socrative

Answer Key
February 24, 2021

Java OCA Chapter 3 Review ('Copy')

1. What is output by the following code? 1: public class Fish { 2: public static void main(String[] args) { 3: int numFish = 4; 4: String fishType = "tuna"; 5: String anotherFish = numFish + 1; 6: System.out.println(anotherFish + " " + fishType); 7: System.out.println(numFish + " " + 1); 8: } }

- **A** 4 1
- **B** 41
- **C** 5
- **D** 5 tuna
- **E** 5tuna
- **F** 51tuna
- ✓ **G** The code does not compile.

i Line 5 does not compile. This question is checking to see if you are paying attention to the types. numFish is an int and 1 is an int. Therefore, we use numeric addition and get 5. The problem is that we can't store an int in a String variable. Supposing line 5 said String anotherFish = numFish + 1 + "";. In that case, the answer would be options A and D. The variable defined on line 5 would be the string "5", and both output statements would use concatenation.

2. Which of the following are output by this code? (Choose all that apply) 3: String s = "Hello"; 4: String t = new String(s); 5: if ("Hello".equals(s)) System.out.println("one"); 6: if (t == s) System.out.println("two"); 7: if (t.equals(s)) System.out.println("three"); 8: if ("Hello" == s) System.out.println("four"); 9: if ("Hello" == t) System.out.println("five");

- ✓ **A** one
- **B** two
- ✓ **C** three
- ✓ **D** four
- **E** five
- **F** The code does not compile.

i The code compiles fine. Line 3 points to the String in the string pool. Line 4 calls the String constructor explicitly and is therefore a different object than s. Lines 5 and 7 check for object equality, which is true, and so print one and three. Line 6 uses object reference equality, which is not true since we have different objects. Line 7 also compares references but is true since both references point to the object from the string pool. Finally, line 8 compares one object from the string pool with one that was explicitly constructed and returns false.

3. Which are true statements? (Choose all that apply)

- A An immutable object can be modified.
- ✓ B An immutable object cannot be modified.
- ✓ C An immutable object can be garbage collected.
- D An immutable object cannot be garbage collected.
- ✓ E String is immutable.
- F StringBuffer is immutable.
- G StringBuilder is immutable.

i Immutable means the state of an object cannot change once it is created. Immutable objects can be garbage collected just like mutable objects. String is immutable. StringBuilder can be mutated with methods like append(). Although StringBuffer isn't on the exam, you should know about it anyway in case older questions haven't been removed.

4. What is the result of the following code? 7: StringBuilder sb = new StringBuilder(); 8: sb.append("aaa").insert(1, "bb").insert(4, "ccc"); 9: System.out.println(sb);

- A abbaaccc
- ✓ B abbaccca
- C bbaaaccc
- D bbaaccca
- E An exception is thrown.
- F The code does not compile.

i This example uses method chaining. After the call to append(), sb contains "aaa". That result is passed to the first insert() call, which inserts at index 1. At this point sb contains abbaa. That result is passed to the final insert(), which inserts at index 4, resulting in abbaccca.

5. What is the result of the following code? 2: String s1 = "java"; 3: StringBuilder s2 = new StringBuilder("java"); 4: if (s1 == s2) 5: System.out.print("1"); 6: if (s1.equals(s2)) 7: System.out.print("2");

- A 1
- B 2
- C 12
- D No output is printed.
- E An exception is thrown.
- ✓ F The code does not compile.

i The question is trying to distract you into paying attention to logical equality versus object reference equality. It is hoping you will miss the fact that line 4 does not compile. Java does not allow you to compare String and StringBuilder using ==.

6. What is the result of the following code? public class Lion { public void roar(String roar1, StringBuilder roar2) { roar1.concat("!!!"); roar2.append("!!!"); } public static void main(String[] args) { String roar1 = "roar"; StringBuilder roar2 = new StringBuilder("roar"); new Lion().roar(roar1, roar2); System.out.println(roar1 + " " + roar2); } }

- A roar roar
- ✓ B roar roar!!!
- C roar!!! roar
- D roar!!! roar!!!
- E An exception is thrown.
- F The code does not compile.

i A String is immutable. Calling concat() returns a new String but does not change the original. A StringBuilder is mutable. Calling append() adds characters to the existing character sequence along with returning a reference to the same object.

