Chapter 4 – Arrays & Strings

Answers

1. A,B

   The “main” method in a class needs to have the following 2 modifiers “public” and “static” listed first, then the return type of “void” then the method name “main”. It does not matter if the word “public” comes before the word “static” or vice versa, but it is customary to put the word “public” first.

   Then the method needs to accept a String array. The name of the variable that holds the reference to this array does not matter, but it is customary to use the word “args”.

   The most popular way to declare the “main” method is as follows: “public static void main (String args[])”

   Answers A and B are both fine since they follow the rules mentioned above. Answers C and D are not legal since they have the return value “void” before the modifier “static” and this is not legal. Answer E is not legal since it is missing the modifier “public” and this is mandatory for the “main” method.

2. A,B

   The “main” method in a class needs to have the following 2 modifiers “public” and “static” listed first, then the return type of “void” then the method name “main”. It does not matter if the word “public” comes before the word “static” or vice versa, but it is customary to put the word “public” first.

   Then the method needs to accept a String array. The name of the variable that holds the reference to this array does not matter, but it

is customary to use the word “args”.

The most popular way to declare the “main” method is as follows:
“public static void main (String args[])”

Answers A and B are both fine since they follow the rules mentioned above. To define an array the [] can be placed either before the variable name (here args) or after.

Answer C is not legal since the method is not defined to accept any parameters.

Answer D is not legal since the parameter defined is a 2 dimensional array.

Answer E is not legal since the return value “void” is mentioned first, and it needs to be placed after the keywords “public” and “static”.

3. B,E

The “main” method in a class needs to have the following 2 modifiers “public” and “static” listed first, then the return type of “void” then the method name “main”. It does not matter if the word “public” comes before the word “static” or vice versa, but it is customary to put the word “public” first.

Then the method needs to accept a String array. The name of the variable that holds the reference to this array does not matter, but it is customary to use the word “args”.

The most popular way to declare the “main” method is as follows:
“public static void main (String args[])”

Answers B and E are legal since they follow the rules mentioned above. They are the same just the words “public” and “static” are flipped.

Answers A and C are not legal since the “void” keyword comes second in the list, and it needs to be the third keyword listed here.

Answer D is not legal since it is declaring the method to be “private” and it must be “public”.

4. B

following 2 modifiers “public” and “static” listed first, then the return type of “void” then the method name “main”. It does not matter if the word “public” comes before the word “static” or vice versa, but it is customary to put the word “public” first.

Then the method needs to accept a String array. The name of the variable that holds the reference to this array does not matter, but it is customary to use the word “args”.

The most popular way to declare the “main” method is as follows: “public static void main (String args[])”

Answer B is correct since it follows the rules mentioned above for the “main” method.

Answer A is incorrect since the parameter received only has a type and not a name. The “args” part is missing.

Answers C and D are incorrect since the “void” return type appears second in the list, and it needs to appear third. And D is also incorrect since no parameters are defined to be passed.

Answer E is incorrect since the name of the method needs to be “main” and here it is “mainMethod”.

5. B,C,D

In this example an int array is declared, and has a dimension of 10. By default each spot in the array has the default value of 0, since it is an int array.

Answers B and C are correct since they are accessing valid spots in the array, and we know that each spot was initialized to be 0.

Answer D is correct since we know that the length of the array is 10. An array is a Java object, and it has a built in variable called “length” that holds the number of spots in the array. Since it is a variable value and not a method, it is accessed with “length” and not with “length()”

Answer A is incorrect since the number of spots in an array is the length – 1, which is 9 here. The indexes in this array go from 0 to 9 (10 spots in all).

Answer E is incorrect since null is a value assigned for an object, and not for a primitive type.

6. A

In this example there is an int variable called “index”, which is initialized to the value of 2. And there is an array of Strings called “vec”, which is declared and filled with 9 Strings on one line.

On line 3 the String reference held in one of the spaces in the array is changed. First the value of “index” is decreased by 1, making “index” now hold a value of 1, since before it was 2. Then the String in vec[1], which is “David” is altered. A new String object is created that holds the text “David#”, and its reference is placed into the vec[1] spot.

Answer A is correct since that is the only name that had a # added

to it.

7. E

   In this example there is an int variable called “index”, which is initialized to the value of 2. And there is an array of Strings called “vec”, which is declared and filled with a lot of Strings on one line.

