Elite Quiz 14

Question 1

Correct

Mark 1.00 out of 1.00

What is the output of the following code snippet?

```
13: System.out.print("a");
14: try {
15:System.out.print("b");
16:throw new IllegalArgumentException();
17: } catch (RuntimeException e) {
18:System.out.print("c");
19: } finally {
20:System.out.print("d");
21: }
22: System.out.print("e");
```

Select one or more:

a. abe

b. abce

c. abde

[✓ d. abcde] ✓

Explanation: D. The code starts running and prints a and b on lines 13 and 15. Line 16 throws an exception, which is caught on line 17. After line 18 prints c , the finally block is run
and d is printed. Then the try statement ends and e is printed on line 22.

e. The code does not compile.

f. An uncaught exception is thrown.

Your answer is correct.

The correct answer is: abcde

Question 2

Correct

Mark 1.00 out of 1.00

What is the output of the following code?

```
1: public class Deer {
2: public Deer() { System.out.print("Deer"); }
3: public Deer(int age) { System.out.print("DeerAge"); }
4: private boolean hasHorns() { return false; }
5: public static void main(String[] args) {
6: Deer deer = new Reindeer(5);
7: System.out.println(","+deer.hasHorns());
8: }
9: }
10: class Reindeer extends Deer {
11: public Reindeer(int age) { System.out.print("Reindeer"); }
12: public boolean hasHorns() { return true; }
13: }
```

Select one or more:

[✓ a. DeerReindeer,false] ✓

Explanation: A. The code compiles and runs without issue, so options G and H are incorrect. First, the Reindeer object is instantiated using the constructor that takes an int value. Since there is no explicit call to the parent constructor, the default no-argument super() is inserted as the first line of the constructor. The output is then Deer , followed by
Reindeer in the child constructor, so only options A and B can be correct. Next,
the method hasHorns() looks like an overridden method, but it is actually a hidden
method since it is declared private in the parent class. Because the hidden method is
referenced in the parent class, the parent version is used, so the code outputs false ,
and option A is the correct answer.

b. DeerReindeer,true

c. ReindeerDeer,false

d. ReindeerDeer,true

e. DeerAgeReindeer,false

f. DeerAgeReindeer,true

g. The code will not compile because of line 7.

h. The code will not compile because of line 12.

Your answer is correct.

The correct answer is: DeerReindeer,false

Question 3

Correct

Mark 1.00 out of 1.00

What is the output of the following program?

```
1: public class BearOrShark {
2:public static void main(String[] args) {
3:int luck = 10;
4:if((luck>10 ? luck++: --luck)<10) {
5:System.out.print("Bear");
6:} if(luck<10) System.out.print("Shark");
7: } }
```

Select one or more:

- ☐ a. Bear
- ☐ b. Shark
- [✓ c. BearShark] ✓

Explanation : C. The code compiles and runs without issue, so options D and E are correct.
Remember that only one of the right-hand ternary expressions will be evaluated at runtime.
Since luck is not less than 10 , the second expression, --luck , will be evaluated, and
since the pre-increment operator was used, the value returned will be 9 , which is lessthan 10 . So the first if-then statement will be visited and Bear will be output. Notice
there is no else statement on line 6. Since luck is still less than 10 , the second if-then
statement will also be reached and Shark will be output; therefore, the correct answer
is option C.

- ☐ d. The code will not compile because of line 4.
- ☐ e. The code will not compile because of line 6.
- ☐ f. The code compiles without issue but does not produce any output.

Your answer is correct.

The correct answer is: BearShark

Question 4

Correct

Mark 1.00 out of 1.00

Which of the following statements can be inserted in the blank line so that the code will compile successfully? (Choose all that apply)

```
public interface CanSwim {}
public class Amphibian implements CanSwim {}
class Tadpole extends Amphibian {}
public class FindAllTadPole {
public static void main(String[] args) {
List<Tadpole> tadpoles = new ArrayList<Tadpole>();
for(Amphibian amphibian : tadpoles) {
___________ tadpole = amphibian;
} } }
```

Select one or more:

- [✓ a. CanSwim] ✓
- ☐ b. Long
- [✓ c. Amphibian] ✓
- ☐ d. Tadpole
- [✓ e. Object] ✓

Explanation: A, C, E. The for-each loop automatically casts each Tadpole object to an
Amphibian reference, which does not require an explicit cast because Tadpole is a subclass of Amphibian . From there, any parent class or interface that Amphibian inherits from is
permitted without an explicit cast. This includes CanSwim , the interface Amphibian
implements, and Object , which all classes extend from, so options A and E are correct.
Option C is also correct since the reference is being cast to the same type, so no explicit
cast is required. Option B is incorrect, since Long is not a parent of Amphibian . Option
D is incorrect as well, although an explicit cast to Tadpole on the right-hand side of the
expression would be required to allow the code to compile.

Your answer is correct.