7. Which are the results of the following code? (Choose all that apply) String letters = "abcdef"; System.out.println(letters.length()); System.out.println(letters.charAt(3)); System.out.println(letters.charAt(6));

- A 5
- ✓ B 6
- C c
- ✓ D d
- ✓ E An exception is thrown.
- F The code does not compile.

i length() is simply a count of the number of characters in a String. In this case, there are six characters. charAt() returns the character at that index. Remember that indexes are zero based, which means that index 3 corresponds to d and index 6 corresponds to 1 past the end of the array. A StringIndexOutOfBoundsException is thrown for the last line.

8. Which are the results of the following code? (Choose all that apply) String numbers = "012345678"; System.out.println(numbers.substring(1, 3)); System.out.println(numbers.substring(7, 7)); System.out.println(numbers.substring(7));

- ✓ A 12
- B 123
- C 7
- ✓ D 78
- ✓ E A blank line
- F An exception is thrown.
- G The code does not compile.

i substring() has two forms. The first takes the index to start with and the index to stop immediately before. The second takes just the index to start with and goes to the end of the String. Remember that indexes are zero based. The first call starts at index 1 and ends with index 2 since it needs to stop before index 3. The second call starts at index 7 and ends in the same place, resulting in an empty String. This prints out a blank line. The final call starts at index 7 and goes to the end of the String.

9. What is the result of the following code? 3: String s = "purr"; 4: s.toUpperCase(); 5: s.trim(); 6: s.substring(1, 3); 7: s += " two"; 8: System.out.println(s.length());

- **A** 2
- **B** 4
- ✓ **C** 8
- **D** 10
- **E** An exception is thrown.
- **F** The code does not compile.

i This question is trying to see if you know that String objects are immutable. Line 4 returns "PURR" but the result is ignored and not stored in s. Line 5 returns "purr" since there is no whitespace present but the result is again ignored. Line 6 returns "ur" because it starts with index 1 and ends before index 3 using zero-based indexes. The result is ignored again. Finally, on line 6 something happens. We concatenate four new characters to s and now have a String of length 8.

10. What is the result of the following code? 13: String a = ""; 14: a += 2; 15: a += 'c'; 16: a += false; 17: if (a == "2cfalse") System.out.println("=="); 18: if (a.equals("2cfalse")) System.out.println("equals");

- **A** Compile error on line 14
- **B** Compile error on line 15
- **C** Compile error on line 16
- **D** Compile error on another line
- **E** ==
- ✓ **F** equals
- **G** An exception is thrown.

i a += 2 expands to a = a + 2. A String concatenated with any other type gives a String. Lines 14, 15, and 16 all append to a, giving a result of "2cfalse". The if statement on line 18 returns false because the values of the two String objects are the same using object equality. The if statement on line 17 returns false because the two String objects are not the same in memory. One comes directly from the string pool and the other comes from building using String operations.

11. What is the result of the following code? 4: int total = 0; 5: StringBuilder letters = new StringBuilder("abcdefg"); 6: total += letters.substring(1, 2).length(); 7: total += letters.substring(6, 6).length(); 8: total += letters.substring(6, 5).length(); 9: System.out.println(total);

- A 1
- B 2
- C 3
- D 7
- ✓ E An exception is thrown.
- F The code does not compile.

> i Line 6 adds 1 to total because substring() includes the starting index but not the ending index. Line 7 adds 0 to total. Line 8 is a problem: Java does not allow the indexes to be specified in reverse order and the code throws a StringIndexOutOfBoundsException.

12. What is the result of the following code? (Choose all that apply) StringBuilder numbers = new StringBuilder("0123456789"); numbers.delete(2, 8); numbers.append("-").insert(2, "+"); System.out.println(numbers);

- ✓ A 01+89–
- B 012+9–
- C 012+–9
- D 0123456789
- E An exception is thrown.
- F The code does not compile.

> i First, we delete the characters at index 2 until the character one before index 8. At this point, 0189 is in numbers. The following line uses method chaining. It appends a dash to the end of the characters sequence, resulting in 0189–, and then inserts a plus sign at index 2, resulting in 01+89–.

