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. **[0.667] (IP:281473913979145 | 03:37:48 | 03:38:54 | 01:06 | 66.016)**   
   Which two cause a compiler error? (Choose two)
   1. float[] f1 = new float[3];
   2. ® float f2[] = new float[];
   3. float f3[] = new float[3];
   4. x ® float[] = new float(3);
   5. float f5[] = { 1.0f, 2.0f, 2.0f };
   6. x   float f4[] = new float[] { 1.0f. 2.0f. 3.0f};
6. **[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.
7. **[0.000] (IP:281473913979145 | 03:40:02 | 03:43:24 | 03:22 | 201.884)**   
   Fill in the blank in the following code fragment so that each element of the array is assigned twice the value of its index.  
     
     
   int[] array = new int[10];  
     
   // scan the array  
   for ( int index=0; index < array.length; index++ )  
   {   
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
   }
   1. x   array[ index ] = 2\*array[ index ];
   2. index = 2\*index;
   3. ® array[ index ] = 2\*index;
   4. array[ 2\*index ] = 2\*index;
8. **[0.000] (IP:281473913979145 | 03:43:24 | 03:44:45 | 01:21 | 80.694)**   
   Given:  
   1. public class Test {  
   2. public static void main(String [] args) {  
   3. System.out.println(args.length > 4 &&  
   4. args[4].equals(“-d”));  
   5. }  
   6. }  
   If the program is invoked using the command line:  
   java Test One Two Three –d  
   What is the result?
   1. An exception is thrown at runtime.
   2. true
   3. x   Compilation fails.
   4. ® false
9. **[0.000] (IP:281473913979145 | 03:44:45 | 03:46:05 | 01:20 | 79.656)**   
   What is the output of the following code fragment:  
     
   int[] z = new int[9];  
     
   z[0] = 7;  
   z[1] = 3;  
   z[2] = 4;  
     
   System.out.println( z[0] + z[1] + " " + z[5] );
   1. The program is defective and will not compile.
   2. x   7 3 0
   3. 7 3 4
   4. ® 10 0
10. **[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" .
11. **[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.
12. **[0.000] (IP:281473913979145 | 03:47:25 | 03:50:11 | 02:46 | 165.402)**   
    Given:  
    12. float f[][][] = new float[3][][];  
    13. float f0 = 1.0f;  
    14. float[][] farray = new float[1][1];  
    What is valid?
    1. ® f[0] = farray;
    2. x   f[0] = farray[0][0];
    3. f[0] = f0;
    4. f[0] = farray[0];
13. **[0.000] (IP:281473913979145 | 03:50:11 | 03:51:00 | 00:49 | 49.306)**   
    What is the meaning of null?
    1. ® A reference variable that contains null is not referring to an object.
    2. It is a special value used to indicate an error condition.
    3. It is another name for zero.
    4. x   It is the String object that contains no characters.
14. **[0.500] (IP:281473913979145 | 03:51:00 | 03:51:41 | 00:41 | 40.611)**   
    Given:  
    1. public interface Foo {  
    2. int k = 4;  
    3. }  
    Which three are equivalent to line 2? (Choose three)
    1. volatile int k = 4;
    2. abstract int k = 4;
    3. protected int k = 4;
    4. ® static int k = 4;
    5. ® public int k = 4;
    6. ® final int k = 4;
15. **[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
16. **[0.000] (IP:281473913979145 | 03:52:52 | 03:54:40 | 01:48 | 82.248)**   
    Does a programmer always know how long an array will be when the program is being written?
    1. Yes---the program will not compile without the length being declared.
    2. x   No---arrays can grow to whatever length is needed.
    3. Yes---otherwise the program will not run correctly.
    4. ® No---the array object is created when the program is running, and the length might change from run to run.
17. **[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] );
18. **[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
19. **[1.000] (IP:281473913979145 | 03:57:04 | 04:00:56 | 03:52 | 231.75)**   
    Given:  
      
    int[][] items =  
    { {0, 1, 3, 4},  
    {4, 3, 99, 0, 7 },  
    {3, 2} } ;  
      
    Which of the following fragments prints out every element of items?
    1. for ( int row=0; row < items.length; row++ )  
       {  
       for ( int row=0; row < items[row].length; row++ )  
       System.out.print( items[row][col] + " ");  
       System.out.println();  
       }
    2. for ( int row=0; row < items.length; row++ )  
       {  
       System.out.println();  
       for ( int col=0; col < items[col].length; col++ )  
       System.out.print( items[row][col] + " ");  
       }
    3. x ® for ( int row=0; row < items.length; row++ )  
       {  
       System.out.println();  
       for ( int col=0; col < items[row].length; col++ )  
       System.out.print( items[row][col] + " ");  
       }
    4. for ( int row=0; row < items.length; row++ )  
       {  
       System.out.println();  
       for ( int col=0; col < items.length; col++ )  
       System.out.print( items[row][col] + " ");  
       }
20. **[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" ;
21. **[0.000] (IP:281473913979145 | 04:02:06 | 04:03:39 | 01:33 | 93.247)**   
    Fill in the blanks of the following code fragment so that the elements of the array are printed in reverse order, starting with the last element.  
      