The correct answer is: CanSwim, Amphibian, Object

Question **5**
Correct
Mark 1.00 out of 1.00

Which of the following are checked exceptions? (Choose all that apply)

Select one or more:
- [✓ a. Exception] ✓
- b. IllegalArgumentException
- [✓ c. IOException] ✓

Explanation: A, C. Option A is the exception base class, which is a checked exception. Options B, D, and E extend RuntimeException directly or indirectly and therefore are unchecked exceptions. Option F is a throwable and not an exception, and so should not be caught or declared.

- d. Tadpole
- e. Object

Your answer is correct.
The correct answer is: Exception, IOException

Question **6**
Correct
Mark 1.00 out of 1.00

What is the output of the following application?

```
1: public class CompareValues {
2:public static void main(String[] args) {
3:int x = 0;
4:while(x++ < 10) {}
5:String message = x > 10 ? "Greater than" : false;
6:System.out.println(message+","+x);
7:}
8: }
```

Select one or more:
- a. Greater than,10
- b. false,10
- c. Greater than,11
- d. false,11
- e. The code will not compile because of line 4.
- [✓ f. The code will not compile because of line 5.] ✓

Explanation: F. In this example, the ternary operator has two expressions, one of them a String and the other a boolean value. The ternary operator is permitted to have expressions that don't have matching types, but the key here is the assignment to the String reference.
The compiler knows how to assign the first expression value as a String , but the second boolean expression cannot be set as a String ; therefore, this line will not compile.

Your answer is correct.
The correct answer is: The code will not compile because of line 5.

Question 7

Correct

Mark 1.00 out of 1.00

What change would allow the following code snippet to compile? (Choose all that apply)

```
3: long x = 10;
4: int y = 2 * x;
```

Select one or more:

- a. No change; it compiles as is.
- [✓ b. Cast x on line 4 to int.] ✓
- [✓ c. Change the data type of x on line 3 to short.] ✓
- [✓ d. Cast 2 * x on line 4 to int.] ✓
- e. Change the data type of y on line 4 to short.
- [✓ f. Change the data type of y on line 4 to long.] ✓

Explanation: B, C, D, F. The code will not compile as is, so option A is not correct. The value 2 *x is automatically promoted to long and cannot be automatically stored in y , which is in an int value. Options B, C, and D solve this problem by reducing the long value to
int . Option E does not solve the problem and actually makes it worse by attempting
to place the value in a smaller data type. Option F solves the problem by increasing the
data type of the assignment so that long is allowed.

Your answer is correct.

The correct answer is: Cast x on line 4 to int., Change the data type of x on line 3 to short., Cast 2 * x on line 4 to int., Change the data type of y on line 4 to long.

Question 8

Correct

Mark 1.00 out of 1.00

What is the output of the following code snippet?

```
3: int x = 4;
4: long y = x * 4 - x++;
5: if(y<10) System.out.println("Too Low");
6: else System.out.println("Just right");
7: else System.out.println("Too High");
```

Select one or more:

- a. Too Low
- b. Just Right
- c. Too High
- d. Compiles but throws a NullPointerException.
- e. The code will not compile because of line 6.
- [✓ f. The code will not compile because of line 7.] ✓

Explanation: F. The code does not compile because two else statements cannot be chained together without additional if-then statements, so the correct answer is option F. Option E is incorrect as Line 6 by itself does not cause a problem, only when it is paired with Line 7. One way to fix this code so it compiles would be to add an if-then statement on
line 6. The other solution would be to remove line 7.

Your answer is correct.

The correct answer is: The code will not compile because of line 7.

Question **9**

Correct

Mark 1.00 out of 1.00

What is the output of the following code?
1: public class TernaryTester {
2:public static void main(String[] args) {
3:int x = 5;
4:System.out.println(x > 2 ? x < 4 ? 10 : 8 : 7);
5: }}

Select one or more:

- a. 5
- b. 4
- c. 10
- [✓ d. 8] ✓

Explanation: D. As you learned in the section "Ternary Operator," although parentheses are not required, they do greatly increase code readability, such as the following equivalent
statement: System.out.println((x > 2) ? ((x < 4) ? 10 : 8) : 7)
We apply the outside ternary operator fi rst, as it is possible the inner ternary expression
may never be evaluated. Since (x>2) is true , this reduces the problem to:
System.out.println((x < 4) ? 10 : 8)
Since x is greater than 2 , the answer is 8 , or option D in this case.

- e. 7
- f. The code will not compile because of line 4.

Your answer is correct.

The correct answer is: 8

Question **10**

Correct

Mark 1.00 out of 1.00

What is the output of the following code snippet?
3: boolean x = true, z = true;
4: int y = 20;
5: x = (y != 10) ^ (z=false);
6: System.out.println(x+", "+y+", "+z);

Select one or more:

- a. true, 10, true
- [✓ b. true, 20, false] ✓

Explanation: B. This example is tricky because of the second assignment operator embedded in line 5. The expression (z=false) assigns the value false to z and returns false for the entire expression. Since y does not equal 10 , the left-hand side returns true ; therefore, the exclusive or (^) of the entire expression assigned to x is true . The output reflects these assignments, with no change to y , so option B is the only correct answer. The
code compiles and runs without issue, so option F is not correct.

