SALE IS ON 🔥 | FEW HOURS LEFT | BUY 2 & GET ADDITIONAL 25% OFF | Use Coupon - FLASHSALE

SKILLCERTPRO
IT CERTIFICATION TRAININGS

0

Java / By SkillCertPro

Practice Set 12

Your results are here!! for" Java SE 8 Programmer I - OCA (1Z0-808) Practice Test 12 "

38 of 60 questions answered correctly

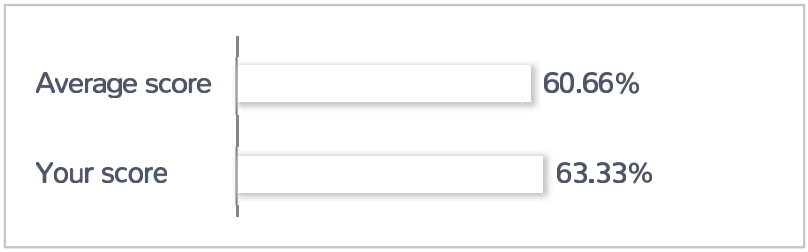
Your time: 07:56:40

Your Final Score is : 38

You have attempted : 60

Number of Correct Questions : 38 and scored 38

Number of Incorrect Questions : 22 and Negative marks 0



You can review your answers by clicking view questions.

Important Note : Open Reference Documentation Links in New Tab (Right Click and Open in New Tab).

Restart Test | View Answers

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34

35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51

52 53 54 55 56 57 58 59 60

Correct Incorrect

Review Question

1. Question

Which of the options are correct for the following code? (Select three)

```
public class Prime { // line 1
public static void main(String[] args) { // line 2
char a = 'a'; // line 3
char b = -1; // line 4
char c = '2'; // line 5
integer d = 1000; // line 6
System.out.println(++a + b++ * c – d); // line 7
}
}
```

- [ ] Line 5 fails to compile
- [x] **Line 6 fails to compile**
- [ ] Line 3 fails to compile
- [x] **Line 4 fails to compile**
- [ ] **Line 7 fails to compile**

Incorrect

The code at line 4 fails to compile because you can't assign a negative value to a primitive char data type without casting. There is no primitive data type with the name "integer." The valid data types are int. The variable d remains undefined on line 7 because its dec- laration fails to compile on line 6.

2. Question

Fill in the blank: The concept of multiple inheritances is implemented in Java by extending one class and implementing _____.

- one or more constructors
- one or more methods
- one or more classes
- **one or more interfaces**

Correct

3. Question

Which one of the following cannot be used in place of the dashed (_____) line to compile the code?

```
interface Foo {
}

public class FooBar implements Foo {
public _____ getX() {
return this;
}
}
```

- **Class**
- Foo
- FooBar
- Object

Correct

This is an instance of the current class. So, any FooBar can be used as a return type of the method. Again FooBar implements Foo, and Object is the superclass of instances in Java. So both Foo and Object are valid.

4. Question

Which of the following modifiers is not implicitly applied to all interface variables.

- **abstract**
- final
- public

static

Correct

Variables can be implicitly public, static, and final in the interface. However, they cannot be declared as abstract in the interfaces, nor in the classes.

5. Question

What does super keyword in the line 8 do in the following code?

```
public class Animal {
public void animalMethod() {
System.out.print (“An animal.”);}
}
public class Bird extends Animal {
public void animalMethod() {
System.out.print(“A Bird. “);
super.animalMethod();}
}
public class AnimalTest{
public static void main(String[] args) {
Bird b = new Bird();
b.animalMethod(); }
}
```

- **Call the overridden method from parent class**
- Prevent calling the overridden method from parent class
- Work as an instance variable
- Work as a return type for the void method

Correct

Super keyword forces Java to call the version of the method in the superclass instead of in the current class.

6. Question

Which of the following statements about a finally block is true?

- Every line of the finally block is guaranteed to be executed.

- The finally statement requires brackets [ ] .
- **The finally statement requires curly bracket { }**
- The finally block cannot throw an exception.

Incorrect

A finally block can throw an exception, so not every line of the finally block would be executed. An if statement requires brackets {}.

We Are Here!