    int[] egArray = { 2, 4, 6, 8, 10, 1, 3, 5, 7, 9 };  
      
    for ( int index= \_\_\_\_\_\_\_\_ ; \_\_\_\_\_\_\_\_\_\_\_\_\_ ; \_\_\_\_\_\_\_\_\_\_\_\_\_\_ )  
    System.out.print( egArray[ index ] + " " );
    1. x   index = length-1; index > 0; index--
    2. index = 0; index < egArray.length; index--
    3. ® index = egArray.length-1; index >= 0; index--
    4. index = length; index < 0; index--
22. **[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] ;
23. **[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
24. **[0.000] (IP:281473913979145 | 04:05:20 | 04:06:10 | 00:50 | 50.065)**   
    You want a class to have access to members of another class in the same package. Which  
    is the most restrictive access that accomplishes this objective?
    1. ® default access
    2. x   private
    3. transient
    4. public
    5. protected
25. **[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
26. **[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
27. **[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
28. **[0.600] (IP:281473913979145 | 04:07:40 | 04:08:19 | 00:39 | 39.173)**   
    Which three statements are true? (Choose three)
    1. The default constructor initializes method variables.
    2. x   If a class lacks a no-arg constructor, the compiler always creates a default constructor.
    3. ® The default constructor has the same access as its class.
    4. x ® The default constructor invoked the no-arg constructor of the superclass.
    5. x ® The compiler creates a default constructor only when there are no other constructors for the class.
29. **[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;
30. **[0.000] (IP:281473913979145 | 04:08:58 | 04:09:28 | 00:30 | 30.289)**   
    What is the output of the following code fragment:  
      
    int[] ar = {2, 4, 6, 8 };  
      
    ar[0] = 23;  
    ar[3] = ar[1];  
      
    System.out.println( ar[0] + " " + ar[3] );
    1. ® 23 4
    2. x   23 2
    3. 31
    4. 2 8
31. **[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
32. **[0.000] (IP:281473913979145 | 04:09:48 | 04:11:18 | 01:30 | 90.222)**   
    Given:  
    1. public class ArrayTest {  
    2. public static void main(String[] args) {  
    3. float fl[], f2[];  
    4. fl = new float[10];  
    5. f2 = f1;  
    6. System.out.println(“f2[0]= “ + f2[0]);  
    7. }  
    8. }  
    What is the result?
    1. An error at line 6 causes compile to fail.
    2. ® It prints f2[0] = 0.0.
    3. An error at line 5 causes compile to fail.
    4. It prints f2[0] = NaN.
    5. x   An error at line 6 causes an expectation at runtime.
33. **[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[];
34. **[1.000] (IP:281473913979145 | 04:11:56 | 04:12:33 | 00:37 | 37.161)**   
    Which three form part of correct array declarations? (Choose three)
    1. x ® public final int [] a
    2. private int [3] a []
    3. x ® static int [] a
    4. x ® public int a []
    5. public [] int a
    6. private int a [3]
35. **[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] );
36. **[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
37. **[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} };
38. **[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
39. **[0.000] (IP:281473913979145 | 04:17:13 | 04:18:00 | 00:47 | 47.124)**   
    What is the difference between  
    String rats;   
    and   
    String[] rats; ?
    1. x   The first initializes rats to null; the second initializes rats to an array of nulls
    2. The first constructs a single String object; the second constructs an array of String objects.
    3. There is no difference; both declare rats to be a reference variable
    4. ® The first declares rats to be a reference to a String object, the second declares rats to be a reference to an array of String references
40. **[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
41. **[0.000] (IP:281473913979145 | 04:18:22 | 04:18:45 | 00:23 | 23.027)**   
    What is the output of the following code fragment:  
      
    int[] y = new int[5];  
      
    y[0] = 34;  
    y[1] = 88;  
      
    System.out.println( y[0] + " " + y[1] + " " + y[5] );
    1. 0 34 88
    2. ® The program is defective and will not compile.
    3. x   34 88 0
    4. 34 88 88
42. **[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] );
43. **[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
44. **[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++ ;
45. **[0.000] (IP:281473913979145 | 04:20:56 | 04:21:29 | 00:33 | ------ )**   
    Assume the following:  
    int[] a = {25, -3, 6, 0, -3, 6};  
    int[] b = new int[100];  
    int[] c = null;  
    \_\_\_\_\_\_\_\_\_\_\_\_\_ Value of a.length ?
    1. 2
    2. x   3
    3. ® 6
    4. 4

**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