13. What is the result of the following code? StringBuilder b = "rumble"; b.append(4).deleteCharAt(3).delete(3, b.length() - 1); System.out.println(b);

- A rum
- B rum4
- C rumb4
- D rumble4
- E An exception is thrown.
- ✓ F The code does not compile.

> i This is a trick question. The first line does not compile because you cannot assign a String to a StringBuilder. If that line were StringBuilder b = new StringBuilder("rumble"), the code would compile and print rum4. Watch out for this sort of trick on the exam. You could easily spend a minute working out the character positions for no reason at all.

14. Which of the following can replace line 4 to print "avaJ"? (Choose all that apply) 3: StringBuilder puzzle = new StringBuilder("Java"); 4: // INSERT CODE HERE 5: System.out.println(puzzle);

- ✓ **A** puzzle.reverse();
- **B** puzzle.append("vaJ$").substring(0, 4);
- ✓ **C** puzzle.append("vaJ$").delete(0, 3).deleteCharAt(puzzle.length() - 1);
- **D** puzzle.append("vaJ$").delete(0, 3).deleteCharAt(puzzle.length());
- **E** None of the above

i The reverse() method is the easiest way of reversing the characters in a StringBuilder; therefore, option A is correct. Option B is a nice distraction—it does in fact return "avaJ". However, substring() returns a String, which is not stored anywhere. Option C uses method chaining. First it creates the value "JavavaJ$". Then it removes the first three characters, resulting in "avaJ$". Finally, it removes the last character, resulting in "avaJ". Option D throws an exception because you cannot delete the character after the last index. Remember that deleteCharAt() uses indexes that are zero based and length()counts starting with 1.

15. Which of these array declarations is not legal? (Choose all that apply)

- **A** int[][] scores = new int[5][];
- **B** Object[][][] cubbies = new Object[3][0][5];
- ✓ **C** String beans[] = new beans[6];
- **D** java.util.Date[] dates[] = new java.util.Date[2][];
- ✓ **E** int[][] types = new int[];
- ✓ **F** int[][] java = new int[][];

i Option C uses the variable name as if it were a type, which is clearly illegal. Options E and F don't specify any size. Although it is legal to leave out the size for later dimensions of a multidimensional array, the first one is required. Option A declares a legal 2D array. Option B declares a legal 3D array. Option D declares a legal 2D array. Remember that it is normal to see on the exam types you might not have learned. You aren't expected to know anything about them.

16. Which of these compile when replacing line 8? 7: char[]c = new char[2]; 8: // INSERT CODE HERE

- **A** int length = c.capacity;
- **B** int length = c.capacity();
- ✓ **C** int length = c.length;
- **D** int length = c.length();
- **E** int length = c.size;
- **F** int length = c.size();
- **G** None of the above

i Arrays define a property called length. It is not a method, so parentheses are not allowed.

17. Which of these compile when replacing line 8? 7: ArrayList l = new ArrayList(); 8: // INSERT CODE HERE

- A int length = l.capacity;
- B int length = l.capacity();
- C int length = l.length;
- D int length = l.length();
- E int length = l.size;
- ✓ F int length = l.size();
- G None of the above

i The ArrayList class defines a method called size().

18. Which of the following are true? (Choose all that apply)

- ✓ A An array has a fixed size.
- B An ArrayList has a fixed size.
- ✓ C An array allows multiple dimensions.
- ✓ D An array is ordered.
- ✓ E An ArrayList is ordered.
- F An array is immutable.
- G An ArrayList is immutable.

i An array is not able to change size and can have multiple dimensions. Both an array and ArrayList are ordered and have indexes. Neither is immutable. The elements can change in value.

19. Which of the following are true? (Choose all that apply)

- A Two arrays with the same content are equal.
- ✓ B Two ArrayLists with the same content are equal.
- ✓ C If you call remove(0) using an empty ArrayList object, it will compile successfully.
- D If you call remove(0) using an empty ArrayList object, it will run successfully.
- E None of the above

i An array does not override equals() and so uses object equality. ArrayList does override equals() and defines it as the same elements in the same order. The compiler does not know when an index is out of bounds and thus can't give you a compiler error. The code will throw an exception at runtime, though.