7. Question

What is the output of the following application?

```
package yoyo;
public class TestGame {
public String runTest(boolean spinner, boolean roller) {
if(spinner = roller) return “up”;
else return roller ? “down” : “middle”;
}
public static final void main(String pieces[]) {
final TestGame tester = new TestGame();
System.out.println(tester.runTest(false,true));
}
}
```

- **up**
- middle
- down
- The code does not compile.

Incorrect

The code compiles without issue, so Option D is incorrect. The key here is that the if-then statement in the runTest() method uses the assignment operator (=) instead of the (==) operator. The result is that spinner is assigned a value of true, and the statement (spinner = roller) returns the newly assigned value. The method then returns up, making Option A the correct answer. If the (==) operator had been used in the if-then statement, then the process would have branched to the else statement, with down being returned by the method.

8. Question

Fill in the flank: In Java, _____ method are considered virtual and capable of being overridden in a subclass.

- static
- final
- **protected instance methods**
- private

Correct

In Java, only non-static, non-final, and non-private methods are considered virtual and capable of being overridden in a subclass.

9. Question

Which line of code causes an exception?

```
String[][] bird = new String[2][5];
bird[1][1] = "AAAA";
bird[1][0] = "BBBBBBB";
bird[1][2] = "CCCCCCCCC";
bird[2][1] = "DDDDDDDDD";
```

- **line 5**
- line 2
- line 4
- line 3

Correct

This is an 2D array of size 2X5. Line 5 try to reference the third element of the outer array. Since there is no such position, it throws an exception.

10. Question

Which of one is a different array than the others?

- int[] peeps[] = new int[1][1];
- int[][] peeps = new int[1][1];
- **int[] peeps[] = new int[][] { { 0, 0 } };**
- int[] peeps[] = new int[][] { {0}, {0} };

Correct

The bracket can be before or after the variable name and produce the same array. 4th one also a 2-dimensional array. So 3rd one is different

11. Question

What does the code output?

```
public class Numbering{
public static void main(String[] args) {
Float[] number = new Float[4];
number[0]=1.0f;
number[1]=0.5f;
number[2]=2.55f;
System.out.println(Arrays.toString(number)); }
}
```

- [1.0, 0.5, 2.55]
- [1.0, 0.5, 2.55, 0]
- {1.0, 0.5, 2.55, null}
- **[1.0, 0.5, 2.55, null]**

Correct

The use of static method toString() of Arrays class will print the string representation of objects in an array. The 4th object in the array was not initialize, so it will give null value in 4th place. Output comes with square bracket [ ]

12. Question

What is the output of the following code?

```
StringBuilder india = new StringBuilder("99");
india.append(" 999");
```

india.append(" 1999");

System.out.print(india);

- 99
- 1999
- 3097
- **99 999 1999**

Correct

Since StringBuilder is mutable, each call of append will concatenate the value to the string india.

13. Question

What is the output of the following?

List players = new ArrayList<>(1);

players.add("Messy");

players.add("Fredrik");

players.add("Ronaldo");

players.remove(1);

System.out.println(players);

- **[Messy, Ronaldo]**
- [Fredrik, Ronaldo]
- [Messy, Fredrik, Ronaldo]
- [Messy]

Correct

While the ArrayList is declared with an initial capacity of one element, it is free to expand as more elements are added. Each of the three calls to the add() method adds an element to the end of the ArrayList. The remove() method call deletes the element at index 1, which is Fredrik. So, finally we have Messy, Ronaldo in the list players.

14. Question

Which code will compile without any error?

- Predicate string = (StringBuilder ba) -> true;
- Predicate string = (StringBuilder ba) {return -> true };
- Predicate string = (StringBuilder ba) -> {return true; };
- None of the above

Incorrect

A Predicate is defined as a type mapping to a boolean. return is allowed, but extra semiclone (;) is an e.

15. Question

In Java inheritance, which class cannot be a parent-class?

- object class
- abstract class
- public class
- **final class**

Correct

Final class cannot be a parent-class. Because it cannot be extended and we cannot derive any class form it.

16. Question

Which of the following is true about methods in an interface in java?

- Private and protected access modifiers can also be used to declare methods in interface
- **An interface can contain abstract method**
- Interface cannot have default method
- None of the above

Correct

Along with abstract methods, an interface may also contain constants, default methods, static methods.