   On line 3 the String reference held in one of the spaces in the array is changed. First the value of “index” is increased by 1, making “index” now hold a value of 3, since before it was 2. Then the String in vec[3], which is “D” is altered. A new String object is created that holds the text “D#”, and its reference is placed into the vec[3] spot.

   Answer E is correct since that is the only letter that had a # added to it.

8. B

   The String class is defined as a “final” class, so it cannot be sub classed.

9. B

   The “==” operator checks if the references of the 2 String objects point to the same object. The “equals” method checks if the contents of the String objects are the same.

   Line 1 adds the String text “abba” to the String pool, since it declares a String using just =” “ and not using the new keyword. Line 2 creates a new String using the new keyword, which does not add this String to the pool, so the 2 objects are different objects. Therefore the “equals” test returns true and the word “Tel-Aviv” is printed.

10. A,B

    Line 1 adds the String text “abba” to the String pool, since it declares a String using just =” “ and not using the new keyword. The

variable “str1” holds a reference to this String object in the pool.

Line 2 also defines a new String in the same manner as line 1, so in this case it looks in the String pool and sees that a String object with this text already exists there, so the variable “str2” points to this same String object in the pool that “str1” points to.

Since both variables contain references to the same object, and also therefore the text is the same for both, both the “==” and the “equals” return true, so the words Amsterdam and Tel-Aviv are both printed.

11. E

In this example 3 variables of type StringBuffer are created, called str1, str2 and str3. On line 1 str1 is assigned a reference to a StringBuffer object that holds the text “abba”. On line 2 str2 is assigned a reference to a StringBuffer object that holds the text “abcd”.

On line 3 the text “Haifa” is appended to the text in str2, so the text in the object that str2 refers to is now “abcdHaifa”. Since the text in a StringBuffer is mutable, str2 still refers to the same StringBuffer object.

12. C

A compilation error will occur on line 6 since that line is trying to compare the reference of a String object to the reference of a StringBuffer object, and this is not legal.

13. A

A StringBuffer object is mutable, which means that its contents can be changed and it still remains the same object.

14. C

This code will compile fine.

15. A

The size of an array can be specified with a variable as long as that variable holds an integer value.

16. B,D,E

In this example on line 3, an evaluation takes place first, from left to right. Since the variable “num” at this point holds the value of 1 (which was assigned to it on line 2), the “vec[num]” expression on line 3 is evaluated to be “vec[1]”.

Then “num” is evaluated, it is just a reference to the variable “num”. And then the constant 0 is evaluated, which requires no work.

After this the assignment takes place, which is done from right to left. First the variable “num” is assigned the value of 0. Then the expression “vec[1]” is assigned the value of num, which is 0.

Answer B is correct since the first value in the array, in spot 0, was never changed, and is still 10.

Answer D is correct since we said above that the variable “num” was assigned the value of 0 on line 3.

And answer E is correct since the value of vec[1] is 0, and that is the value of num.

Answer A is incorrect since we already said that the value of num was changed from 1 to 0 on line 3. And answer C is incorrect since we already said that the value held in the vec[1] spot of the array was changed to be 0, and does not hold the number 11 anymore.

17. D

A Java class can have a pool of literal Strings. The idea of this

String pool is that if the content of more than one String is the same, then different variables can reference the same String object in the pool, to cut down on memory used.

In this example the variables str1 and str2 each hold a reference to a separate String object even though their contents is the same, since they were created with the “new” keyword. Therefore the evaluation on line 7 will return false, and the word “Hello” will not be printed.

On lines 9 and 10 the “intern()” method is called on each of these String objects. The method “intern()” returns a reference to a String object, and here that reference is not being placed anywhere. If line 9 were to be: “str1 = str1.intern()” then the str1 variable would hold a reference to an object in the pool, but in our case that does not happen and str1 holds that same reference that it originally held. The same is with str2 on line 10. Therefore the evaluation on line 11 also returns false, so nothing is printed to the screen.

18. A

A Java class can have a pool of literal Strings. The idea of this String pool is that if the content of more than one String is the same, then different variables can reference the same String object in the pool, to cut down on memory used.

