socrative

Answer Key
February 24, 2021

OCA Bonus Test 3 (pt 2 of 2) ('Copy')

1. Which of the following methods compile? (Choose all that apply)

✓ **A**
```
boolean isWaddling1() {
return true;
}
```

B
```
public boolean isWaddling2() {
return 0;
}
```

C
```
boolean isWaddling3() {
return null;
}
```

D
```
boolean private isWaddling4() {
return null;
}
```

✓ **E**
```
final boolean isWaddling5() {
return false;
}
```

F
```
boolean final isWaddling6() {
return false;
}
```

i boolean is a primitive and can only be true or false. Options B and C are incorrect because they do not return boolean values. Option D is incorrect because the access modifier (private) needs to come before the return type. Option F is incorrect because final needs to be come before the return type. To remember, this think of public static void main(String[] args). The access modifier comes first. Then come optional specifiers like static and final. After that is the return type. Since you have to memorize the main() method signature, this is good to use for comparison. You did memorize the main() method signature, right?

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

A Garbage collection runs when System.gc() is called.

✓ **B** finalize() is called if an object is garbage collected.

C finalize() is always called on an object.

D finalize() can be called twice on the same object.

✓ **E** finalize() can be called twice for the same class.

i System.gc() is a hint that it would be nice to run garbage collection. It does not guarantee that it actually gets run. finalize() is called when the object is first attempted to be garbage collected. It will not be called

more than once for the same instance. If the program has multiple objects for a class, each one will be called. If the program ends before garbage collection runs, finalize() will never be called.

3. What gets printed when running the following?

```
1: public class A {
2: private String value;
3: public void go() {
4: try {
5: System.out.print(value.toString());
6: System.out.print("1");
7: } catch (NullPointerException e) {
8: System.out.print("2");
9: System.exit(0);
10: } finally {
11: System.out.print("3");
12: }
13: System.out.print("4");
14: }
15: public static void main(String[] args) {
16: new A().go();
17: } }
```

A 1

✓ **B** 2

C 3

D 4

E Stack trace of a NullPointerException

F The code does not compile.

i On line 5, the code throws a NullPointerException because the instance variable *value* is still using the default value of null. This is caught on line 7 and then line 8 prints 2. Line 9 immediately ends the program and nothing else is printed.

4. The following code appears in a file named Flight.java. What is the result of compiling this source file?

```
1: public class Flight {
2: private FlightNumber number;
3:
4: public Flight(FlightNumber number) {
5: this.number = number;
6: } }
7: public class FlightNumber {
8: public int value;
9: public String code; }
```

A The code compiles successfully and two bytecode files are generated: Flight.class and FlightNumber.class.

B The code compiles successfully and one bytecode file is generated: Flight.class.

C A compiler error occurs on line 2.

D A compiler error occurs on line 4.

✓ E A compiler error occurs on line 7.

i The code does not compile because no more than one public class is allowed to be defined in the same file. Either the FlightNumber class must not be declared public or it should be moved to its own source file named FlightNumber.java. The compiler error occurs on line 7, so the answer is option E.

5. Which of the following are true about ternary operators? (Choose all that apply)

A The left-hand side of the expression must evaluate to a boolean expression or numeric value of 0 or 1 at runtime.

B The data types of the two expressions on the right-hand side of the equation must match or be able to be numerically promoted to one another.

✓ C All ternary operators can be rewritten as if-then-else statement.

✓ D If the expression on the left-hand side evaluates to true, then the rightmost expression will not be evaluated.

E The terms must be compile-time constant values.

i Option A is incorrect, because only boolean expressions are allowed, and 0 and 1 cannot be parsed as boolean values in Java. Option B is incorrect, because the data types of the expression can be completely different, such as String and int. That said, if the ternary operator is used on the right-hand side of an assignment operator, then the values do need to match or be able to be numerically promoted to one another. That limitation is introduced by the assignment operator, though, not the ternary operator. Option C is correct because ternary operators are just glorified if-then-else statements. As of Java 7, option D is correct, since only one of the right-hand side expressions of the ternary operation is evaluated at runtime. Option E is incorrect, because this phrase relates to the data type of case statement values within switch statements.

