socrative

Answer Key
February 24, 2021

OCA Bonus Test 3 (pt 1 of 2) ('Copy')

1. Which of the following lambda expressions can be passed to a function of Predicate<String> type? (Choose all that apply)

- [✓ **A**] check(s → s.isEmpty());
- **B** check(s -→ s.isEmpty());
- [✓ **C**] check((String s) → s.isEmpty());
- **D** check((String s) -→ s.isEmpty());
- **E** check((StringBuilder s) → s.isEmpty());
- **F** check((StringBuilder s) -→ s.isEmpty());

i Predicate<String> takes a parameter list of one parameter using the specified type. Option E is incorrect because it specifies the wrong type. Options B, D, and F are incorrect because they use the wrong syntax for the arrow operator.

2. Which of the following pairs compiles and runs a Java program? (Choose all that apply)
A.javac Zoo

- **A** javac Zoo
- **B** javac Zoo.class
- [✓ **C**] javac Zoo.java
- [✓ **D**] java Zoo
- **E** java Zoo.class
- **F** java Zoo.java

i Option C is correct because compile code requires passing a Java class. Option D is correct because running a Java class omits the .class extension.

3. What is the result of the following code snippet?

```
3: int x = 10 % 2 + 1;
4: switch(x) {
5: case: 0 System.out.print("Too High"); break;
6: case: 1 System.out.print("Just Right"); break;
7: default: System.out.print("Too Low");
8: }
```

- A Too High
- B Just Right
- C Too Low
- D JustRightTooLow
- E The code will not compile because of line 3.
- [✓ F] The code will not compile because of lines 5 and 6.

This question is designed to test your ability to spot syntax errors with switch statements. In particular, the colon (:) goes after the value in the case statement, not before; therefore, neither line 5 nor line 6 will compile, and option F is the correct answer. If the colon was moved after the values, the output would be Just Right and the answer would be option B.

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

- A ArithmeticException
- B ClassCastException
- [✓ C] java.io.IOException
- D NoClassDefFoundError
- E StackOverflowError

A checked exception is a class that extends Exception, but not RuntimeException. Option C is the only checked exception in the list. Options A and B are runtime exceptions. Options D and E are errors.

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

- [✓ A] List<Integer> l1 = new ArrayList();
- [✓ B] List<Integer> l2 = new ArrayList<>();
- [✓ C] List<Integer> l3 = new ArrayList<Integer>();
- D List<> l4 = new ArrayList<Integer>();
- E List<Integer> l5 = new List<Integer>();
- F ArrayList<int> l6 = new List<int>();

Option A compiles since it is allowed to use generics on just one side in a declaration. Option B compiles using the diamond operator. Option C is a longer form of option B; it spells out the generics type. Option D does not compile because the diamond operator is only allowed on the right side. Option E does not compile because List is only allowed on the left side since it's an interface rather than a concrete type. Option F does not compile because primitives are not allowed to be ArrayList types. Autoboxing only works when working with the ArrayList, not when creating it.

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

[✓ A] public void swim(int... distance) {}

B public void swim(int... distance, int time) {}

[✓ C] public void swim(int time, int... distance) {}

D public void swim(int... distance, int,,, time) {}

E public void swim(int time, ...int distance) {}

i The varargs parameter must be the last parameter in the parameter list. The ellipsis (...) must go after the type that will be made into a vararg.

7. What is the result of the following class?

```
1: import java.util.function.*;
2:
3: public class Panda {
4: int age;
5: public static void main(String[] args) {
6: Panda p1 = new Panda();
7: p1.age = 1;
8: check(p1, p → {p.age < 5});
9: }
10: private static void check(Panda panda, Predicate<Panda> pred) {
11: String result = pred.test(panda) ? "match" : "not match";
12: System.out.print(result);
13: } }
```

A match

B not match

[✓ C] Compiler error on line 8

D Compiler error on line 10

E Compile error on line 11

F A runtime exception is thrown.

i Line 8 uses braces around the body. This means the return keyword and semicolon are required.

8. Given the following code, which statements are true? (Choose all that apply)

```
import static java.lang.System.out;
class Flower { }
public class Bee {
void pollinate(Flower flower) { out.println(1); }
void pollinate(Flower... flower) { out.println(2); }
void pollinate(int... numFlowers) { out.println(3); }
void pollinate(Integer numFlowers) { out.println(4); }
void pollinate(String s) { out.println(5); }
void pollinate(Object obj) { out.println(6); }
}
```

✓ **A** Calling pollinate(new Flower()) prints 1.