On line 5 a new String is created with a plain assignment, and without the “new” keyword, and therefore its contents is placed directly into a String pool. On line 6 another String is created, with the “new” keyword and with the “intern()” method. When a String is created with the “new” keyword alone it is not added to the String pool. When the “intern()” method is used, it searches the String pool to see if it already contains a String with the same contents, which is “abc” here. If it contains the String literal already then a reference to that String object is returned, which is what is happening here. If the String literal were not to be found in the pool then the new String

would be added to the pool and a reference returned.

On line 7 the evaluation on true since str1 and str2 both hold references to the same String object, therefore the text is printed to the screen.

19. A

In this example the 2 variables str1 and str2 hold references to the same String object since their content is the same (“abc”), and since the intern() method was called when both of them were created. Since the 2 variables refer to the same object the evaluation on line 7 returns true, and the text “Hello” is printed.

On lines 9 and 10 the calls to “intern()” have no effect since their return value, a String, is not placed anywhere.

The evaluation on line 11 returns true as well, it is identical to the evaluation done on line 7. The word “Israel” is then printed.

20. A

When the “trim()” method is called on a String a new String object is created that has the whitespace characters from either end of the String removed, and a reference to this new String object is returned

21. A

The “equals” method will always return false if the objects being compared are of different types.

22. C,D

On line 1 a new String variable called str1 is assigned a reference to a String object holding the text “Shalom”. On line 2 a new String variable called str2 is assigned a reference to the same object that str1 points to. On line 3 another variable called str3 is assigned a reference to a newly created String object, which also holds the text “Shalom”. Since the “new” keyword is used here, a new object is

created, so str3 refers to a different object than str1 and str2 refer to. On line 4, the str4 variable is assigned a reference to the same String object that str3 refers to.

On line 5 str4 is assigned a reference to a new String object that holds the text “Shalom Israel”. Since a String object is immutable, a new object is created here, and its reference is placed in the str4 variable. The str3 variable now refers a different String object than str4 refers to. Therefore the check on line 6, which is checking if the 2 variables refer to the same object returns false. The check on line 8 also returns false since the text in the 2 String objects that these variables refer to is now different. On is “Shalom” and the other is “Shalom Israel”.

The checks on lines 10 and 12 both return true since the two variables, str1 and str2 both refer to the same String object, and therefore contain the same text value as well.

23. A

In this example there is a class called Dot that holds 2 instance variables of type int called x and y. On line 5 a new array of type Dot is created and filled, and its reference is placed into a variable called “vec1”. On line 6 a new array of “Dot” objects is created, and its size is set to 3, with each spot holding a null value for now. This array is called vec2.

On line 7 all of the references in vec1 are copied to the array in vec2 with the “arraycopy” method. Therefore, for example, the 0 spot in both arrays both hold references to the same Dot object, and the same is true for the number 1 and 2 spots in the array.

On line 8 the x and y values in the Dot object referenced by the 0 spot in the vec1 array are both assigned the value of 1. And then on line 9 the x and y values in the Dot object referenced by the 1 spot in the vec2 array are both assigned the value of 1.

On line 10 when the values of vec2[0] and vec1[1] are added, each of these values is 1 so the sum is 2. Since both of the 0 spots in vec1 and vec2 refer to the same object, when the x and y values of the object referenced in the 0 spot values were changed on line 8, either array would get the same values in return.

24. A

The “arrayCopy” method copies the contents of one array onto another array. If object references reside in the original array, then copies of these references get placed in the new array. Therefore the cells in the 2 arrays both have references that point to the same objects.

25. A

In this example a String object is created, holding the text “abba”, and its reference is placed into the variable str1. On the next line a StringBuffer object is created, also holding the text “abba”, and its reference is placed into the variable str2. On the following line the text in the object that str2 refers to is changed to be “abba Haifa”. This is valid since a StringBuffer object is mutable. This reference is then placed into a second StringBuffer variable called str3.

The comparison “if(str2==str3)” returns true since str2 and str3 are both StringBuffer variables that refer to the same StringBuffer object, so therefore the text “Paris” is printed. The comparison “if(str1.equals(str2))” returns false since str1 refers to a String object and str2 refers to a StringBuffer object, so the text “Tel-Aviv” is not printed.