17. Question

What is the result of the following?

```
import java.time.*;
import java.time.format.*;
public class HowLong {
public static void main(String[] args) {
LocalDate newYears = LocalDate.of(2017, 1, 1);
Period period = Period.ofDays(1);
DateTimeFormatter format = DateTimeFormatter.ofPattern("mm-dd-yyyy");
System.out.print(format.format(newYears.minus(period)));
}
}
```

- 42736
- 42735
- The code does not compile.
- **The code compiles but throws an exception at runtime.**

Correct

When creating a formatter object, remember that MM represents month while mm represents minute. Since there are not minutes defined on a LocalDate object, the code throws an UnsupportedTemporalTypeException. You don't need to know the name of the exception, but you do need to know that an exception is thrown.

18. Question

What is the correct statement about throw and throws keyworda?

- throw is used to throw an exception & throws is used to declare an exception
- throw is used in method implementation & throws is used in method signature
- using throw keyword one can throw only 1 exception at a time & throws can declare multiple exception at a time
- **All are correct answers**

Correct

19. Question

What is the output of the following application?

```
public class WithException{
public static void main(String[] dribble) {
try {
System.out.print(2);
throw new ClassCastException();
} catch (ArrayIndexOutOfBoundsException e) {
System.out.print(0);
} catch (Throwable e) {
System.out.print(5);
} finally {
System.out.print(0);
}
System.out.print(1);
}
}
```

- **2501**
- 501
- 51
- 2

Correct

The try block throws a ClassCastException. Since ClassCastException is not a subclass of ArrayIndexOutOfBoundsException, the first catch block is skipped. For the second catch block, ClassCastException is a subclass of Throwable, so that block is executed. Afterward, the finally block is executed and then control returns to the main() method with no exception being thrown. so it will print 2501

20. Question

What is the result of the following code?

```
6: int count = 0;
7: do {
8: do {
9: count++;
10: } while (count < 2); 11: break; 12: } while (true); 13: System.out.println(count);
```

- [x] 2
- [ ] 3
- [ ] The code does not compile
- [ ] This is an infinite loop

Correct

At first this code appears to be an infinite loop. However, there is a break statement. On line 6, count is to 0. On line 9, it is changed to 1. Then the condition on line 10 runs. count is less than 2 so the inner loop continues. Then count is set to 2 on the next iteration of the inner loop. The loop condition on line 10 runs again and this time is false. The inner loop is completed. Then line 11 of the outer loop runs and sends execution to after the loop on line 13. At this point count is still 2, so Option A is correct.

21. Question

Which keywords are required with a try statement?

- [ ] this
- [ ] super
- [ ] finalize
- [x] **catch or finally**

Correct

A try statement requires a catch or a finally block. Without one of them, the code will not compile. Note that finalize is not a keyword, but a method inherited from java.lang.Object

22. Question

What is the result of compiling and running the following application?

```
package castles;
class DragonException extends Exception {}
public class Lair {
public void openDrawbridge() throws Exception { // r1
try {
throw new Exception("This Exception");
} catch (RuntimeException e) {
throw new DragonException(); // r2
```

```
} finally {
throw new RuntimeException("Or maybe this one");
}
}

public static void main(String[] moat) throws Exception {
new Lair().openDrawbridge(); // r3
}
}
```

- The code does not compile because of line r1.
- The code does not compile because of line r2.
- The code does not compile because of line r3.
- **The code compiles, but a stack trace is printed at runtime**

Incorrect

The openDrawbridge() is capable of throwing a variety of exceptions, including checked Exception and DragonException as well as an unchecked RuntimeException. All of these are handled by the fact that the method declares the checked Exception class in the method signature, which all the exceptions within the class inherit. For this reason, the openDrawbridge() method compiles without issue. The call to openDrawbridge() in the main() method also compiles without issue because the main() method declares Exception in its signature. For these reasons, the code compiles but a stack trace is printed at runtime, making Option D the correct answer. In case you are wondering, the caller would see RuntimeException: Or maybe this one in the stack trace at runtime, since the exception in the finally block replaces the one from the try block. Note that the exception in the catch block is never reached because the RuntimeException type declared in the catch block does not handle Exception.