B Calling pollinate(new Flower()) prints 2.

C Calling pollinate(1)) prints 3.

✓ **D** Calling pollinate(1)) prints 4.

✓ **E** Calling pollinate("flower")) prints 5.

F Calling pollinate("flower")) prints 6.

i Java prefers the most specific overloaded signature it can find. It prefers a single object over a vararg parameter, making option A correct. It prefers an autoboxed parameter over a vararg parameter, making option D correct. It prefers a subclass over a superclass, making option E correct.

9. Which exception will the following throw?

```
int[] nums = new int[] { 1, 0, 2 };
Object p = nums;
int[] two = (int[]) p;
System.out.println(10 / two[2]);
```

A ArrayIndexOutOfBoundsException

B ClassCastException

C IllegalArgumentException

D NumberFormatException

✓ **E** None of the above

i There is nothing wrong with this code. It outputs 5. We've included this question as a reminder that just because you are asked about exceptions doesn't mean one is thrown.

10. Which of these lines compile? (Choose all that apply)

```
1: public class Rich {
2: public void money() {
3: int _million = 1_000_000;
4: double aThousand = 1_000_.00;
5: double 100 = 100;
6: int hundred = 100.00;
7: float ten = 10d;
8: short one = 1;
9: } }
```

[✓ **A**] 3

B 4

C 5

D 6

E 7

[✓ **F**] 8

i Option A is correct. Identifiers are allowed to begin with underscores. Numeric values are allowed to have underscores between two digits. Option B is incorrect because underscores are not allowed adjacent to a decimal point. Option C is incorrect because identifiers are not allowed to start with digits. Option D is incorrect because decimal values are not allowed in int variables. They can only go in float or double types. Option E is incorrect because 10d declares a double and a double is bigger than a float. Option F is correct because it is a straightforward variable assignment.

11. What is the result of the following code?

```
1: public class ModSample {
2: public static void main(String[] args) {
3: int y = 4;
4: int x = 10 - ++y / 5;
5: System.out.println(x % y);
6: } }
```

[✓ **A**] 4

B 0

C 7

D 5

E The answer cannot be determined until runtime.

F The code will not compile because of line 4.

i The code compiles and runs without issue, so options E and F are incorrect. In the order of operation, the first operator applied is the pre-increment operator, resulting in y being assigned a value of 5, and the value returned from the operator assigned the new value of 5. Next, we apply the division operator since division has a higher order of precedence than subtraction, resulting in an expression of x = 10 - 5 /5 = 10 - 1 = 9. Finally, we perform modular arithmetic using the new values of x and y—9 and 5, respectively. If we divide these two values we get 0; therefore, the remainder of 9 % 5 is 4, so option A is the correct answer.

12. What is the output when new PrayingMantis("green") is called?

```
1: abstract class Insect {
2: public Insect(int age) { System.out.println("1"); }
3: public Insect(String color) { this(5); System.out.println("2"); }
4: }
5: public class PrayingMantis extends Insect {
6: public PrayingMantis(String color) {
7: System.out.println("3");
8: }
9: }
```

A 123

B 13

C The code will not compile because of line 1.

D The code will not compile because of line 3.

[✓ E] The code will not compile because of line 7.

F The code compiles but throws an exception at runtime.

i The code does not compile properly, so options A, B, and F are incorrect. The compilation error is based on the fact that Insect does not define a no-argument constructor; therefore, its subclasses must explicitly call the parent constructor with the super() command. Since the first line of the constructor in PrayingMantis is not a call to this()or super(), the compiler will attempt to automatically insert a call to the default no-argument constructor super(). Since there is none, the class will not compile, and option E is the correct answer. Lines 1 and 3 are implemented correctly and compile without issue, so options C and D are incorrect.

