













Score: 0 / 30 (0%)

## Chapter 4 Test

### TRUE/FALSE

-  1. Java methods can return only primitive types (int, double, boolean, etc).
- Answer:** F  
Java methods can also return objects, such as a String
- Points:** 0 / 1
-  2. Formal parameters are those that appear in the method call and actual parameters are those that appear in the method header.
- Answer:** F  
The question has the two definitions reversed. Formal parameters are those that appear in the method header, actual parameters are the parameters in the method call (those being passed to the method)
- Points:** 0 / 1
-  3. All Java classes must contain a main method which is the first method executed when the Java class is called upon.
- Answer:** F  
Only the *driver program* requires a main method. The driver program is the one that is first executed in any Java program (except for Applets), but it may call upon other classes as needed, and these other classes do not need main methods
- Points:** 0 / 1
-  4. Java methods can return more than one item if they are modified with the reserved word continue, as in public continue int foo() { ... }
- Answer:** F  
All Java methods return a single item, whether it is a primitive data type an object, or void. The reserved word continue is used to exit the remainder of a loop and test the condition again
- Points:** 0 / 1
-  5. The following method header definition will result in a syntax error: public void aMethod();
- Answer:** T  
The reason for the syntax error is because it ends with a ";" symbol. It instead needs to be followed by { } with 0 or more instructions inside of the brackets. An abstract method will end with a ";" but this header does not define an abstract method
- Points:** 0 / 1


-  6. A method defined in a class can access the class' instance data without needing to pass them as parameters or declare them as local variables.
- Answer:** T  
The instance data are globally available to all of the class' methods and therefore the methods do not need to receive them as parameters or declare them locally. If variables of the same name as instance data were declared locally inside a method then the instance data would be "hidden" in that method because the references would be to the local variables
- Points:** 0 / 1
-  7. The interface of a class is based on those data instances and methods that are declared public.
- Answer:** T  
The interface is how an outside agent interacts with the object. Interaction is only available through those items declared to be public in the class' definition
- Points:** 0 / 1
-  8. Defining formal parameters requires including each parameters type.
- Answer:** T  
In order for the compiler to check to see if a method call is correct, the compiler needs to know the types for the parameters being passed. Therefore, all formal parameters (those defined in the method header) must include their type. This is one element that makes Java a *Strongly Typed* language
- Points:** 0 / 1
-  9. A class may contain methods but not variable declarations.
- Answer:** F  
A class may contain both methods and variable declarations
- Points:** 0 / 1
-  10. A constructor is a method that gets called automatically whenever an object is created, for example with the new operator.
- Answer:** T  
The constructor is used to initialize an object and it gets called whenever a new object is created from a particular class
- Points:** 0 / 1
-  11. A constructor must have the same name as its class.
- Answer:** T  
A constructor is required to have the same name as the class
- Points:** 0 / 1

 12. A constructor must always return an int.

Answer: F

In fact, a constructor cannot return anything. It has no return value, not even void. When a constructor is written, no return value should be listed


Points: 0 / 1

 13. The body of a method may be empty.

Answer: T

The method body may have 0 or more instructions, so it could have 0. An empty method body is simply { } with no statements in between.


Points: 0 / 1

 14. The return statement must be followed a single variable that contains the value to be returned.

Answer: F

The return statement may be following by any expression whose type is the same as the declared return type in the method header. For example, return x\*y+6; is a valid return statement. The statement return; is also valid for a method that does not return anything (void)


Points: 0 / 1

 15. The number and types of the actual parameters must match the number and types of the formal parameters.

Answer: T

The names of the corresponding actual and formal parameters may be different but the number and types of the parameters must match


Points: 0 / 1

 16. The different versions of an overloaded method are differentiated by their signatures.

Answer: T

A method's signature include the number, types, and order of its parameters. With overloaded methods, the name is the same, but the the methods must differ in their parameters


Points: 0 / 1

 17. If a method takes a double as a parameter, you could pass it an int as the actual parameter.

Answer: T

Since converting from an int to a double is a widening conversion, it is done automatically, so there would be no error.


Points: 0 / 1

 18. A method defined without a return statement will cause a compile error.

Answer: F

. If a method is declared to return void then it doesn't need a return statement. However, if a method is declared to return a type other than void, then it must have a return statement


Points: 0 / 1

 19. The println method on System.out is overloaded.

Answer: T

The println method includes versions that take a String, int, double, char, and boolean

Points: 0 / 1


 20. An object may be made up of other objects.

Answer: T

An aggregate object is an object that has other objects as instance data

Points: 0 / 1

#### MULTIPLE CHOICE


 21. The behavior of an object is defined by the object's

- a. instance data
- b. constructor
- c. visibility modifiers
- d. methods
- e. all of the above

Answer: D

The methods dictate how the object reacts when it is passed messages. Each message is implemented as a method, and the method is the code that executes when the message is passed. The constructor is one of these methods but all of the methods combine dictate the behavior. The visibility modifiers do impact the object's performance indirectly

Points: 0 / 1

 22. Which of the following reserved words in Java is used to create an instance of a class?


- a. class
- b. public
- c. public or private, either could be used
- d. import
- e. new

Answer: E

The reserved word "new" is used to instantiate an object, that is, to create an instance of a class. The statement new is followed by the name of the class. This calls the class' constructor. Example: Car x = new Car(); will create a new instance of a Car and set the variable x to it

