socrative

Answer Key
February 24, 2021

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

1. Suppose we have the following class named ShowDate. Which one of the following statements is true?

```
1: import java.util.Date;
2: public class ShowDate {
3: public static void main(String [] args) {
4: Date a = new Date();
5: Date b = new Date();
6: Date c = a;
7: System.out.println(c.toString());
8: a = null;
9: c = null;
10: } }
```

A The Date object from line 4 is eligible for garbage collection immediately following line 6.

B The Date object from line 4 is eligible for garbage collection immediately following line 8.

✓ C The Date object from line 4 is eligible for garbage collection immediately following line 9.

D The code does not compile.

E The code throws an exception.

i The Date object from line 5 has two references to it: a and c. It becomes eligible for garbage collection after line 10, when both a and c no longer point to the object. Therefore, the answer is option C.

2. What is the result of this code?

```
1: public class Dino {
2: final String species = "Triceratops";
3: double weight;
4: public Dino(double weight) {
5: this.weight = weight;
6: species = "Raptor";
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 5
- ✓ B Compiler error on line 6
- C Compiler error on line 9
- D Compiler error on line 10
- E 500

i *species* is final, which means it must be set exactly once. It must be set in the line it is declared on, in an instance initializer, or in the constructor. Since it is set twice, the second one gives a compiler error.

3. What is the result of the following code?

```
public class A {
private int i = 6;
private int j = i;
public void A() {
i = 5;
}
public static void main(String[] args) {
A a = new A();
System.out.println(a.i + a.j);
} }
```

- A 55
- B 56
- C 66
- D 10
- E 11
- ✓ F 12
- G The code does not compile.

i This example has a method that looks similar to a constructor but is not one. The void return type is the giveaway that we are dealing with a method. i and j both get initialized to 6. Since both numbers are integers, Java adds them rather than concatenating them.

4. What is the output of the following program?

```
1: public class Image {
2: public int width, height;
3: public void showImage() throws Exception {
4: if(width < 0 || height < 0)
5: throw new Exception("invalid image size");
6: else
7: System.out.print("1");
8: }
9: public static void main(String [] args) {
10: Image image = new Image();
11: image.width = -10;
12: try {
13: image.showImage();
14: System.out.print("2");
15: }catch(Exception e) {
16: System.out.print("3");
17: }
18: System.out.print("4");
19: }
20: }
```

- **A** 124
- **B** 234
- ✓ **C** 34
- **D** 3 and a stack trace for Exception
- **E** 34 and a stack trace for Exception
- **F** The code does not compile.

i The showImage() method is invoked on line 13 on an Image object with a negative *width*. This causes an Exception to be thrown on line 5 and caught on line 15. 3 is printed on line 16 and then 4 is printed on line 18. No stack trace is dumped because the exception was handled.

5. Which of the following are correct differences between abstract class and interfaces? (Choose all that apply)

- **A** Abstract classes can declare static methods, but interfaces cannot.
- **B** Interfaces can declare public static final variables, but abstract classes cannot.
- ✓ **C** An interface may extend another interface but not an abstract class.
- ✓ **D** Interfaces support multiple inheritance, whereas abstract classes support single inheritance.
- **E** An abstract class can be declared in a separate file, whereas an interface must be bundled within an existing class file.
- **F** Only abstract classes can include abstract methods.

i As of Java 8, an interface can include static methods, so option A is incorrect. Both interfaces and abstract classes can declare public static final variables, so option B is incorrect. Interfaces can only extend other interfaces, whereas classes can only extend other classes, so option C is correct. A class

may implement multiple interfaces but only extend one class, so option D is correct. Interfaces can be in their own file with no class defined, so option E is incorrect. Finally, both abstract classes and interfaces can define abstract methods, so option F is incorrect.

6. What is the output of the following program?

