user:Ahsan Ahsan Habib - 1144421

test:**R-18 Core Java mock test 3**  
R-18 Core Java mock test 3

start time:2013-08-24 03:31:30

end time:2013-08-24 04:21:29

test time:00:49:59

points:30.767 / 45.000 (68%)

correct:31 / 45 (69%)

comment:

1. **[1.000] (IP:281473913979145 | 03:31:30 | 03:34:34 | 03:04 | 184.294)**   
   Examine the following program fragment:  
     
     
   int[] array = { 1, 4, 3, 6, 8, 2, 5};  
   int what = array[0];  
     
   // scan the array  
   for ( int index=0; index < array.length; index++ )  
   {   
   if ( array[ index ] > what )   
   what = array[ index ];   
   }  
   System.out.println( what );   
     
   What does the fragment write to the monitor?
   1. 1
   2. 1 4 3 6 8 2 5
   3. x ® 8
   4. 5
2. **[1.000] (IP:281473913979145 | 03:34:34 | 03:35:39 | 01:05 | 56.223)**   
   What is the output of the following code fragment:  
     
   int[] zip = new int[5];  
     
   zip[0] = 7;  
   zip[1] = 3;  
   zip[2] = 4;  
   zip[3] = 1;  
   zip[4] = 9;  
     
   int j = 3;  
     
   System.out.println( zip[ j-1 ] );
   1. x ® 4
   2. 7
   3. 3
   4. 1
3. **[1.000] (IP:281473913979145 | 03:35:39 | 03:36:49 | 01:10 | 69.342)**   
   The no-argument constructor provided by the compiler when no constructor is explicitly provided in the code
   1. is always "friendly"
   2. is always public
   3. depends on the compilation options of javac
   4. x ® always defaults to the access modifier provided for the class.
4. **[1.000] (IP:281473913979145 | 03:36:49 | 03:37:48 | 00:59 | 58.836)**   
   For which of the following applications is an array NOT suitable:
   1. Holding the scores on twelve midterms exams of a class.
   2. x ® Holding the name, social security number, age, and income of one individual.
   3. Holding the temperature readings taken every hour throughout a day.
   4. Holding the total sales a store made in each of twelve months.
5. **[1.000] (IP:281473913979145 | 03:38:54 | 03:40:02 | 01:08 | 68.134)**   
   What does the following statement do?  
   int[] values = new int[10] ;
   1. x ® It declares values to be a reference to an array object and constructs an array object containing 10 integers which are initialized to zero.
   2. It declares values to be a reference to an array object which does not yet exist,   
      but will contain 10 zeros when it does
   3. It declares values to be a reference to an array object, but initializes it to null
   4. It declares values to be a reference to an array which contains 10 references to int variables.
6. **[1.000] (IP:281473913979145 | 03:46:05 | 03:46:39 | 00:34 | 34.573)**   
   What does the following statement do? String glarch;
   1. It declares an array of String objects named glarch.
   2. It constructs a String object named glarch.
   3. x ® It declares a reference variable glarch which is initialized to null.
   4. It constructs a String object which will contain the characters "glarch" .
7. **[1.000] (IP:281473913979145 | 03:46:39 | 03:47:25 | 00:46 | 45.912)**   
   What does the following statement do?  
     
   int[] values = new int[10] ;
   1. x ® It declares values to be a reference to an array object and constructs an array object containing 10 integers which are initialized to zero.
   2. It declares values to be a reference to an array object, but initializes it to null.
   3. It declares values to be a reference to an array which contains 10 references to int variables.
   4. It declares values to be a reference to an array object which does not yet exist,  
      but will contain 10 zeros when it does.
8. **[1.000] (IP:281473913979145 | 03:51:41 | 03:52:52 | 01:11 | 70.384)**   
   Examine the following:  
     
   double[][] values =  
   { {1.2, 9.0, 3.2},  
   {9.2, 0.5, 1.5, -1.2},  
   {7.3, 7.9, 4.8} } ;  
     
   what is in values[2][1] ?
   1. 9.2
   2. There is no such array element.
   3. 7.3
   4. x ® 7.9
9. **[1.000] (IP:281473913979145 | 03:54:40 | 03:56:11 | 01:31 | 90.95)**   
   Say that names has been declared  
   String[] names = new String[10] ;   
   and that further statements (not shown) have put String references into some of the slots.   
   Which of the following fragments prints out every String, but skips null references?
   1. for ( int j = 0; names[j] != null; j++ )  
      System.out.println( names[j] );
   2. for ( int j = 0; j < names.length && names[j] != null ; j++ )  
      System.out.println( names[j] );
   3. x ® for ( int j = 0; j < names.length; j++ )  
      if ( names[j] != null )  
      System.out.println( names[j] );
   4. for ( int j = 0; j < names.length; j++ )  
      System.out.println( names[j] );
10. **[1.000] (IP:281473913979145 | 03:56:11 | 03:57:04 | 00:53 | 52.283)**   
    What is the output of the following code fragment:  
      