Points: 0 / 1




-  23. If a method does not have a return statement, then
- a. it will produce a syntax error when compiled
  - b. it must be a void method
  - c. it can not be called from outside the class that defined the method
  - d. it must be defined to be a public method
  - e. it must be an int, double, or String method

**Answer:** B

All methods are implied to return something and therefore there must be a return statement. However, if the programmer wishes to write a method that does not return anything, and therefore does not need a return statement, then it must be a void method (a method whose header has "void" as its return type)


**Points:** 0 / 1

-  24. Consider a sequence of method invocations as follows: main calls m1, m1 calls m2, m2 calls m3 and then m2 calls m4, m3 calls m5. If m4 has just terminated, what method will resume execution?
- a. m1
  - b. m2
  - c. m3
  - d. m5
  - e. main

**Answer:** B

: Once a method terminates, control resumes with the method that called that method. In this case, m2 calls m4, so that when m4 terminates, m2 is resumed.


**Points:** 0 / 1

-  25. A class' constructor usually defines
- a. how an object is initialized
  - b. how an object is interfaced
  - c. the number of instance data in the class
  - d. the number of methods in the class
  - e. if the instance data are accessible outside of the object directly

**Answer:** A

The constructor should be used to "construct" the object, that is, to set up the initial values of the instance data. This is not essential, but is typically done. The interface of an object is dictated by the visibility modifiers used on the instance data and methods

**Points:** 0 / 1

-  26. Having multiple class methods of the same name where each method has a different number of or type of parameters is known as
- a. encapsulation
  - b. information hiding
  - c. tokenizing
  - d. importing
  - e. method overloading

**Answer:** E

When methods share the same name, they are said to be overloaded. The number and type of parameters passed in the message provides the information by which the proper method is called

**Points:** 0 / 1

 27. For questions 7 use the following class definition

```
import java.text.DecimalFormat;
public class Student
{
    private String name;
    private String major;
    private double gpa;
    private int hours;

    Student(String newName, String newMajor, double newGPA, int newHours)
    {
        name = newName;
        major = newMajor;
        gpa = newGPA;
        hours = newHours;
    }

    public String toString()
    {
        DecimalFormat df = new DecimalFormat("xxxx");    // xxxx needs to be replaced
        return name + "\n" + major + "\n" + df.format(gpa) + "\n" + hours
    }
}
```


Which of the following could be used to instantiate a new Student s1?

- a. Student s1 = new Student();
- b. s1 = new Student();
- c. Student s1 = new Student("Jane Doe",  
"Computer Science", 3.333, 33);
- d. new Student s1 = ("Jane Doe",  
"Computer Science", 3.333, 33);
- e. new Student(s1);

**Answer:** C

To instantiate a class, the object is assigned the value returned by calling the constructor preceded by the reserved word new, as in new Student(). The constructor might require parameters, and for Student, the parameters must be are two String values, a double, followed by an int

**Points:** 0 / 1

 28. For questions 8 use the following class definition

```
import java.text.DecimalFormat;
public class Student
{
    private String name;
    private String major;
    private double gpa;
    private int hours;

    Student(String newName, String newMajor, double newGPA, int newHours)
    {
        name = newName;
        major = newMajor;
        gpa = newGPA;
        hours = newHours;
    }

    public String toString()
    {
        DecimalFormat df = new DecimalFormat("xxxx");    // xxxx needs to be replaced
        return name + "\n" + major + "\n" + df.format(gpa) + "\n" + hours
    }
}
```

Assume that another method has been defined that will compute and return the student's class rank (Freshman, Sophomore, etc). It is defined as:

```
public String getClassRank()
```


Given that s1 is a student, which of the following would properly be used to get s1's class rank?

- a. s1 = getClassRank();
- b. s1.toString();
- c. s1.getHours();
- d. s1.getClassRank();
- e. getClassRank(s1);

**Answer:** D

To call a method of an object requires passing that object a message which is the same as the method name, as in object.methodname(parameters). In this situation, the object is s1, the method is getClassRank, and this method expects no parameters. Answers a and e are syntactically illegal while answer b returns information about the Student but not his/her class rank, and there is no "getHours" method so c is also syntactically illegal.

**Points:** 0 / 1

 29. What does the following code compute?


```
int num = 0;
for(int j = 0; j < 1000; j++)
{
    c.flip();
    if(c.isHeads()) num++;
}
double value = (double) num / 1000;
```

- a. the number of Heads flipped out of 1000 flips
- b. the number of Heads flipped in a row out of 1000 flips
- c. the percentage of heads flipped out of 1000 flips
- d. the percentage of times neither Heads nor Tails were flipped out of 1000 flips
- e. nothing at all

**Answer:** C

The code iterates 1000 times, flipping the Coin and testing to see if this flip was a 0 ("Heads") or 1 ("Tails"). The variable num counts the number of Heads and the variable value is then the percentage of Heads over 1000

**Points:** 0 / 1

 30. Consider a method defined with the header: public void foo(int a, int b). Which of the following method calls is legal?

- a. foo(0, 0.1);
- b. foo(0 / 1, 2 \* 3);
- c. foo(0);
- d. foo();
- e. foo(1 + 2, 3 \* 0.1);

**Answer:** B

The only legal method call is one that passes two int parameters. In the case of answer b, 0 / 1 is an int division (equal to 0) and 2 \* 3 is an int multiplication. So this is legal. The answers for a and e contain two parameters, but the second of each is a double. The answers for c and d have the wrong number of parameters

**Points:** 0 / 1