20. What is the result of the following statements? 6: List<String> list = new ArrayList<String>(); 7: list.add("one"); 8: list.add("two"); 9: list.add(7); 10: for(String s : list) System.out.print(s);

- A onetwo
- B onetwo7
- C onetwo followed by an exception
- ✓ D Compiler error on line 9
- E Compiler error on line 10

i The code does not compile because list is instantiated using generics. Only String objects can be added to list and 7 is an int.

21. What is the result of the following statements? 3: ArrayList<Integer> values = new ArrayList<>(); 4: values.add(4); 5: values.add(5); 6: values.set(1, 6); 7: values.remove(0); 8: for (Integer v : values) System.out.print(v);

- A 4
- B 5
- ✓ C 6
- D 46
- E 45
- F An exception is thrown.
- G The code does not compile.

i After line 4, values has one element (4). After line 5, values has two elements (4, 5). After line 6, values has two elements (4, 6) because set() does a replace. After line 7, values only one element (6).

22. What is the result of the following? int[] random = { 6, -4, 12, 0, -10 }; int x = 12; int y = Arrays.binarySearch(random, x); System.out.println(y);

- A 2
- B 4
- C 6
- ✓ D The result is undefined.
- E An exception is thrown.
- F The code does not compile.

i The code compiles and runs fine. However, an array must be sorted for binarySearch() to return a meaningful result.

23. What is the result of the following? 4: List<Integer> list = Arrays.asList(10, 4, -1, 5); 5: Collections.sort(list); 6: Integer array[] = list.toArray(new Integer[4]); 7: System.out.println(array[0]);

- ✓ **A** -1
- **B** 10
- **C** Compiler error on line 4
- **D** Compiler error on line 5
- **E** Compiler error on line 6
- **F** An exception is thrown.

i Line 4 creates a fixed size array of size 4. Line 5 sorts it. Line 6 converts back to an array. The brackets aren't in the traditional place, but they are still legal. Line 7 prints the first element, which is now –1.

24. What is the result of the following? 6: String [] names = {"Tom", "Dick", "Harry"}; 7: List<String> list = names.asList(); 8: list.set(0, "Sue"); 9: System.out.println(names[0]);

- **A** Sue
- **B** Tom
- ✓ **C** Compiler error on line 7
- **D** Compiler error on line 8
- **E** An exception is thrown.

i Converting from an array to an ArrayList uses Arrays.asList(names). There is no asList() method on an array instance. If this code were corrected to compile, the answer would be option A.

25. What is the result of the following? List<String> hex = Arrays.asList("30", "8", "3A", "FF"); Collections.sort(hex); int x = Collections.binarySearch(hex, "8"); int y = Collections.binarySearch(hex, "3A"); int z = Collections.binarySearch(hex, "4F"); System.out.println(x + " " + y + " " + z);

- **A** 0 1 –2
- **B** 0 1 –3
- **C** 2 1 –2
- ✓ **D** 2 1 –3
- **E** None of the above
- **F** The code doesn't compile.

i After sorting, hex contains [30, 3A, 8, FF]. Remember that numbers sort before letters and strings sort alphabetically. This makes 30 comes before 8. A binary search correctly finds 8 at index 2 and 3A at index 1. It cannot find 4F but notices it should be at index 2. The rule when an item isn't found is to negate that index and subtract 1. Therefore we get –2–1, which is –3.

26. Which of the following are true statements about the following code? (Choose all that apply) 4: List<Integer> ages = new ArrayList<>(); 5: ages.add(Integer.parseInt("5")); 6: ages.add(Integer.valueOf("6")); 7: ages.add(7); 8: ages.add(null); 9: for (int age : ages) System.out.print(age);

- ✓ **A** The code compiles.
- ✓ **B** The code throws a runtime exception.
- **C** Exactly one of the add statements uses autoboxing.
- ✓ **D** Exactly two of the add statements uses autoboxing.
- **E** Exactly three of the add statements uses autoboxing.

i Lines 5 and 7 use autoboxing to convert an int to an Integer. Line 6 does not because valueOf() returns an Integer. Line 8 does not because null is not an int. The code does compile. However, when the for loop tries to unbox null into an int, it fails and throws a NullPointerException.