```
1: public class Supper {
2: public static void eat() throws IOException {
3: try {
4: System.out.print("1");
5: throw new IOException();
6: }catch(IOException e) {
7: System.out.println("2");
8: throw e;
9: }finally {
10: System.out.println("3");
11: }
12: }
13: public static void main(String [] args) {
14: eat();
15: System.out.println("4");
16: }
17: }
```

- A 1234
- B 12 and a stack trace for IOException
- C 123 and a stack trace for IOException
- D 1234 and a stack trace for IOException
- ✓ E The code does not compile.

i Line 14 attempts to invoke the eat() method, which declares the checked exception IOException. Because main() neither handles nor declares the IOException, line 14 generates a compiler error.

7. 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.ofLocalizedDateTime(FormatStyle.SHORT);
System.out.println(d.format(f));
```

- ✓ 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.

The Period to subtract is 1 year, 2 months, and 3 days.

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

- ✓ **A** All objects in Java are accessed via a reference.
- **B** By casting an object to a superclass reference, you permanently lose access to methods defined in the subclass.
- ✓ **C** The type of the object determines which properties exist within the object in memory.
- ✓ **D** The type of the reference to the object determines which methods and variables are accessible to the Java program.
- **E** By casting an object to a subclass reference, you may add new attributes to the object in memory.

Although objects exist in memory, they are accessed only via references, so option A is correct. Option B is incorrect, as you can reclaim access to lost methods by explicitly casting the object to a subclass of the object. Options C and D are correct and define the difference between and object and a reference. Finally, option E is incorrect since casting an object does not modify the object in memory.

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

```
1: public class Giraffe {
2: public int height;
3: public int getHeight() {
4: return height;
5: }
6: public void setHeight(int height) {
7: this.height = height;
8: }
9: }
```

- **A** This class currently exhibits data encapsulation.
- **B** This class currently exhibits immutability.
- ✓ **C** Making line 2 private would exhibit data encapsulation.
- **D** Making line 2 private would exhibit immutability.
- ✓ **E** Making line 2 private and removing lines 6–8 would exhibit data encapsulation.
- ✓ **F** Making line 2 private and removing lines 6–8 would exhibit immutability

Data encapsulation requires private instance variables. Although setters are allowed in an encapsulated class, they are not required. Immutability requires private instance variables and no setters.

10. What is the output of the following code?

```
LocalDate date = LocalDate.of(2018, Month.APRIL, 30).plusMonths(-1)
.plusYears(20);
System.out.println(date.getYear() + " " + date.getMonth() + " "
+ date.getDayOfMonth());
```

A 2018 MARCH 30

B 2018 APRIL 30

C 2018 MAY 30

✓ D 2038 MARCH 30

E 2038 APRIL 30

F 2038 MAY 30

i The date starts out as April 30, 2018. Adding a negative month is like subtracting a month, which bring us to March 30, 2018. Then adding 20 years brings us to March 30, 2038, making option D the correct answer.

11. What is the result of the following code?

```
1: public class PrintCharacters {
2: public static void main(String[] args) {
3: char value = 'c';
4: do System.out.print(value++);
5: while (value <= 'f');
6: } }
```

A cde

✓ B cdef

C def

D defg

E The code will not compile because of line 4.

F The code will not compile because of line 5.

i The code compiles and runs without issue. Because char is equivalent to an unsigned short in Java, the post-increment operator can run on it without issue. The loop runs four times, so only options B and D could be correct. Since the post-increment operator was used, the first value output is 'c', so the answer is option B. If the pre-increment operator had been used instead, the answer would be option D.

12. Which is true of the following code?

```
1: package abc;
2: import java.math.*;
3: public class W {
4: public void method() { }
5: int a;
6: }
```

A The code will not compile if line 1 is removed.

B The code will not compile if line 2 is removed.

✓ C The code will not compile if line 3 is removed.

D The code will not compile if line 4 is removed.

E The code will not compile if line 5 is removed.

i package and import statements are optional in a class. A class will compile without any fields or methods. The only thing required is the actual class declaration on line 3.

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

```
3: int x = 10 % 2;
4: int y = 3 / 5 + ++x;
5: int z += 4 * x;
6: System.out.print(x+","+y+","+z);
```

A 0,1,0

B 1,1,4

C 2,1,8

D 0,2,1

E The code will not compile because of line 4.