    int[] egArray = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9 };  
      
    for ( int index= 0 ; index < 5 ; index++ )  
    System.out.print( egArray[ index ] + " " );
    1. 2 4 6 8 10 1 3 5 7 9
    2. 2 4 6 8
    3. 2 4 6 8 10 1
    4. x ® 2 4 6 8 10
11. **[1.000] (IP:281473913979145 | 04:00:56 | 04:02:05 | 01:09 | 69.709)**   
    Given the declaration  
    String[] names = new String[10] ;   
    Which of the following statements puts a reference to the String "Hello" in the last slot of the array?
    1. names[10] = "Hello" ;
    2. names[0] = "Hello" ;
    3. String[ names.length-1 ] = "Hello" ;
    4. x ® names[9] = "Hello" ;
12. **[1.000] (IP:281473913979145 | 04:03:39 | 04:05:01 | 01:22 | 82.536)**   
    Given the following:  
      
    long[][] stuff ;  
      
    Which of the following statements constructs an array with 5 rows of 7 columns each and assign its reference to stuff ?
    1. stuff = long[7][5] ;
    2. stuff = long[5][7] ;
    3. stuff = new stuff[5][7] ;
    4. x ® stuff = new long[5][7] ;
13. **[1.000] (IP:281473913979145 | 04:05:02 | 04:05:20 | 00:18 | 18.569)**   
    What is the output of the following code fragment:  
      
    int[] zip = new int[5];  
      
    zip[0] = 7;  
    zip[1] = 3;  
    zip[2] = 4;  
    zip[3] = 1;  
    zip[4] = 9;  
      
    System.out.println( zip[ 2 + 1 ] );
    1. 4
    2. 3 7
    3. 4 3
    4. x ® 1
14. **[1.000] (IP:281473913979145 | 04:06:10 | 04:06:55 | 00:45 | 44.953)**   
    You want to limit access to a method of a public class to members of the same class.  
    Which access accomplishes this objective?
    1. protected
    2. default access
    3. transient
    4. x ® private
    5. public
15. **[1.000] (IP:281473913979145 | 04:06:56 | 04:07:14 | 00:18 | 18.015)**   
    Given the following:  
      
    double[][] things =  
    { {1.2, 9.0},  
    {9.2, 0.5, 0.0},  
    {7.3, 7.9, 1.2, 3.9} } ;  
      
    What is the value of things.length ?
    1. 9
    2. 2
    3. x ® 3
    4. 4
16. **[1.000] (IP:281473913979145 | 04:07:14 | 04:07:40 | 00:26 | 10.443)**   
    Given the following:  
      
    double[][] things =  
    { {1.2, 9.0},  
    {9.2, 0.5, 0.0},  
    {7.3, 7.9, 1.2, 3.9} } ;  
      
    What is the value of things[2].length ?
    1. x ® 4
    2. 9
    3. 3
    4. 2
17. **[1.000] (IP:281473913979145 | 04:08:19 | 04:08:58 | 00:39 | 39.047)**   
    Given the following:  
      
    int[][] items =  
    { {0, 1, 3, 4},  
    {4, 3, 99, 0, 7 },  
    {3, 2} } ;  
      
    Which of the following statements replaces the 99 with 77?
    1. items[2][1] = 77;
    2. items[ 99 ] = 77;
    3. items[2][3] = 77;
    4. x ® items[1][2] = 77;
18. **[1.000] (IP:281473913979145 | 04:09:28 | 04:09:48 | 00:20 | 19.33)**   
    What are the legal indexes for the array ar, given the following declaration:  
      
    int[] ar = {2, 4, 6, 8 }
    1. 2, 4, 6, 8
    2. x ® 0, 1, 2, 3
    3. 0, 2, 4. 6
    4. 1, 2, 3, 4
19. **[1.000] (IP:281473913979145 | 04:11:18 | 04:11:56 | 00:38 | 37.472)**   
    Which of the following declares an array of int named img?
    1. x ® int[] img;
    2. int img = int[];
    3. int img;
    4. new int img[];
20. **[1.000] (IP:281473913979145 | 04:12:33 | 04:14:14 | 01:41 | 100.701)**   
    Say that names has been declared  
    String[] names = new String[10] ;   
    and that further statements (not shown) have put String references into some of the slots.   
    Which of the following fragments prints out the slots of the array from last to first, skipping slots that contain null?
    1. for ( int j = names.length; j < names.length; j++ )  
       if ( names[j] != null )  
       System.out.println( names[j] );
    2. for ( int j = names.length; j >= 0; j++ )  
       if ( names[j] != null )  
       System.out.println( names[j] );
    3. x ® for ( int j = names.length-1; j >= 0; j-- )  
       if ( names[j] != null )  
       System.out.println( names[j] );
    4. for ( int j = 0; j < names.length; j++ )  
       if ( names[j] != null )  
       System.out.println( names[j] );
21. **[1.000] (IP:281473913979145 | 04:14:14 | 04:15:40 | 01:26 | 86.001)**   
    What is the output of the following code fragment:  
      
    int[] egArray = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9 };  
      
    for ( int index= 0 ; index < egArray.length ; index = index + 2 )  
    System.out.print( egArray[ index ] + " " );
    1. 2 4 6 8 10 1 3 5 7 9
    2. x ® 2 6 10 3 7
    3. 2 6 10 3 7 0
    4. 4 8 1 5 9
22. **[1.000] (IP:281473913979145 | 04:15:40 | 04:16:35 | 00:55 | 54.952)**   
    You want to create a table that looks like:  
      