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

```
1: public class Dino {
2: static final String species;
3: double weight;
4: { species = "Raptor"; }
5: public Dino(double weight) {
6: this.weight = weight;
7: }
8: public static void main(String[] args) {
9: Dino dino = new Dino(500);
10: System.out.println(dino.weight);
11: } }
```

[✓ A] Compiler error on line 2

[✓ B] Compiler error on line 4

C Compiler error on line 9

D Compiler error on line 10

E 500

i *species* is final and static, which means it must be set exactly once. It must be set in the line it is declared on or in a static initializer. Since it is in an instance initializer, line 4 gives a compiler error. Instance initializers are run each time the object is constructed. However, a static constant can only be set once. Line 2 also gives a compiler error because the final variable is not set at all.

14.

What does the following output?

```
List numberList = Arrays.asList(5, 10, -5, -10);
Collections.sort(numberList);
int five = Collections.binarySearch(numberList, 5);
int four = Collections.binarySearch(numberList, 4);
System.out.println(five + four);
```

- **A** -2
- ✓ **B** -1
- **C** 0
- **D** 1
- **E** 2
- **F** An exception is thrown.
- **G** The code doesn't compile.

i The list is sorted as [–10, –5, 5, 10]. The index of 5 is 2. Since 4 isn't found, we consider the index where it would go, which is 2. Per the rule, we negate that and subtract 1, giving us –2–1, or –3. Adding the two results of 2 and –3, we get –1 as the answer.

15. How many checked exceptions are declared in this code?

```
class One extends Exception { }
class Two extends One { }
class Three extends Error { }
class Four extends RuntimeException { }
class Five extends Four { }
```

- **A** 1
- ✓ **B** 2
- **C** 3
- **D** 4
- **E** 5

i Options A and B are checked exceptions because they extend Exception, but not RuntimeException. Option C is an error. Options D and E are unchecked (runtime) exceptions.

16. What is the result of compiling the following code? (Choose all that apply)

```
1: abstract class Mammal {
2: public abstract int getAge();
3: }
4: abstract class Animal {
5: public int getAge();
6: }
7: abstract interface Herbivore {}
8: public class HoneyBadger extends Mammal, Animal implements Herbivore {
9: int getAge() { return 5; }
10: }
```

- A The code compiles without issue.
- B The code will not compile because of line 2.
- [✓ C] The code will not compile because of line 5.
- [✓ D] The code will not compile because of line 8.
- [✓ E] The code will not compile because of line 9.
- F The code compiles but throws an exception at runtime.

i The code does not compile properly, so options A and F are incorrect. The declaration of the abstract method on line 2 compiles without issue, so option B is not correct. The method getAge() is not marked abstract, so it must provide a method body. Since it does not provide a method body, it does not compile and option C is correct. Line 8 will throw a compiler error, since HoneyBadger cannot extend two classes. A class may only extend one other class, although it can implement multiple interfaces; therefore, option D is correct. Finally, line 9 will also throw a compiler error as the default access modifier is more restrictive than the public access modifier declared in either parent class.

17. Which of the following are ordered from smallest to largest? (Choose all that apply)

- A byte, int, short, long
- [✓ B] byte, short, int, long
- C short, byte, int, long
- D short, int, byte, long
- E double, float
- [✓ F] float, double

i A byte is 8 bits. A short is 16 bits. An int is 32 bits. A long is 64 bits. A double is twice as big as a float. We're sorry that this is memorization, but you really do need to know the relative sizes.

18. What is the output of the following code snippet?

```
3: int j = 1;
4: for(int i=0; i<4 && j<3; ++i) {
5: ++j;
6: System.out.print(i+j);
7: }
```

A 0213

[✓ B] 24

C 6

D 13

E The code will not compile because of line 4.

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

The code compiles and runs without issue, so options E and F are incorrect. During the first iteration of the expression j is incremented to 2, so i + j evaluates to 0 + 2; therefore, 2 is output. At the end of the first execution of the loop, i is updated to 1. On the second execution of the loop, the boolean expression still evaluates to true so the loop continues. Now, j is updated to 3, so line 6 becomes 1 + 3, which outputs 4. At the end of the second execution of the loop, i is updated to 2. At the beginning of the third execution of the loop, j < 3 is false, as j is 3; therefore, the loop terminates since the entire boolean expression is false. Because the output was 24, the correct answer is option B. Note that option A would have been correct if the print statement, rather than adding the arithmetic values, had performed String concatenation, such as System.out.print(i+""+j);.

