**Select a best possible answer for each of the following:**

1: Information is passed to a method in D

1. the method name
2. that method’s return
3. the method body
4. the arguments to the method

2: A well-designed method D

1. performs multiple unrelated tasks
2. repeats code found in other methods
3. contains thousands of lines of code
4. performs a single, well-defined task

3: Which of the following can be an argument to a method? B

1. Constants.
2. Variables.
3. Expressions.
4. All of the above.

4: Declaring main as static allows the JVM to invoke main C

1. without knowing the name of the class in which main is declared.
2. by creating an object of the class in which main is declared.
3. without creating an instance of the class in which main is declared.
4. None of the above.

5: Variables should be declared as *fields* only if C.

1. they are local variables
2. they are used only within a method
3. they are required for use in more than one method or their values must be saved between calls to the class’s methods
4. they are arguments

6: Identifiers in Java have \_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_ scopes?

1. method, class.
2. class, block.
3. block, statement.
4. statement, file.

7: Which of the following statements describes *block scope*?

1. It begins at the opening { of the class declaration and terminates at the closing }.
2. It limits label scope to only the method in which it is declared.
3. It begins at the identifier's declaration and ends at the terminating right brace (}).
4. It is valid for one statement only.

8: Overloaded methods always have the same \_\_\_\_\_\_\_\_\_.

1. method name
2. return type
3. number of parameters
4. order of the parameters

9: Which of the following methods are overloaded with respect to one another?

public int max (int a, int b) { … }

public double max (double a, double b) { … }

public int max (int a, int b, int c) { … }

public double max (double a, double b, double c) { … }

1. A and B are overloaded; C and D are overloaded.
2. A and C are overloaded; B and D are overloaded.
3. A, B and C are overloaded.
4. All four methods are overloaded.

10: A Java class can have which of the following methods?

(see if the method declarations are different 🡪 E not different from A because the signature is not different – all have 1 int as parameter)

1. void foo(int a)
2. void foo(int a, int b)
3. void foo(double a)
4. void foo(double a, double b)
5. void foo(int b)
6. All of the above.
7. A, B, D, E.
8. A, B, C, D.
9. A, C, D, E.

11: Method calls cannot be distinguished by \_\_\_\_\_\_\_\_.

1. method name
2. return type
3. parameter lists
4. method signature

12: Arrays are \_\_\_\_\_\_\_\_.

a. variable-length entities

b. fixed-length entities

c. data structures that contain up to 10 related data items

d. used to draw a sequence of lines, or “rays”

13: Which of the following statements about arrays are *true*?

a. An array is a group of variables containing values that all have the same type.

b. Elements are located by index.

c. The length of an array c is determined by the expression c.length();. c.length (not a method)

d. The zeroth element of array c is specified by c[0].

14: Consider the array:

s[0] = 7  
s[1] = 0  
s[2] = -12  
s[3] = 9  
s[4] = 10  
s[5] = 3  
s[6] = 6

The value of s[s[6] - s[5]] is:

a. 0.

b. 3.

c. 9.

d. 0.

15: What do the following statements do?

double[] array;  
array = new double[14];

a. Create a double array containing 13 elements.

b. Create a double array containing 14 elements.

c. Create a double array containing 15 elements.

d. Declare but do not create a double array.

16: Invalid possibilities for array indices include .

a. Positive integers.

b. Negative integers.

c. Zero.

d. None of the above.

17: Which of the following statements is false?

a. When an argument is passed by reference, the called method can access the argument’s value in the caller directly but cannot modify it.

b. All arguments in Java are passed by value.

c. To pass an individual array element to a method, use the indexed name of the array.

d. To pass an object reference to a method, simply specify in the method call the name of the variable that refers to the object.

18: Which of the following class members should usually be **private**?

1. Methods.
2. Constructors.
3. Variables (or fields).
4. All of the above.

19: When must a program *explicitly* use the this reference?

1. Accessing a private variable.
2. Accessing a public variable.
3. Accessing a local variable.
4. Accessing an instance variable that is shadowed by a local variable.

20: A constructor *cannot*:

1. be overloaded. (you can have multiple constructors)
2. initialize variables to their defaults.
3. specify return types or return values.
4. have the same name as the class.

21: Using **public** *set* methods helps provide data integrity if:

1. The instance variables are **public**.
2. The instance variables are **private**.
3. The methods perform validity checking.
4. Both b and c.

22: Which of the following statements is *false*?

1. Each class declaration that begins with the access modifier private must be stored in a file that has the same name as the class and ends with the .java filename extension.

(creates a class file)

b. Every class declaration contains keyword class followed immediately by the class’s name.

c. Class, method and variable names are identifiers.

d. An object has attributes that are implemented as instance variables and carried with it throughout its lifetime.

23: Which of the following statements is *false*?

a. The method’s return type specifies the type of data returned to a method’s caller.

b. Empty parentheses following a method name indicate that the method does not require any parameters to perform its task.

c. When a method that specifies a return type other than void is called and completes its task, the method must return a result to its calling method

d. Classes often provide public methods to allow the class’s clients to *set* or *get* private instance variables; the names of these methods must begin with *set* or *get*. (can call methods anything you want)

24. Explain the difference between abstraction and modularization.

Abstraction is when you ignore the details of the parts and focus on a higher level of a program, while modularization is when you divide a whole into well-defined parts that can be built and examined separately, and that interact in well-defined ways.

Abstraction = focusing on big picture, not focusing on details (don’t know details of how the method is implemented, just use the method)

Modularization = separate program into separate classes with separate methods to organize the code

25. What happens when you run the following code? Explain.

public class Dummy{

private void foo(){

System.out.println("I am a method");

}

}

public class Test{

public static void main(String[] args) throws Exception {

Dummy d = new Dummy(); d.foo();

}}

You will get a compilation error because you can’t call a private method in another class that it is not declared in.