

## AP MidTerm Test

### MULTIPLE CHOICE

1. A cast is required in which of the following situations?
- using charAt to take an element of a String and store it in a char
  - storing an int in a double
  - storing a double in a double
  - storing a double in an int
  - all of the above require cast

**Answer:** D

For a, charAt returns a char, so there is no problem. In b, the situation is a widening operation taking a narrower type and storing the value in a wider type. Only in d is there a situation where a wider type is being stored in a narrower type, so a cast is required.

**Points:** 0 / 1

2. Which statement would we use to create an object from a class called **Thing**?
- Thing something;
  - Thing something = Thing();
  - Thing something = new Thing;
  - Thing something = new Thing();
  - new thing() = something;

**Answer:** D

**Points:** 0 / 1

3. Suppose we have a variable named **something** that is a reference to a **Thing** object. How would we call the method **dolt** on our **Thing** object?
- dolt()
  - something.dolt()
  - dolt(something)
  - something/dolt
  - something(dolt)

**Answer:** B

**Points:** 0 / 1

4. Which of the sets of statements below will add 1 to x if x is positive and subtract 1 from x if x is negative but leave x alone if x is 0?

- |  |  |
|--|--|
| a. if (x > 0) x++;<br>else x--;                      | d. if (x == 0) x = 0;<br>else x++;<br>x--; |
| b. if (x > 0) x++;<br>else if (x < 0) x--;           | e. x++;<br>x--;                            |
| c. if (x > 0) x++;<br>if (x < 0) x--;<br>else x = 0; |  |

**Answer:** B

if x is positive, x++ is performed else if x is negative x-- is performed and otherwise, nothing happens, or x is unaffected. In a, c, d and e, the logic is incorrect. In a, x-- is done if x is not positive, thus if x is 0, x becomes -1 which is the wrong answer. In c, if x is positive, then x++ is performed. In either case, the next statement is executed and if x is not negative, the else clause is performed setting x to 0. So if x is positive, it becomes 0 after this set of code. In d, x++ and x-- are both performed if x is not 0. And in e, this code does not attempt to determine if x is positive or negative, it just adds one and then subtracts 1 from x leaving x the same

**Points:** 0 / 1

- 5.

Given the nested if-else structure below, answer question 5.

```
if (a > 0)
    if (b < 0)
        x = x + 5;
    else
        if (a > 5)
            x = x + 4;
        else
            x = x + 3;
    else
        x = x + 2;
```

If x is currently 0, a = 5 and b = 5, what will x become after the above statement is executed?

- |      |      |
|------|------|
| a. 0 | d. 4 |
| b. 2 | e. 5 |
| c. 3 |      |

**Answer:** C

: Since (a > 0) is true, the next condition is checked. Since (b < 0) is false, the else clause for this condition is executed. Since (a > 5) is false the else clause for this condition is executed, which is x = x + 3. Therefore, 3 is added to x, so it is now 3.

**Points:** 0 / 1

6. Assume that count is 0, total is 20 and max is 1. The following statement will do which of the following?

```
if (count != 0 && total / count > max) max = total / count;
```

- a. The condition short circuits and the assignment statement is not executed
- b. The condition short circuits and the assignment statement is executed without problem
- c. The condition does not short circuit causing a division by zero error
- d. The condition short circuits so that there is no division by zero error when evaluating the condition, but the assignment statement causes a division by zero error
- e. The condition will not compile because it uses improper syntax

**Answer:** A

Since count is 0, (count != 0) is false. Because the left-hand side of an && condition is false, the condition is short circuited, and so the right-hand side is not evaluated. Thus, a potential division by zero error is avoided. Because the condition is false, the statement max = total / count is not executed, again avoiding a potential division by zero error.

**Points:** 0 / 1

7. 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 a

**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

8. 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

9. 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

10. Instance data for a Java class

- a. are limited to primitive types (e.g., int, double, char)
- b. are limited to Strings
- c. are limited to objects(e.g., Strings, classes defined by other programmers)
- d. may be primitive types or objects, but objects must be defined to be private
- e. may be primitive types or object

**Answer:** E

The instance data are the entities that make up the class and may be any type available whether primitive or object, and may be public or private. By using objects as instance data, it permits the class to be built upon other classes. This relationship where a class has instance data that are other classes is known as a has-a relationship

**Points:** 0 / 1



11. In order to implement Comparable in a class, what method(s) must be defined in that class?
- a. equals
  - b. compares
  - c. both lessThan and greaterThan
  - d. compareTo
  - e. both compares and equals

Answer: D

The Comparable class requires the definition of a compareTo method that will compare two objects and determine if one is equal to the other, or if one is less than or greater than the other and respond with a negative int, 0 or a positive int. Since compareTo responds with 0 if the two objects are equal, there is no need to also define an equals method.