19. Which of the following statements about objects and references are true in Java? (Choose all that apply)

[✓ A] By explicitly casting an object to a subclass, you can gain access to methods and variables that were hidden from access.

B If you compare two distinct references to the same object with ==, the result will evaluate to false.

[✓ C] All objects in memory can be referenced using a reference of type java.lang.Object.

D When you create an object in Java, you get direct access to the object in memory.

E Removing all references to an object deletes the object from memory.

It is true that casting an object to a subclass may grant access to new methods and variables that were not previously available in the superclass reference, so option A is correct. Option B is incorrect—comparing two references with == that point to the same object results in a true value. All objects inherit from java.lang.Object, so all objects in memory can be access with a reference of that type, so option C is true. All objects are accessed via a reference in Java, so option D is incorrect since you can never get direct access. Finally, removing all references to an object makes the object eligible for garbage collection, but the rules of garbage collection do not guarantee when this deletion from memory will occur.

20. What is true of the following code?

```
1: public class Deer {
2: private void freeze() { stayStill(); }
3: public static void inTheHeadlights() { freeze(); }
4: private static void stayStill() { }
5: }
6: public class Car {
7: public static void drive(Deer deer) {
8: deer.inTheHeadlights();
9: } }
```

A Line 2 does not compile.

✓ **B** Line 3 does not compile.

C Line 4 does not compile.

D Line 8 does not compile.

E The code compiles and runs without exception if Car.drive(null) is called.

F The code compiles but throws an exception if Car.drive(null) is called.

A static method is not allowed to call an instance method. Line 3 attempts to do this. If the code were to compile, it would not throw an exception when Car.drive(null) is called. Since inTheHeadlights() is static, an object is not needed to call it. Java just looks at the type of the reference, which is Deer.

21. Which of the following are true statements about interface methods? (Choose all that apply)

✓ **A** They can be declared abstract, static, or default.

B They are assumed to be protected.

✓ **C** They can be overridden with an abstract method in an abstract class that implements the interface.

✓ **D** A class can implement two interfaces that declare the same abstract method.

E A class can implement two interfaces that declare the same default method without overriding it.

✓ **F** If an interface method is marked abstract, then it cannot be marked static or default.

An interface may only be marked at most one of the following: abstract, static, or default, so options A and F are correct. Option B is incorrect since interface methods are assumed to be public. Option C is correct since an interface method can be overridden with an abstract method that follows the appropriate rules for method overriding. A class that implements two interfaces can have duplicate abstract and static methods, so option D is correct. A class that implements two interfaces can only have a duplicate default method if the class overrides the default method; otherwise, a compiler error will be created, so option E is incorrect.

22. Given the following code, which of the statements are true? (Choose all that apply)

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

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

- [✓ A] If Tadpole is in package animal, t.ribbit() will compile if given default access.
- [✓ B] If Tadpole is in package animal, t.ribbit() will compile if given protected access.
- C If Tadpole is in package animal, t.ribbit() will compile if given private access.
- D If Tadpole is in package baby, t.ribbit() will compile if given default access.
- [✓ E] If Tadpole is in package baby, t.ribbit() will compile if given protected access.
- F If Tadpole is in package baby, t.ribbit() will compile if given private access.

i If the two classes are in the same package, they will compile if ribbit() has public, protected, or default access. If the two classes are not in the same package, they will compile if ribbit() has public or protected access.

23. Given the following definitions, which of the following could be added to collection of type List<Whale>? (Choose all that apply)

```
public interface HasTail {};
public class Mammal {}
public class Whale extends Mammal implements HasTail {}
public class Narwhal extends Whale {}
```

- A new HasTail()
- B new Mammal()
- [✓ C] new Whale()
- [✓ D] new Narwhal()
- E new Object()
- [✓ F] null

i Option A is incorrect, since HasTail is an abstract interface and cannot be instantiated directly. Option B is incorrect because Mammal is a superclass of Whale, whereas polymorphism only allows subclasses of Whale to be used. Options C and D are correct because both inherit from Whale. Option E is incorrect for the same reason that option B is incorrect. Finally, option F is correct, since null can be passed for any object.

24. What is the output of the following code?

```
LocalDateTime d = LocalDateTime.of(2015, 5, 10, 11, 22, 33);
DateTimeFormatter f = DateTimeFormatter.ofPattern("hh:MM");
System.out.println(d.format(f));
```

✓ **A** 11:05

B 11:22

C 22:05

D 22:33

E The code does not compile.