6. Which statements are true for a traditional try statement (not a try-with-resources statement)? (Choose all that apply)

- A If a try statement has a catch clause, it is required to have a finally clause.
- ✓ B If a try statement does not have a catch clause, it is required to have a finally clause.
- C If a try statement has a finally clause, it is required to have exactly one catch clause.
- D If a try statement has a finally clause, it is required to have at least one catch clause.
- E A try statement can exist without a catch or finally clause.
- ✓ F A try statement can have both a catch and a finally clause.

i A traditional try statement must have a finally clause and/or one or more catch clauses.

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

- A A StringBuilder can be cast to a String.
- ✓ B A String can be passed to the constructor of a StringBuilder.
- C StringBuilder is immutable.
- D StringBuilder is thread-safe.
- E String is more efficient than StringBuilder when concatenating thousands of values.
- ✓ F StringBuilder is more efficient than String when concatenating thousands of values.

i String and StringBuilder are not in the same class hierarchy and cannot be cast. new StringBuilder("Java ") is valid code. You can write new String(stringBuilder.toString()), but that isn't what option B says. String is immutable but StringBuilder is not. The OCA exam won't require you to know that StringBuffer is thread-safe and StringBuilder is not, but you should know that just in case. Finally, an advantage of StringBuilder is that it can keep appending values without having to create new objects.

8. Which are true statements about the following code? (Choose all that apply)

```
1: public class I {
2:
3: int ii;
4: public void i() {
5:
6: for (int i = 0; i < 5; i++) {
7:
8: }
9: } }
```

- **A** int j = i; can be inserted at line 2.
- **B** int j = i; can be inserted at line 5.
- ✓ **C** int j = i; can be inserted at line 7.
- **D** int j = ii; can be inserted at line 2.
- ✓ **E** int j = ii; can be inserted at line 5.
- **F** The code does not compile.

i i has a scope of the for loop. Therefore, it can be referenced on line 7 but not lines 2 or 5. i has a scope of anywhere in the class once it is defined. Therefore, it can be referenced on line 5. It cannot be referenced on line 2 since it has not been declared yet.

9. What is the result of the following?

```
List<String> fish = new ArrayList<>();
fish.add("goldfish"); fish.add("minnow");
fish.remove("goldfish"); fish.remove("shark");
System.out.println(fish.size());
```

- **A** 0
- ✓ **B** 1
- **C** 2
- **D** An exception is thrown.
- **E** The code does not compile.

i Removing "goldfish" works and decreases the size from 2 to 1. Since "shark" is not in *fish*, remove() returns false but does not change *fish*. Therefore the size is still 1.

10. 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.minus(p);
DateTimeFormatter f = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT);
System.out.println(f.format(d));
```

- **A** 3/7/14 11:22 AM
- **B** 6/7/14 11:22 AM
- **C** 5/10/15 11:22 AM
- ✓ **D** 6/10/15 11:22 AM
- **E** The code does not compile.
- **F** A runtime exception is thrown.

i d.minus(p) isn't assigned anywhere so the result is ignored. Since months are one-based indexes, the answer is option C.

11.

Which lines in Tadpole give a compiler error? (Choose all that apply)

```
1: package animal;
2: public class Frog {
3: protected void ribbit() { }
4: void jump() { }
5: }

1: package other;
2: import animal.*;
3: public class Tadpole extends Frog {
4: public static void main(String[] args) {
5: Tadpole t = new Tadpole();
6: t.ribbit();
7: t.jump();
8: Frog f = new Tadpole();
9: f.ribbit();
10: f.jump();
11: } }
```

- **A** 5
- **B** 6
- ✓ **C** 7
- **D** 8
- ✓ **E** 9
- ✓ **F** 10

i jump() has default (package private) access, which means it can only be accessed from the same package. Tadpole is not in the same package as Frog, causing lines 7 and 10 to give a compiler error. ribbit() has protected access, which means it can only be accessed from a subclass reference or in the

same package. Line 6 is fine because Tadpole is a subclass. Line 9 does not compile because the variable reference is to a Frog. This is the trickiest question you can get on this topic on the exam.

12. Which is the result of the following?

```
List listOne = new ArrayList();
listOne.add(0.5); listOne.add(3.5);
List listTwo = new ArrayList();
listTwo.add(3.5); listTwo.add(0.5);
if(listOne == listTwo)
System.out.println("A");
else if(listOne.equals(listTwo))
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 == uses object reference equality. Since listOne and listTwo point to different objects, they are not ==. equals() looks for the same elements in the same order. Since the numbers are in different orders, the lists are not equals() either.