✓ F The code will not compile because of line 5.

i Line 5 uses the compound operator on the variable z, which has not yet been defined. The left-hand side of the expression must reference a variable that is already declared. The code will therefore not compile, and option F is correct.

14. What is the result of the following program?

```
1: public class Egret {
2: private String color;
3: public Egret(String color) {
4: color = color;
5: }
6: public static void main(String[] args) {
7: Egret e = new Egret();
8: System.out.println("Color:" + e.color);
9: } }
```

- **A** Color:
- **B** Color:null
- **C** Color:White
- **D** Compiler error on line 4
- ✓ **E** Compiler error on line 7
- **F** Compiler error another line

i Line 7 does not compile because there isn't a constructor defined with no arguments. If lines 3–5 were removed, the default constructor would be used. In this case, a constructor is declared, so the default constructor is not generated.

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

```
1: public class Plant {
2: public boolean flowering;
3: public Leaf [] leaves;
4: }
5: class Leaf {
6: public String color;
7: public int length; }
```

- ✓ **A** The code compiles successfully and two bytecode files are generated: Plant.class and Leaf.class.
- **B** The code compiles successfully and one bytecode file is generated: Plant.class.
- **C** A compiler error occurs on line 1.
- **D** A compiler error occurs on line 3.
- **E** A compiler error occurs on line 5.
- **F** A compiler error occurs on another line.

i The code does not contain any compiler errors. It is valid to define multiple classes in a single file as long as only one of them is public and the others have the default access.

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

- A do while loops can execute the loop body exactly zero times.
- B do while loops contain an increment clause.
- ✓ C for loops can execute the loop body exactly zero times.
- ✓ D for loops contain an increment clause.
- ✓ E while loops can execute the loop body exactly zero times.
- F while loops contain an increment clause.

i Options B and F are incorrect because only for loops have an increment clause. Other loops can increment a variable inside the loop, but this is not a clause that is part of the loop construct. Option A is incorrect because do while loops check the condition after the first loop execution.

17. Given the following code, which statements can fill in the blank on line 5 of Car to have the code compile? (Choose all that apply)

```
1: package my.vehicles;
2:public class Vehicle {
3: public String make;
4: protected String model;
5: private int year;
6: int mileage;
7: }

1: package my.vehicles.cars;
2: import my.vehicles.*;
3: public class Car extends Vehicle {
4: public Car() {
5: _____
6: } }
```

- ✓ A make = "Honda";
- ✓ B model = "Pilot";
- C year = 2009;
- D mileage = 15285;
- E None of the above

i The make field in Vehicle is public, so it is accessible anywhere. Therefore, option A is a correct answer. The model field is protected, so it is accessible in child classes. Because Car extends Vehicle, option B is a correct answer. The year field is private in Vehicle, so option C does not compile. The mileage field has the default access, but Car is in a different package than Vehicle, so option D is incorrect.

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

A String[] grades; grades = {"B", "C", "F", "A", "D"};

✓ B String[] grades; grades = new String[] {"B", "C", "F", "A", "D"};

✓ C String[] grades = {"B", "C", "F", "A", "D"};

✓ D String grades [] = {"B", "C", "F", "A", "D"};

E String grades[]; grades = new []String {"B", "C", "F", "A", "D"};

F String []grades[] = {"B", "C", "F", "A", "D"};

i Option A attempts to use an anonymous initializer, which is only allowed in the declaration. Option B correctly separates the declaration and initialization. Option C correctly uses an anonymous initializer. Option D shows the brackets can be either before or after the variable name. Option E is incorrect because the brackets are not allowed to be before the type. Option F is incorrect because the initializer is a one-dimensional array whereas the declaration is a two-dimensional array.

19. Which is the result of the following?

```
14: List<String> numbers = new ArrayList<>();
15: numbers.add("4"); numbers.add("7");
16: numbers.set(1, "5");
17: numbers.add("8");
18: numbers.remove(0);
19: for (String number : numbers) System.out.print(number);
```

A 48

✓ B 58

C 478

D 578

E 78

F An exception is thrown.