23. Question

What is the result of this code?

```
public static void main(String args[]) {
int a[] = new int[] {10, 2, 12, 3, 4, 16, 8, 9, 15};
int n = 6;
n = a[a[n] / 2];
System.out.println(a[n]/2);
}
```

- 4
- **2**
- 12
- Code will run with an exception

Correct

Array a contains 9 elements. n contains 6 thus in next line n value is assigned as 4, because 6th eleme. is 8 and 8/2 = 4. So a[4] = 4 and a[4] / 2 = 2.

24. Question

Suppose setX() is method of GeometricObject clas and setRadius() is a method of Circle class. What is the output for this code?

```
GeometricObject c = new Circle();
c.setX(1);
System.out.println( c.setRadius(10) );
```

- 10
- 1
- **Code will not complie**

Correct

This is a Polymorphism question. Here c (the variable) is of type GeometricObject but the object it references is of the subtype Circle. You can only directly access the fields and methods of an object's declared type. So c.setRadius() has an error and code will not compile.

25. Question

What is the result of the following code?

```
class Phone {
static void call() {
System.out.println("Call-Phone");
}
}
class SmartPhone extends Phone{
static void call() {
```

```
System.out.println("Call-SmartPhone");
}
}
class TestPhones {
public static void main(String... args) {
Phone phone = new Phone();
Phone smartPhone = new SmartPhone();
phone.call();
smartPhone.call();
}
}
```

- Call-Phone Call-SmartPhone
- Call-Phone null
- **Call-Phone Call-Phone**
- Code throws an exception

Incorrect

Code has no error. The static method belongs to a class, not to its objects. call() is a static method, smartPhone.call() calls the method call() defined in the class Phone.

26. Question

Fill in the blank. In Java object-oriented programming, a new class can be designed by extending existing class. This is an example of _____.

- **Inheritance**
- Encapsulation
- Polymorphism
- Interface

Correct

27. Question

Given the following definitions of the classes ColorPencil and TestColor, which option, if used to replace //INSERT CODE HERE, will initialize the instance variable color of the reference variable myPencil with the String literal value “RED”?

```
class ColorPencil {
String color;
ColorPencil(String color) {

//INSERT CODE HERE

}}
class TestColor {
ColorPencil myPencil = new ColorPencil(“RED”);
}
```

- **this.color = color;**
- color = color;
- super.color = color;
- color = RED;

Correct

The second option will assign the value of the method parameter to itself. Last two options are not valid declaration in Java.

28. Question

What does the following code output?

```
public class Bee{
public static void main(String a[]){
boolean b = true;
Boolean booleanObj1 = new Boolean(b);
Boolean booleanObj2 = new Boolean(“false”);
System.out.println(booleanObj1.booleanValue());
}
}
```

- FALSE

- (●) TRUE
- ( ) Compiling error
- ( ) Compile, but throws an exception at runtime

Correct

You can create Boolean wrapper class object by passing either boolean primitive value or boolean value string to the Boolean constructor. So, there is no error. The booleanValue() method of Boolean class is get the primitive equivalent of this Boolean object. booleanObj1 has value “true” because b = true.

29. Question

What is the output of this code?

```
public class Cue{
public static void main(String[] args) {

for (int i = 1; i < 10; ++i) { if (i > 3 && i < 8) { continue; } System.out.print(i + " "); } } }
```

- ( ) 1 2 3 4 5 6 7 8 9
- (●) **1 2 3 8 9**
- ( ) 4 5 6 7
- ( ) Code has compilation error

Correct

The continue keyword causes the loop to immediately jump to the next iteration of the loop. When the value of i becomes more than 3 and less than 8, continue statement is executed, which skips the execution of System.out.print(i); statement.

30. Question

Which is the correct statement about unchecked exceptions? (Select three)

- [x] **Unchecked exceptions are not checked at compile time**
- [ ] They are checked at compile time
- [ ] Program will not execute if it throws an unchecked exception
- [x] **All Unchecked exceptions are direct sub classes of RuntimeException class**

- [x] If your program is throwing an unchecked exception, even if you didn't handle/declare that exception, the program won't give a compilation error

Correct

31. Question

Given the following code, which of the following statements are true? (Select 2 options)

```
class MyExam {
void question() {
try {
question();
}catch (StackOverflowError e) {
System.out.println("caught");
}
}
public static void main(String args[]) {
new MyExam().question();
}
}
```

- [x] The code will print caught
- [x] The code would print caught if question() throws the exception NullPointerException
- [ ] The code would print caught if StackOverflowError were a runtime exception
- [ ] The code won't print caught

Incorrect

The control will be transferred to the exception handler for StackOverflowError when it's encountered. Hence it will print caught. Exception handlers execute when the corresponding runtime exceptions are thrown. An exception handler for the class StackOverflow can't handle exceptions of the class NullPointerException because NullPointerException is not a superclass of StackOverflowError.