    12 -9 8  
    7 14  
    -32 -1 0  
      
    Which of the following will work?
    1. x ® double[][] table =  
       { {12, -9, 8},  
       {7, 14},  
       {-32, -1, 0} };
    2. double[][] table =  
       { 12, -9, 8,   
       7, 14,  
       -32, -1, 0} ;
    3. double[][] table =  
       { {12, -9, 8},   
       {7, 14, 0},  
       -32, -1, 0} };
    4. double[][] table =  
       { {12, -9, 8}  
       {7, 14}  
       {-32, -1, 0} };
23. **[1.000] (IP:281473913979145 | 04:16:35 | 04:17:13 | 00:38 | 37.845)**   
    Given:  
    1. public class Test {  
    2. private static float[] f = new float[2];  
    3. public static void main(String args[]) {  
    4. System.out.println(“f[0] = “ + f[0]);  
    5. }  
    6. }  
    What is the result?
    1. x ® f[0] = 0.0
    2. An exception is thrown at runtime.
    3. Compilation fails.
    4. f[0] = 0
24. **[1.000] (IP:281473913979145 | 04:18:00 | 04:18:22 | 00:22 | 21.657)**   
    What is the output of the following code fragment:  
      
    int[] ar = {2, 4, 6, 8 };  
    System.out.println( ar[0] + " " + ar[1] );
    1. 8
    2. 2 6
    3. x ® 2 4
    4. 6 8
25. **[1.000] (IP:281473913979145 | 04:18:45 | 04:19:20 | 00:35 | 35.031)**   
    Say that names has been declared  
      
    String[] names = new String[10] ;   
      
    and that further statements (not shown) have put String references into some of the slots.  
      
    Which of the following fragments prints out every String, but skips null references?
    1. x ® for ( int j = 0; j < names.length; j++ )  
       if ( names[j] != null )  
       System.out.println( names[j] );
    2. for ( int j = 0; names[j] != null; j++ )  
       System.out.println( names[j] );
    3. for ( int j = 0; j < names.length; j++ )  
       System.out.println( names[j] );
    4. for ( int j = 0; j < names.length && names[j] != null ; j++ )  
       System.out.println( names[j] );
26. **[1.000] (IP:281473913979145 | 04:19:20 | 04:19:47 | 00:27 | 26.842)**   
    What is the base type of the array given below?   
    Color[] example = new Color[20];
    1. x ® Color
    2. No base type
    3. Both Color and example
    4. example
27. **[1.000] (IP:281473913979145 | 04:19:47 | 04:20:56 | 01:09 | 68.681)**   
    Say that names has been declared  
      
    String[] names = new String[10] ;   
      
    and that further statements (not shown) have put String references into some of the slots.  
      
    Which of the following fragments counts the number of non-null slots in the array?
    1. int count = 0;  
       while ( names[ count ] != null )  
       {  
       count++ ;  
       }
    2. x ® int count = 0;  
       for ( int j = 0; j < names.length; j++ )  
       if ( names[j] != null )  
       count++ ;
    3. int j = 0;  
       for ( int count = 0; count < names.length; count++ )  
       if ( names[j] != null )  
       j++ ;
    4. int j = 0;  
       int count = 0;  
       while ( names[ ++j ] != null )  
       count++ ;

**topics**

* 30.767 / 45 (68%) 31 / 45 (69%) **Core Java**
  + 20 / 27 (74%) 20 / 27 (74%) Array (new) 4-1
  + 1 / 1 (100%) 1 / 1 (100%) Constructor 4-1
  + 0.667 / 1 (67%) 1 / 1 (100%) Array 6-2
  + 5 / 7 (71%) 5 / 7 (71%) Array 4-1
  + 0 / 2 (0%) 0 / 2 (0%) General2 4-1
  + 0.5 / 1 (50%) 0 / 1 (0%) Access Modifier 6-3
  + 0 / 1 (0%) 0 / 1 (0%) Access Modifier 5-1
  + 1 / 2 (50%) 1 / 2 (50%) General2 5-1
  + 0.6 / 1 (60%) 1 / 1 (100%) Constructor 5-3
  + 1 / 1 (100%) 1 / 1 (100%) Array 6-3