13. Which of the following can fill in the blank to make the code compile? (Choose all that apply)

```
class LimpException extends Exception{}
class HurtException extends LimpException{}
public void run() {
try {
split();
}catch(HurtException e) {

} catch(_____ e) {
}
}
private void split() throws HurtException {}
```

- ✓ **A** Exception
- **B** HurtException
- **C** IOException
- **D** LimpException
- ✓ **E** RuntimeException

i Option A is correct because it is more general than HurtException. Option E is correct because any method might throw a runtime exception without declaring it. Option B is incorrect because two catch blocks with the same exception are not allowed. Option C is incorrect because the compiler knows there

is no way split() throws an IOException. Option D is incorrect because HurtException was already caught and no other LimpExceptions are thrown from split(). Java recognizes it as dead code.

14. Which are true of the following code?

```
1: public class C {
2: String seq = "c";
3: static { seq += "g"; }
4: { seq += "z"; }
5: public static void main(String[] args) {
6: C c1 = new C();
7: C c2 = new C();
8: System.out.println(c1.seq);
9: } }
```

- A The code compiles and outputs gzz.
- B The code compiles and outputs gzgz.
- C If the lines that do not compile are removed, the code outputs cg.
- ✓ D If the lines that do not compile are removed, the code outputs cz.
- E If the lines that do not compile are removed, the code outputs cgzz.
- F If the lines that do not compile are removed, the code outputs cgzz.

i The static block on line 3 does not compile because it references an instance variable. Without that line of code, each instance has its own seq variable to set to cz.

15. Which of the following fill in the blank to print out "Java"? (Choose all that apply)

```
String letters = " Java ";
String answer = _____________
System.out.println(answer);
```

- ✓ A letters.trim();
- ✓ B letters.substring(1).trim();
- C letters.trim().substring(1);
- D letters.charAt(0) + letters.substring(1).toLowerCase();
- ✓ E letters.replace(" ", "");
- F letters.replace(" J", "");

i Option A is the straightforward approach that you should use to get rid of whitespace from the beginning and end of a String. Option B removes the first character (a space) and then calls trim() on the rest of the String. Option E works as well, explicitly replacing the spaces with empty strings. Option C tries to trick you with the reverse. Order matters when chaining. Trimming first gives you "Java" and then substring() removes the first character of that String, giving you "ava". Option D takes the first character (a space) and the rest of the String in lowercase, giving you " java". Option F only gets rid of the leading space, giving you "Java ".

16. Which are true statements about the following code? (Choose all that apply)

```
public class Type {
int integer;
static int num;
public Type() {
int type;
} }
```

- ✓ **A** The code compiles.
- **B** The code does not compile.
- **C** Exactly one variable is automatically initialized.
- ✓ **D** Exactly two variables are automatically initialized.
- ✓ **E** num is a class variable.
- **F** integer is a class variable.

i The code compiles fine. *num* is a class variable, *integer* is an instance variable, and *type* is a local variable. Class and instance variables are automatically initialized to their default values.

17. What is the output of the following code?

```
1: public class Fish {
2: private static String getColor() { return "Yellow"; }
3: public static void main(String[] args) {
4: new Pufferfish().printDescription();
5: }
6: }
7: class Pufferfish extends Fish {
8: protected static String getColor() { return "Green"; }
9: public void printDescription() {
10: System.out.println(super.getColor()+","+this.getColor()+","+getColor());
11: }
12: }
```

- **A** Yellow,Green,Green
- **B** Yellow,Green,Yellow
- **C** Green,Yellow,Green
- **D** Green,Green,Green
- **E** The code will not compile because of line 8.
- ✓ **F** The code will not compile because of line 10.
- **G** The code compiles but throws an exception at runtime.

i Even though the main() method is defined within the parent Fish class, the actual call to super.getColor() is in the child class Pufferfish. Within the Pufferfish class, super.getColor() is not accessible, as the method is marked private in the parent class; therefore, the code does not compile and option F is the correct answer. If the modifier was changed from private to protected in the Fish class, then the method would be hidden by the subclass and the correct output would be Yellow,Green,Green, or option A.

18. What is the result of this code?

```
public class Chicken {
private void layEggs(int... eggs) {
System.out.print("many " + eggs[0] + " ");
}
private void layEggs(int eggs) {
System.out.print("one " + eggs + " ");
}
public static void main(String[] args) {
Chicken c = new Chicken();
c.layEggs(1, 2);
c.layEggs(3);
c.layEggs(null);
}
}
```

A many 1 one 3 one 2

B many 2 one 3 one 2

C many 1 many 3 one 2

D many 2 one 3 many 2

E The code does not compile.