32. Question

What is the code to get today's date in the format like Friday, March 30, 2018?

LocalDate date = LocalDate.today();
System.out.println(date.format(DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL)));

LocalDate date = LocalDate.now();
System.out.println(date.format(DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL)));

LocalDate date = LocalDate.now();
System.out.println(format(DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL)));

None of these is correct code

Correct

You have to use java DateTimeformatter: static DateTimeFormatter ofLocalizedDate(FormatStyle style). Style FULL = Friday, March 30, 2018, LONG = March 30, 2018, MEDIUM = Mar 30, 2018, SHORT = 30.03.18

33. Question

What code in the line 15 will remove the compilation error?

```
public interface Consumable {
public abstract int getCalorieCount();
}
public class TimsProduct{
private String name;
private double cost;
private double price;
public TimsProduct(String name, double cost, double price) {
this.name = name;
this.cost = cost;
this.price = price;}
}
public class Donut extends TimsProduct implements Consumable {

\\ YOUR CODE HERE

public int getCalorieCount() {
return 0;
}
}
```

private Donut(String name, double cost, double cost) { this(name, cost, price); }

- private Donut(String name, double cost, double cost) { super(name, cost, price); }
- public int getName() { return name;}
- Code will compile without any code in the specified line

Incorrect

Since you are implementing Consumable class and extending TimsProduct class, you must need to create constructor. Super keyword used to call variables from parent class. You can't use this keyword, because used for calling a constructor inside another constructor.

34. Question

What code of line will give an output 12-31-2017

```
public class Pinaki{
public static void main(String[] args) {
LocalDate year = LocalDate.of(2018, 1, 10);
Period p = Period.ofDays(10);
DateTimeFormatter f = DateTimeFormatter.ofPattern("MM-dd-yyyy");

// YOUR CODE GOES HERE
}}
```

- System.out.print(year.minus(p));
- System.out.print(f.format(year.minus(p)));
- System.out.print(f.format(p));
- System.out.print(format(year.minus(p)));

Incorrect

Yuo have to call formt with the object. The code starts by correctly creating a date representing January 10, 2018, and a period representing 10 day. It then explicitly defines the format as month followed by day followed by year. Finally, the code subtracts 10 day, giving us 12-31-2017.

35. Question

What is the result of the code?

```
public class Pip{
public static void main(String args[]){
```

```
StringBuffer x = new StringBuffer(“Hello there “);
StringBuffer y = new StringBuffer(“Java 8”);
x.append(y);
System.out.println(x);
}}
```

- Code will not compile
- Hello there
- **Hello there Java 8**
- Java 8

Correct

append() method of class StringBuffer is used to concatenate the string representation to the end of invoking string

36. Question

What is the output of the following code?

```
public class GuruArray2 {
public static void main(String args[]) {
int[] arr1;
int[] arr2 = new int[3];
char[] arr3 = {‘a’, ‘b’};
arr1 = arr2;
arr1 = arr3;
System.out.println(arr1[0] + ” : ” + arr1[1]);
}
}
```

- a : 0
- 0
- a : b
- **Compilation error**

Correct

Because a char value can be assigned to an int value, you might assume that a char array can be assigned to an int array. But we're talking about arrays of int and char primitives, which aren't the same as a primitive int or char. Arrays them- selves are reference variables, which refer to a collection of objects of similar type.