26. E

In this example 2 arrays are created of type “Dot”. The first one “vec1” is filled and the second one “vec2” is not filled, so all of it’s Dot objects will be initialized to null. Then “arraycopy” is called, and all of the contents in “vec1” are copied to the contents in “vec2”. This

now means that both of the arrays contain references to the same Dot objects, for example vec1[0] and vec2[0] both contain references referring to the same Dot object.

Then the values in “vec1[0].x” and in “vec1[0].y” are both set to 2, and the values in “vec2[1].x” and in “vec2[1].y” are both set to 2 as well. Then the sum of “vec2[0].x+vec1[1].y” is printed to the screen. Since the references in the 2 arrays are to the same Dot objects, the values of “vec2[0].x” and “vec1[1].y” are each 2, so the sum printed is 4.

27. B,D,E

In this example an array of type int is declared and filled to hold 3 numbers. Its reference is placed into a variable called “vec”. On the next line a variable called “num” is declared and is ititialized to hold the value of 1. On the following line the number in the vec[1] spot of the array “vec” is assigned the value of 1. Before it held the value of 11.

On the last line the “vec[num]” expression is evaluated first, and at this point the variable “num” holds the value of 1, so the expression here is “vec[1]’. Then the variable “num” is assigned the value of 0, and this value is assigned to vec[1] spot in the array.

Answer A is incorrect since the ending value of “num” is 0 and not 1. Answer B is correct since the number in the vec[0] spot in the array is 10, it was never changed. Answer C is incorrect since the value that vec[1] now holds is 0, and not 11 anymore. Answers D and E are correct since the ending value of “num” was 0, and this is the same value that is in the vec[1] spot.

28. B

The content of a String object is immutable, which means that it cannot be changed.

29. B

The length of this array is 10, the value that was passed in the [].
The last index in this array is 9, since the index count starts from 0.

30. A

In this example first an array of type int is created with 4 elements, and it is initialized to hold the values “1, 2 3 and 4”. Then the variable “sum” is created and is initialized to hold the value 10.
In the “for” loop each number in the array is added to the value of “sum”. When the loop finishes the value of sum” is 20. On the line after the loop, the value of the variable “sum” decreases by 10. Therefore the variable “sum” now holds the value of 10. Then this “sum” value is printed.

31. B

An array is an object in the Java language, and by default has a property called “length”.

The length of an array is the number of elements that it can hold, and here the array is initialized to hold 4 elements, so the length is 4.

32. A

The length of a String object is determined by the method “length()”, and not by a property “length” as is done for an array.

33. E

In this example 2 String arrays are created, one called “vec” and the other called “tavs”. They are each initialized with values.

When the line “vec = tavs” is run, the references in each of the spots in the vec array now point to the String objects in the corresponding spots in the tavs array. Therefore vec[1] and tavs[1] both point to the same String object, which holds the value of “b”, and the same goes for the other spots in the 2 arrays.

Therefore when the values in vec are printed, the resulting String concatenates the values in the [1], [2], and [3] spots of the vec, resulting in “bcd”.

34. E

Boolean variables can only hold the values “true” or “false”. In this example an array of type boolean is created, so each spot in the array can hold one of the two boolean values. This code is trying to assign the values “Boolean”, “0” and “1” to spots in the array, and this is not legal.

35. E

When creating and filling an array in one statement, the [] should not contain any value. The length of the array is determined by the number of elements within the {…}.

36. D

In this example two String objects are created on lines 1 and 2, both holding the value “abba”, and at this point they both hold references to the same String object.

On line 3 the method “toUpperCase()” is called on str2, which creates and returns a new String object that holds the value “ABBA”. A new String object is created here since the value in a String object cannot be changed. The reference to this new String object is placed into the str1 variable, but the str2 variable still holds a reference to the original String object.

Neither the references not the contents of these 2 variables are the same, so nothing is printed.

37. C

This code will not compile on line 1 since this is not the proper way to create an array. This line should be: “String str[] = new String [3];”

38. B

In this example the 2 String variables str1 and str2 hold references to different objects, since they were created with the “new” keyword and the method “intern()” was not called on them.

Since each variable holds different references, the evaluation on line 3 returns false, since the “==” operator compares references. But since the contents of the 2 Strings are the same the evaluation on line 6 returns true, since the “equals” method compares contents. Therefore the word “Tel-Aviv” is printed.