Points: 0 / 1

12. If s is a String, and s = "no"; is performed, then s

- a. stores the String "no"
- b. references the memory location where "no" is stored
- c. stores the characters 'n', 'o'
- d. stores an int value that represents the two characters
- e. stores the character 'n' and a reference to the memory location where the next character, 'o' is stored

Answer: B

Strings are objects and all objects in Java are referenced by the variable declared to be an object. That is, the variable represents the memory location where the object is stored. So, s does not directly store "no" or 'n', 'o', but instead stores a memory location where "no" is stored

Points: 0 / 1

13. An object that refers to part of itself within its own methods can use which of the following reserved words to denote this relationship?

- a. inner
- b. i
- c. private
- d. this
- e. static

Answer: D

The reserved word *this* is used so that an object can refer to itself. For instance, if an object has an instance data x, then *this.x* refers to the object's value x. While *this* is not necessary, it can be useful if a local variable or parameter is named the same as an instance data. The reserved word *this* is also used to refer to the class as a whole, for instance, if the class is going to implement an interface class rather than import an implementation of an interface class

Points: 0 / 1

14. Static methods can not
- a. reference instance data
  - b. reference non-static instance data
  - c. reference other objects
  - d. invoke other static methods
  - e. invoke non-static methods

Answer: B

A static method is a method that is part of the class itself, not an instantiated object, and therefore the static method is shared among all instantiated objects of the class. Since the static method is shared, it cannot access non-static instance data because all non-static instance data are specific to instantiated objects. A static method can access static instance data because, like the method, the instance data is shared among all objects of the class. A static method can also access parameters passed to it

Points: 0 / 1

#### TRUE/FALSE

15. In Java, 'a' and 'A' are considered to be different values.

Answer: T

Characters (char) values are stored using Unicode, and 'a' and 'A' have different Unicode values.

Points: 0 / 1

16. If a, b and c are int variables with a = 5, b = 7, c = 12, then the statement `int z = (a * b - c) / a;` will result in z equal to 4.

Answer: T

$(a * b - c) / a = (5 * 7 - 12) / 5 = (35 - 12) / 5 = 23 / 5$ , and since 23 and 5 are int values, the division is performed as an int division, or  $23 / 5 = 4$ .

Points: 0 / 1

17. Since double to int is a narrowing conversion, there is no way to convert a double to an int.

Answer: F

Going from double to int is a narrowing conversion, but it is possible to do using a cast






Points: 0 / 1







18. The statements `x++`; and `++x`; will accomplish the same thing.

Answer: T

The statements `x++`; and `++x`; both increment x

Points: 0 / 1

-  19. The following for-loop is an infinite loop.
- ```
for(int j = 0; j < 1000; ) i++;
```
- Answer:** T  
This loop initializes j to 0 and compares it to 1000, but does not alter j after each loop iteration. In reality, the loop will terminate with a run-time error eventually once i becomes too large of a value to store
- Points:** 0 / 1
-  20. For questions 6-7, assume that boolean done = false, int x = 10, int y = 11, String s = "Help" and String t = "Goodbye".
- The expression (!done && x <= y) is true
- Answer:** T  
Since done is false, !done is true. Since 10 < 11, x <= y is true. Therefore, the entire expression is true.
- Points:** 0 / 1
-  21. For questions 6-7, assume that boolean done = false, int x = 10, int y = 11, String s = "Help" and String t = "Goodbye".
- The expression (s.concat(t).length() < y) is true
- Answer:** F  
Concatenating s and t gives a String that is 11 characters long and 11 < 11 is false
- Points:** 0 / 1
-  22. 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
-  23. 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

-  24. A class may contain methods but not variable declarations.
- Answer:** F  
A class may contain both methods and variable declarations
- Points:** 0 / 1
-  25. 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
-  26. The == operator performs differently depending on whether the variables being compared are primitives types or objects
- Answer:** T  
If two objects are being compared using ==, then the comparison returns true if the two objects are aliases (the same object). But if two primitive data types are being compared, then the comparison returns true if the two variables store the same value
- Points:** 0 / 1
-  27. Assume that the class Bird has a static method fly(). If b is a Bird, then to invoke fly, you could do Bird.fly();.
- Answer:** T  
Static methods are invoked through the class and not an object of the class. This is because the static method is shared among all instances. So, Bird.fly(); will invoke fly. It should be noted that fly can also be invoked through b, so b.fly(); will also work.
- Points:** 0 / 1
-  28. An object uses the "this" reserved word to refer to itself.
- Answer:** T  
The "this" reserved word refers to the currently executing object
- Points:** 0 / 1
-  29. When designing a class to represent an object, you need to think about the object's state and behavior.
- Answer:** T  
Both state (represented by instance variables) and behavior (represented by methods) should be considered when designing a class.
- Points:** 0 / 1