G The code does not compile.

i It is legal to have multiple statements on the same line. After line 15, the list contains [4, 7]. Line 16 replaces a value, making the result [4, 5]. Line 17 adds a new value to the end, giving us [4, 5, 8]. Finally, line 18 removes the element at index 0, resulting in a final value of [5, 8].

20. What is the output of the following code?

```
1: interface HasHindLegs {
2: int getLegLength();
3: }
4: interface CanHop extends HasHindLegs {
5: public void hop();
6: }
7: public class Rabbit implements CanHop {
8: int getLegLength() { return 5; }
9: public void hop() { System.out.println("Hop"); }
10: public static void main(String[] args) {
11: new Rabbit().hop();
12: }
13: }
```

- A Hop
- B No output
- C The code will not compile because of line 4.
- D The code will not compile because of line 7.
- ✓ E The code will not compile because of line 8.
- F The code compiles but throws an exception at runtime.

i Recall that interfaces can extend other interfaces. In this example, Rabbit, as the first concrete subclass, inherits two abstract methods: getLegLength() and hop(). Although the implementation of hop() is correct, the implementation of getLegLength() is incorrect. The access modifier for all abstract interface methods is assumed to be public, whereas the Rabbit subclass uses the default access modifier, resulting in a subclass implementing a method with a less accessible modifier. Therefore, the code will fail to compile because of line 7, and option E is correct.

21. Which can be inserted in the blank to make this code compile? (Choose all that apply)

```
import _____;

public class StartOfSummer {
public static void main(String[] args) {
LocalDate date = LocalDate.of(2014, Month.JUNE, 21);
}
}
```

- A java.date.*
- B java.date.LocalDate
- ✓ C java.time.*
- ✓ D java.time.LocalDate
- E java.util.*
- F java.util.LocalDate

i The Java 8 date and time classes are in the java.time package.

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

```
public class ExceptionLogger {
public static void logException(RuntimeException e) {
System.out.println("Logging " + e);
}
public static void main(String [] args) {
logException(new _____);
}
}
```

- A Error()
- B Exception()
- C java.io.IOException()
- ✓ D NullPointerException()
- ✓ E RuntimeException()

i Only a RuntimeException or a subclass can be passed into the logException method. NullPointerException is a subclass of RuntimeException. Exception is the parent class of RuntimeException. IOException is a subclass of Exception, but not RuntimeException. Error is a subclass of Throwable, but not Exception.

23. What could be the output of the following code given that e() could be left with a blank implementation or have a one-line implementation that throws any type of exception?

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

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

i If e() does not throw an exception, the output is abcd. If e() throws a runtime exception, the finally block will still run and then the main() method will throw that exception, resulting in the output ab followed by a stack trace. e() cannot throw a checked exception because it does not declare one.

24. What is the output of the following code?

```
1: public abstract class BigCat {
2: protected final void purr() { System.out.println("BigCat purrs!"); }
3: public static void main(String[] args) {
4: BigCat bigCat = new Ocelot();
5: bigCat.purr();
6: }
7: }
8: class Ocelot extends BigCat {
9: protected final void purr() { System.out.println("Ocelot purrs!"); }
10: }
```

A BigCat purrs!

B Ocelot purrs!

C The code will not compile because of line 4.

D The code will not compile because of line 5.

✓ E The code will not compile because of line 9.

i The code fails to compile because the method purr() is marked final in the superclass BigCat, which means the subclass Ocelot cannot override it. Therefore, the declaration of purr() in Ocelot is invalid, and the code will not compile because of line 9.

25. What is the output of the following program?

```
1: public class ColorPicker {
2: public void pickColor() {
3: try {
4: System.out.print("A");
5: fail();
6: } catch (NullPointerException e) {
7: System.out.print("B");
8: } finally {
9: System.out.print("C");
10: }
11: }
12: public void fail() {
13: throw new ArithmeticException();
14: }
15: public static void main(String[] args) {
16: new ColorPicker().pickColor();
17: System.out.print("D");
18: }
19: }
```

A ABCD

B ABD

C A and a stack trace for ArithmeticException

✓ D AC and a stack trace for ArithmeticException

E ACD and a stack trace for ArithmeticException

The pickColor() method is invoked on line 16. A is output on line 4 and then fail() is invoked. An ArithmeticException is thrown and not caught within pickColor(), so the finally block executes and prints C. Then the exception is thrown to main(). Because main() does not catch it either, the method returns to the caller and a stack trace is printed for the ArithmeticException.