✓ F An exception is thrown.

i The code outputs many 1 one 3 and then throws a NullPointerException. While Java will create an empty array if you pass 0 arguments to a vararg parameter, passing null directly still results in the eggs parameter being null. Attempting to access an index on null results in a NullPointerException.

19. What is the result of the following code?

```
1: public class PrintIntegers {
2: public static void main(String[] args) {
3: int y = -2;
4: do System.out.print(++y + " ");
5: while(y <= 5);
6: } }
```

A –2 –1 0 1 2 3 4 5

B –2 –1 0 1 2 3 4

✓ C –1 0 1 2 3 4 5 6

D –1 0 1 2 3 4 5

E The code will not compile because of line 5.

F The code contains an infinite loop and does not terminate.

i Since the pre-increment operator was used, the first value that will be displayed is –1, so options A and B are incorrect. On the second-to-last iteration of the loop, y will be incremented to 5 and the loop will output 5. The loop will continue since 5 <= 5 is true, and on the last iteration 6 will be output. At the end of this last iteration, the boolean expression 6 <= 5 will evaluate to false, and the loop will terminate. Since 6 was the last value output by the loop, the answer is option C.

20. What is printed when running the following snippet?

```
public static void main(String[] args) {
int i = 0;
try {
i += 1;
e();
i += 2;
} catch (Exception e) {
i += 4;
}
System.out.print(i);
}
private static void e() {
throws new IllegalArgumentException();
}
```

- A 1
- B 3
- C 5
- D 7
- ✓ E The code does not compile.
- F An exception is thrown.

i Method e() is incorrect. It should say throw new IllegalArgumentException() to compile. If this change were made, option C would be correct.

21. Which of the following are included in the output of this code? (Choose all that apply)

```
3: String s = new String("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. Both lines 3 and 4 call the String constructor explicitly and both are unique objects different than the literal "Hello" in the string pool. Lines 5 and 7 check for object equality, which is true, so one and three are printed. Lines 6, 8, and 9 use object reference equality, which is not true since we have different objects. s, t, and the literal "Hello" in the string pool represent three different objects.

22. What is the result of the following code?

```
1: public class MessageLogger {
2: public static void log(String s) {
3: System.out.print("s");
4: }
5: public static void log(int i) {
6: System.out.print("i");
7: }
8: public static void log(double d) {
9: System.out.print("d");
10: }
11: public static void main(String[] args) {
12: short s = 123;
13: log(s);
14: log(25.0);
15: log("hello");
16: } }
```

- **A** dds
- **B** dis
- ✓ **C** ids
- **D** iis
- **E** The code does not compile.

i The argument on line 13 is a short. It can be promoted to an int, so log on line 5 is invoked and an i is printed. The argument on line 14 is a double, so the log method on line 8 is invoked and a d is printed. The argument on line 15 is a String, so the log method on line 2 is invoked and an s is printed. Therefore, the output is ids and the answer is option C.

23. What is the result of the following code snippet? (Choose all that apply)

```
3: final int movieRating = 4;
4: int badMovie = 0;
5: switch(badMovie) {
6: case 0:
7: case badMovie: System.out.println("Awful"); break;
8: case 4:
9: case movieRating: System.out.println("Great"); break;
10: default:
11: case (int)'a':
12: case 1*1: System.out.println("Too be determined"); break;
13: }
```

- **A** The code will not compile because of line 6.
- ✓ **B** The code will not compile because of line 7.
- ✓ **C** The code will not compile because of line 8.
- ✓ **D** The code will not compile because of line 9.
- **E** The code will not compile because of line 10.
- **F** The code will not compile because of line 11.

i Clearly, there are some problems with this switch statement! First off, lines 6 and 7 both contain the same value of 0, but badMovie is not a final constant variable; therefore, it is line 7 that will not compile, so option B is correct. Note that if badMovie was made a final constant variable with the same value of 0, then option A would be considered correct too. Since it is not final, though, option A is not correct.
Next, lines 8 and 9 both have the same value of 4, and they are both constant values. Therefore, they will cause the code not to compile, so options C and D are both correct. If either of the lines were removed, then the code would compile.
Finally, options E and F, although expressed a little oddly, both evaluate to constant integer values at compile time, 97 and 1, respectively. Therefore, both are valid since the compiler can determine them, and neither option E nor option F is correct. For the OCA exam, you do not need to know how to convert char to int, but you should be aware of when a numeric cast is being done correctly.

24. Select the following statements that would allow you to assemble a valid Java program? (Choose all that apply)

A: /* hello world */
B: ArrayList list;
C: import java.util.*;
D: package abc;
E: public class Builder {
F: public void method() { }
G: }

- **A** A, D, E, B, F
- ✓ **B** A, D, C, E, F, B
- **C** D, A, E, B, F
- **D** D, A, E, F, B
- ✓ **E** D, C, A, E, B, F
- ✓ **F** D, C, A, E, F, B

i ArrayList is in the java.util package, making the import mandatory. This makes options A, C, and D incorrect since they do not contain the import. package, import, and the class definition must appear in that order. The comment can go anywhere. Within the class, the field and method can be in either order.

25. What is the result of this code? (Choose all that apply)

