if `count` is a prime number between 1 and 10, otherwise the result is false.
- D. A compilation error occurs because variables can't be used more than once in a boolean expression.
- E. A compilation error occurs because each equality test (i.e. `==`) must be surrounded by parentheses.

33.) Consider the following code fragment:

```
for (int j = 1; j <= 10; j++)
    for (int k = 0; k < j; k++)
        System.out.print('*');
```

How many `*` characters are displayed by this code fragment?

- A. 10
- B. 45
- C. 55
- D. 65
- E. 100

34.) Consider the following poorly indented method in a class named `Test`:

```
public int calculate(int x, int y, int z) {
    if (x == 1)
        if (y == 4)
            z = z + 2;
        else
            z = z + 4;
    return z;
}
```

If `testObject` is an instance of the `Test` class, which one of the following values is returned by the message `testObject.calculate(1, 2, 3)`?

- A. 0
- B. 3
- C. 5
- D. 7
- E. none of the above

35.) Consider the following variables that are initialized:

```
int year = 2000;
int month = 11;
int day = 1;
```

Which one of the following boolean expressions does *not* result in short-circuited evaluation?

- A. `year == 1999 || year == 2000 || year == 2001`
- B. `year > 2001 && year < 2020`
- C. `month == 11 && day == 1 || month == 12 && day == 1`
- D. `year == 2000 && month == 12 && day == 22`
- E. `year == 2000 && month == 11 && day == 22`

36.) Consider the following two code fragments:

```
fragment 1
if (a == b) {
    if (c == d) {
        System.out.print("A");
    }
}
else {
    System.out.print("B");
}
```

```
fragment 2
if (a == b && c == d)
    System.out.print("A");
else
    System.out.print("B");
```

Under which of the following conditions will the two fragments produce the same output?

- i.* Only `a` and `b` are the same values.
 - ii.* Only `c` and `d` are the same values.
 - iii.* `a` and `b` are the same values, and `c` and `d` are the same values.
- A. *i* only
 - B. *iii* only
 - C. *i* and *iii* only
 - D. *ii* and *iii* only
 - E. *i*, *ii*, and *iii*

Part III Written Answers

- 1.) Show the output for the code fragment below in the box provided. Show a trace of your execution for partial credit.

```
// out is a properly created OutputBox object
int n = 3;

while (n <= 7) {
    for (int c = n; c > 1; c -- 1) {
        if (n%2 == 0)
            out.print("*");
        else
            out.print("-");
    }
    out.skipLine(1);
    n = n + 1;
}
```

Show your output here.																			
One character per box. Use only as many boxes as needed.																			

- 2.) Consider the following classes and partial lists of their methods:

- Toy class


```
Toy (double price)           // constructs a toy with the specified price
double getPrice ()          // returns the price of the toy
```
- ToyListIterator class


```
boolean hasMoreToys ( ) // returns true if there are more toys in the list
Toy      nextToy ( )     // returns the next toy in the list
```
- ToyChest class


```
ToyChest ( )                // constructs a toy chest
boolean addToy (Toy toy) // adds toy to the toy chest
ToyListIterator getToyListIterator ( )
                        // returns a ToyListIterator for the toy chest
```

Write a code fragment that creates a ToyChest named, myToys, and add 22 toys, where the first toy has the price 1.01, the second toy has the price 1.02, up to the last toy that has the price 1.22.

Assume that the ToyChest, myToys, above has many more toys added to it. Write a code fragment that displays in the console window the total price of all of the toys in this toy chest.