F A runtime exception is thrown.

i hh is for hour. MM is for month. The code is trying to confuse you by using hh:MM rather than hh:mm.

25. What is the result of this code?

```
class Toy{
private boolean containsIce = false;
public boolean containsIce() {
return containsIce;
}
public void removeIce() {
this.containsIce = true;
}
}
public class Otter {
private static void play(Toy toy) {
toy = new Toy();
toy.removeIce();
}
public static void main(String[] args) {
Toy toy = new Toy();
Otter.play(toy);
System.out.println(toy.containsIce());
}
}
```

✓ **A** false

B true

C There is one compiler error.

D There are two compiler errors.

E There are three compiler errors.

F An exception is thrown.

i Since Java is pass by value, reassignments to method parameters are not seen by the caller.

26. What is the result of the following code?

```
4: String numbers = "2468";
5: int total = 0;
6: total += numbers.indexOf("6");
7: total += numbers.length();
8: char ch = numbers.charAt(4);
9: System.out.println(total + " " + ch);
```

- A 5 8
- B 6 8
- C 7 8
- D 8 8
- ✓ E An exception is thrown.
- F The code does not compile.

i Line 6 returns the number 2 since indexes are zero based. Line 7 returns 4, the number of characters in the String. This makes total equal to 6. Line 8 tries to returns the character at index 4. However, there is no character at that position since the highest index is 3 and Java throws a StringIndexOutOfBoundsException.

27. Which of the following lists is a correct sequence for catching exceptions? (Choose all that apply)

- ✓ A NumberFormatException, ClassCastException, RuntimeException
- B NumberFormatException, RuntimeException, ClassCastException
- C RuntimeException, NumberFormatException, ClassCastException
- D RuntimeException, NumberFormatException, Exception
- E NumberFormatException, Exception, RuntimeException
- ✓ F NumberFormatException, RuntimeException, Exception

i The most specific exceptions must be listed first. RuntimeException extends Exception. The others listed in this question extend RuntimeException.

28. Given the following two classes, which of the following statements will compile when inserted on line 2? (Choose all that apply)

```
1: package lilypad;
2: public class Frog { }

1: package lilypad;
2: // INSERT CODE HERE
3: public class Tadpole {
4: private Frog parent;
5: }
```

A import Frog;

[✓ B] import lilypad.Frog;

[✓ C] import lilypad.Tadpole;

D import static lilypad.Tadpole;

E static import lilypad.Tadpole;

[✓ F] The code compiles without any code additions.

i Since both Frog and Tadpole are in the same package, no import at all is needed, making option F correct. Option B is also correct because importing redundant classes is allowed. Option A is not correct because imports must include the full package name. Option C is correct because it is a valid import statement—albeit one that is ignored since we are already in the Tadpole class. Options D and E are incorrect because static imports must import members and not the classname.

29. Which are the minimum changes must be made to the following code for it to properly exhibit encapsulation? (Choose all that apply)

```
1: public class HoneyPot {
2: Honey honey;
3: public Honey getHoney() { return honey; }
4: public void setHoney(Honey h) { honey = h; }
5: public void depositHoney(Bee bee) {
6: honey.add(bee.getHoney());
7: }
8: }
```

[✓ A] Make the variable on line 2 private.

B Make the variable on line 2 public.

C Make the method on line 4 private.

D Make the method on line 5 private.

E Remove line 4.

F Remove lines 5–7.

i Encapsulation is usually implemented with private instance variables and public methods. Calls are allowed to make changes to the instance variables as long as they do so through methods. If the question asked about immutability, the two methods that set/change honey would need to be removed or made private.

30. Given this array declaration, which of the following statements are valid array indexes?
int[][] numbers = {{1,2}, {3}, {4,5,6}};

✓ **A** numbers[0][1]

B numbers[0][2]

✓ **C** numbers[1][0]

D numbers[1][1]

✓ **E** numbers[2][0]

F numbers[3][1]

i Option A is valid and returns 2. Option B is invalid because the array at index 0 is only two elements long. Option C is valid and returns 3. Option D is invalid because the array at index 1 is only one element long. Option E is valid and prints 4. Option F is invalid because there is no array at index 3.