```
public class Change {
public static void change(int size, StringBuilder s) {
s.append("b");
size++;
}
public static void main(String[] args) {
int size = 2;
StringBuilder s1 = new StringBuilder("s1");
change(size, s1);
System.out.println("s1 = " + s1);
System.out.println("size = " + size);
}
}
```

- **A** s1 = s1
- ✓ **B** s1 = s1b
- ✓ **C** size = 2
- **D** size = 3
- **E** The code does not compile.
- **F** An exception is thrown.

i Changes to primitive method parameters are not seen by the caller, so *size* remains 2. Calling methods on method parameters is seen by the caller, so *s1* has b appended.

26. Which exception will the following throw?

```
int[] nums = new int[] { 1, 4, 6};
Object p = nums;
int[] two = (int[]) p;
System.out.println(two[two.length]);
```

✓ **A** ArrayIndexOutOfBoundsException

B ClassCastException

C IllegalArgumentException

D NumberFormatException

E None of the above

i The cast is fine because both types are the same. The last line refers to an invalid index of the array. Remember that array indexes begin with zero.

27. What is the result of the following code?

```
1: class Sparrow extends Bird { }
2: public abstract class Bird {
3: public static void main(String[] args) {
4: Bird b = new Sparrow();
5: Sparrow s = new Sparrow();
6: if (b == s) { System.out.println("=="); }
7: if (b.equals(s)) {
8: System.out.println("bird");
9: Sparrow s1 = new Sparrow();
10: }
11: if (b.equals(s1))
12: System.out.println("sparrow");
13: } }
```

A Nothing is output.

B ==

C bird

D sparrows

✓ **E** The code does not compile.

i This is one of those tricky questions that appears to ask about one thing but is really about something else. The first if statement has a scope of line 6. The second if statement has a scope of lines 7–10. The last if statement has a scope of lines 11–13. The problem is that s1 is declared in the second if statement and is not accessible after it ends. This gives a compiler error on line 11 since there is no s1 variable in scope. And yes, the exam will try to trick you with nonstandard indentation.

28. What could be the output of the following code given that e() could have any implementation?

```
public static void main(String[] args) {
System.out.print("a");
try {
System.out.print("b");
e();
}
System.out.print("c");
}
private static void e() {
// code omitted
}
```

- A a
- B ab
- C abc
- D abc followed by a stack trace
- E ab followed by a stack trace
- ✓ F None of the above; the code does not compile as is.

i A traditional try statement is required to have a catch block or a finally block.

29. What is the result of following code?

```
String s = "a";
s += 1;
s.concat(s);
s.concat(".");
System.out.println(s);
```

- A a
- ✓ B a1
- C a1a
- D a1a.
- E An exception is thrown.
- F The code does not compile.

i Remember that a String is immutable. The calls to concat() ignore the result and have no effect on s. This means the only code that is relevant here are the first two lines and a1 is output. The code does compile because s+=1 expands to s= s+1. Concatenating a String and int is allowed.

30. Which methods will compile if inserted into Joey? (Choose all that apply)

```
class CanNotHopException extends Exception {}
class Kangaroo {
public void hop() throws CanNotHopException { }
}
class Joey extends Kangaroo {
// INSERT CODE HERE
}
```

✓ **A** public void hop() {}

B public void hop() throws Exception {}

✓ **C** public void hop() throws CanNotHopException {}

✓ **D** public void hop() throws RuntimeException {}

E None of these

i Kangaroo's method declares one checked exception. Joey's method is allowed to declare that same exception or any more specific exceptions. It is also allowed to declare runtime exceptions.