26. What is the result of the following?

```
23: List<String> list = Arrays.asList("a", "B", "d", "c");
24: Collections.sort(list);
25: String[] array = list.toArray(new String[4]);
26: for (String string : array) System.out.print(string);
```

A. aBcd

B. aBdc

✓ C. Bacd

D. Compiler error on line 23

E. Compiler error on line 25

F. An exception is thrown.

Line 23 creates a fixed-size list with four elements. Line 24 sorts the list into natural order. Uppercase letters sort before lowercase letters. Line 25 converts back to an array and line 26 outputs each element.

27. What is the result of compiling this class?

```
1: public class Frog {
2: public int Frog() { return 0; }
3: private List<Integer> legs;
4: public Frog() {
5: legs = new ArrayList<Integer>();
6: for (int i = 0; i < 4; i++) {
7: legs.add(i);
8: } } }
```

A. The code compiles successfully.

B. The first compiler error occurs on line 2.

✓ C. The first compiler error occurs on line 3.

D. The first compiler error occurs on line 4.

E. The first compiler error occurs on line 5.

F. The first compiler error occurs on line 6.

G. The first compiler error occurs on line 7.

Because the line numbers begin with line 1, this means the full class is shown here. However, the import statement of java.util.ArrayList is missing. This prevents the code from compiling on line 3 since where ArrayList is from is not specified. The question tries to trick you. Line 2 is a method and not a constructor, but it does compile. Line 7 checks that you know that an int is autoboxed into an Integer. And yes, the exam will appear to ask about one thing while really testing you on another.

28. Which of the following lines do not compile? (Choose all that apply)

```
1: interface Swim {
2: default void swim() { }
3: }
4: class Swimmer implements Swim {
5: public void crawl() {}
6: default void butterfly(boolean fast) {}
7: private int numberStrokes() return 0;
8: secret void secretWeapon() { }
9: }
```

- **A** Line 2
- **B** Line 5
- ✓ **C** Line 6
- ✓ **D** Line 7
- ✓ **E** Line 8
- **F** None; all the lines compile.

i Line 2 does compile starting in Java 8 because default interface methods are allowed. However, it means that method is a default implementation, not that the access level is default. Line 6 does not compile because default methods are only allowed in interfaces and not regular classes. Line 7 does not compile because the method body must be surrounded with {}. Line 8 does not compile because secret is not a keyword in Java.

29. Which of the following are true about the enhanced for loop? (Choose all that apply)

- **A** The terms must be compile-time constant values.
- ✓ **B** Can be used to iterate over arrays
- ✓ **C** Can be used to iterate over List objects
- **D** The data type of the reference on the left-hand side must exactly match the data type of the elements on the right-hand side.
- **E** Allows access to the built-in counter of the current loop position

i The only answers that are true are options B and C, that the enhanced for loop, or for-each loop, can be used to iterate over arrays or List objects. Option A refers to a condition for case statement values within switch statements. Option D is incorrect because the reference on the left-hand side only needs to be a superclass of the data types on the right-hand side, not an exact match. Finally, option E is incorrect because there is no such built-in position pointer.

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

- A check(() → s.isEmpty());
- ✓ B check(s → s.isEmpty());
- C check(String s → s.isEmpty());
- ✓ D check((String s) → s.isEmpty());
- E check((s1) → s.isEmpty());
- F check((s1, s2) → s1.isEmpty());

i Predicate<String> takes a parameter list of one parameter using the specified type. Options A and F are incorrect because they specify the wrong number of parameters. Option C is incorrect because parentheses are required around the parameter list when the type is specified. Option E is incorrect because the name used in the parameter list does not match the name used in the body.