27. What is the result of the following? List<String> one = new ArrayList<String>(); one.add("abc"); List<String> two = new ArrayList<>(); two.add("abc"); if (one == two) System.out.println("A"); else if (one.equals(two)) System.out.println("B"); else System.out.println("C");

- **A** A
- ✓ **B** B
- **C** C
- **D** An exception is thrown.
- **E** The code does not compile.

i The first if statement is false because the variables do not point to the same object. The second if statement is true because ArrayList implements equality to mean the same elements in the same order.

28. Which of the following can be inserted into the blank to create a date of June 21, 2014? (Choose all that apply) import java.time.*; public class StartOfSummer { public static void main(String[] args) { LocalDate date = ________________________ } }

- **A** new LocalDate(2014, 5, 21);
- **B** new LocalDate(2014, 6, 21);
- **C** LocalDate.of(2014, 5, 21);
- ✓ **D** LocalDate.of(2014, 6, 21);
- **E** LocalDate.of(2014, Calendar.JUNE, 21);
- ✓ **F** LocalDate.of(2014, Month.JUNE, 21);

i Options A and B are incorrect because LocalDate does not have a public constructor. Option C is incorrect because months start counting with 1 rather than 0. Option E is incorrect because it uses the old pre–Java 8 way of counting months, again beginning with 0. Options D and F are both correct ways of specifying the desired date.

29. What is the output of the following code? LocalDate date = LocalDate.parse("2018-04-30", DateTimeFormatter.ISO_LOCAL_DATE); date.plusDays(2); date.plusHours(3); System.out.println(date.getYear() + " " + date.getMonth() + " " + date.getDayOfMonth());

- **A** 2018 APRIL 2
- **B** 2018 APRIL 30
- **C** 2018 MAY 2
- ✓ **D** The code does not compile.
- **E** A runtime exception is thrown.

i A LocalDate does not have a time element. Therefore, it has no method to add hours and the code does not compile.

30. What is the output of the following code? LocalDate date = LocalDate.of(2018, Month.APRIL, 40); System.out.println(date.getYear() + " " + date.getMonth() + " " + date.getDayOfMonth());

- **A** 2018 APRIL 4
- **B** 2018 APRIL 30
- **C** 2018 MAY 10
- **D** Another date
- **E** The code does not compile.
- ✓ **F** A runtime exception is thrown.

i Java throws an exception if invalid date values are passed. There is no 40th day in April—or any other month for that matter.

31. What is the output of the following code? LocalDate date = LocalDate.of(2018, Month.APRIL, 30); date.plusDays(2); date.plusYears(3); System.out.println(date.getYear() + " " + date.getMonth() + " " + date.getDayOfMonth());

- **A** 2018 APRIL 2
- ✓ **B** 2018 APRIL 30
- **C** 2018 MAY 2
- **D** 2021 APRIL 2
- **E** 2021 APRIL 30
- **F** 2021 MAY 2
- **G** A runtime exception is thrown.

i The date starts out as April 30, 2018. Since dates are immutable and the plus methods have their return values ignored, the result is unchanged. Therefore, option B is correct.

32. What is the output of the following code? LocalDateTime d = LocalDateTime.of(2015, 5, 10, 11, 22, 33); Period p = Period.of(1, 2, 3); d = d.minus(p); DateTimeFormatter f = DateTimeFormatter.ofLocalizedTime(FormatStyle.SHORT); System.out.println(d.format(f));

- A 3/7/14 11:22 AM
- B 5/10/15 11:22 AM
- C 3/7/14
- D 5/10/15
- ✓ E 11:22 AM
- F The code does not compile.
- G A runtime exception is thrown.

i Even though d has both date and time, the formatter only outputs time.

33. What is the output of the following code? LocalDateTime d = LocalDateTime.of(2015, 5, 10, 11, 22, 33); Period p = Period.ofDays(1).ofYears(2); d = d.minus(p); DateTimeFormatter f = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT); System.out.println(f.format(d));

- A 5/9/13 11:22 AM
- ✓ B 5/10/13 11:22 AM
- C 5/9/14
- D 5/10/14
- E The code does not compile.
- F A runtime exception is thrown.

i Period does not allow chaining. Only the last Period method called counts, so only the two years are subtracted.