- c. false, 20, true
- d. false, 20, false
- e. false, 20, true
- f. The code will not compile because of line 5.

Your answer is correct.

The correct answer is: true, 20, false

Question 11

Correct

Mark 1.00 out of 1.00

What is the output of the following code snippet?

```
3: int x = 0;
4: String s = null;
5: if(x == s) System.out.println("Success");
6: else System.out.println("Failure");
```

Select one or more:

- [ ] a. Success
- [ ] b. Failure
- [ ] c. The code will not compile because of line 4.
- [✓] d. The code will not compile because of line 5. ✓

Explanation: D. The variable x is an int and s is a reference to a String object. The two data types are incomparable because neither variable can be converted to the other variable’s type. The compiler error occurs on line 5 when the comparison is attempted, so the answer is option D.

Your answer is correct.

The correct answer is: The code will not compile because of line 5.

Question 12

Correct

Mark 1.00 out of 1.00

What is the output of the following code snippet?

```
3: int x1 = 50, x2 = 75;
4: boolean b = x1 >= x2;
5: if(b = true) System.out.println("Success");
6: else System.out.println("Failure");
```

Select one or more:

- [✓] a. Success ✓

Explanation: A. The code compiles successfully, so options C and D are incorrect. The value of b after line 4 is false . However, the if-then statement on line 5 contains an assignment, not a comparison. The variable b is assigned true on line 3, and the assignment operator returns true , so line 5 executes and displays Success , so the answer is option A.

- [ ] b. Failure
- [ ] c. The code will not compile because of line 4.
- [ ] d. The code will not compile because of line 5.

Your answer is correct.

The correct answer is: Success

Question **13**

Correct

Mark 1.00 out of 1.00

What is the output of the following code snippet?

```
3: boolean keepGoing = true;
4: int result = 15, i = 10;
5: do {
6: i--;
7: if(i==8) keepGoing = false;
8: result -= 2;
9: } while(keepGoing);
10: System.out.println(result);
```

Select one or more:

a. 7

b. 9

c. 10

[✓ d. 11] ✓

Explanation: D. The code compiles without issue, so option F is incorrect. After the first execution of the loop, i is decremented to 9 and result to 13 . Since i is not 8, keepGoing is false , and the loop continues. On the next iteration, i is decremented to 8 and result to 11 . On the second execution, i does equal 8 , so keepGoing is set to false . At the conclusion of the loop, the loop terminates since keepGoing is no longer true . The value of result is 11 , and the correct answer is option D.

e. The code will not compile because of line 8.

Your answer is correct.

The correct answer is: 11

Question **14**

Correct

Mark 1.00 out of 1.00

What is the output of the following code snippet?

```
3: int count = 0;
4: ROW_LOOP: for(int row = 1; row <=3; row++)
5:for(int col = 1; col <=2 ; col++) {
6:if(row * col % 2 == 0) continue ROW_LOOP;
7:count++;
8:}
9: System.out.println(count);
```

Select one or more:

[✓ a. 1] ✓

Explanation: A. The expression on line 5 is true when row * col is an even number. On the first iteration, row = 1 and col = 1 , so the expression on line 6 is false , the continue is skipped, and count is incremented to 1 . On the second iteration, row = 1 and col = 2 , so the expression on line 6 is true and the continue ends the outer loop with count still at 1 . On the third iteration, row = 2 and col = 1 , so the expression on line 6 is true and the continue ends the outer loop with count still at 1 . On the fourth iteration, row = 3 and col = 1 , so the expression on line 6 is false , the continue is skipped, and count is incremented to 2 . Finally, on the fifth and final iteration, row = 3 and col = 2 , so the expression on line 6 is true and the continue ends the outer loop with count still at 2 . The result of 2 is displayed, so the answer is option B.

b. 2

c. 3

d. 4

e. The code will not compile because of line 6.

Your answer is correct.

The correct answer is: 1

Question 15

Correct

Mark 1.00 out of 1.00

What is the result of the following code snippet?

```
3: final char a = 'A', d = 'D';
4: char grade = 'B';
5: switch(grade) {
6: case a:
7: case 'B': System.out.print("great");
8: case 'C': System.out.print("good"); break;
9: case d:
10: case 'F': System.out.print("not good");
11: }
```

Select one or more:

- [ ] a. great
- [✓ b. greatgood] ✓

Explanation: B. The code compiles and runs without issue, so options C, D, and E are not
correct. The value of grade is 'B' and there is a matching case statement that will cause "great" to be printed. There is no break statement after the case , though, so the next case statement will be reached, and "good" will be printed. There is a break after this case state-
ment, though, so the switch statement will end. The correct answer is thus option B.

- [ ] c. The code will not compile because of line 3.
- [ ] d. The code will not compile because of line 6.
- [ ] e. The code will not compile because of lines 6 and 9.

Your answer is correct.

The correct answer is: greatgood