37. Question

What is the output of the following code?

```
public class GuruString2 {

public static void main(String args[]) {
String ejg = "game".replace('a', 'Z').trim().concat("Aa");
ejg.substring(0, 2);
System.out.println(ejg);
}
}
```

- (●) **gZmeAa**
- ( ) gZmeAaZ
- ( ) gameAa
- ( ) ZAa

Correct

When chained, methods are evaluated from left to right. The first method to execute is replace, not concat. Strings are immutable. Calling the method sub- string on the reference variable ejg doesn't change the contents of the variable ejg. It returns a String object that isn't referred to by any other variable in the code. In fact, none of the methods defined in the String class modify the object's own value. They all create and return new String objects.

38. Question

Which of the following statements about java.lang.Error are most accurate? (Choose two.)

- [x] An Error should be thrown if a file system resource becomes temporarily unavailable.
- [x] **An application should never catch an Error.**
- [ ] Error is a subclass of Exception, making it a checked exception.
- [ ] **It is possible to catch and handle an Error thrown in an application.**

- [ ] An Error should be thrown if a user enters invalid input.

Incorrect

Options A and E are incorrect because they indicate states that the application can possibly recover from. An Error generally indicates an unrecoverable problem. While it is possible to catch an Error, it is strongly recommended that an application never do so, making Options B and D correct. Finally, Option C is incorrect because Error extends from Throwable, not Exception, and is unchecked.

39. Question

What is the output of the following?

```
1: public class Legos {
2: public static void main(String[] args) {
3: StringBuilder sb = new StringBuilder();
4: sb.append("red");
5: sb.deleteCharAt(0);
6: sb.delete(1, 1);
7: System.out.println(sb);
8: }
9: }
```

- ( ) r
- (•) e
- ( ) ed
- ( ) red
- ( ) The code does not compile.
- ( ) The code compiles but throws an exception at runtime.

Incorrect

Line 3 creates an empty StringBuilder. Line 4 adds three characters to it. Line 5 removes the first character resulting in ed. Line 6 deletes the characters starting at position 1 and ending right before position 1. Since there are no indexes that meet that description, the line has no effect. Therefore, Option C is correct.

40. Question

What does the following do?

```
public class Shoot {
interface Target {
boolean needToAim(double angle);
}
static void prepare(double angle, Target t) {
boolean ready = t.needToAim(angle); // k1
System.out.println(ready);
}
public static void main(String[] args) {
prepare(45, d => d > 5 || d < -5); // k2 } }
```

- It prints true.
- It prints false.
- It doesn't compile due to line k1.
- **It doesn't compile due to line k2.**
- It doesn't compile due to another line.

Incorrect

The lambda syntax is incorrect. It should be ->, not =>. Therefore, Option D is correct. If this was fixed, Option A would be correct.

41. Question

Which statement about the following class is correct?

```
package shapes;

abstract class Parallelogram {
private int getEqualSides() {return 0;}
}

abstract class Rectangle extends Parallelogram {
public static int getEqualSides() {return 2;} // x1
}

public final class Square extends Rectangle {
public int getEqualSides() {return 4;} // x2
```

```
public static void main(String[] corners) {
final Square myFigure = new Square(); // x3
System.out.print(myFigure.getEqualSides());
}
}
```

- The code does not compile due to line x1.
- **The code does not compile due to line x2.**
- The code does not compile due to line x3.
- The code compiles and runs without issue.

Incorrect

The code does not compile, so Option D is incorrect. The issue here is that the override of getEqualSides() in Square is invalid. A static method cannot override a non-static method and vice versa. For this reason, Option B is the correct answer.

42. Question

Which of the following statements is correct?

- A reference to a class can be assigned to a subclass reference without an explicit cast.
- **A reference to a class can be assigned to a superclass reference without an explicit cast.**
- A reference to an interface can be assigned to a reference of a class that implements the interface without an explicit cast.
- A reference to a class that implements an interface can be assigned to an interface reference only with an explicit cast.

Incorrect

A reference to a class can be implicitly assigned to a superclass reference without an explicit class, making Option B the correct answer. Assigning a reference to a subclass, though, requires an explicit cast, making Option A incorrect. Option C is also incorrect because an interface does not inherit from a class. A reference to an interface requires an explicit cast to be assigned to a reference of any class, even one that implements the interface. An interface reference requires an explicit cast to be assigned to a class reference. Finally, Option D is incorrect. An explicit cast is not required to assign a reference to a class that implements an interface to a reference of the interface

43. Question

What is a possible output of the following application?

```
package wrap;
public class Gift {
private final Object contents;
protected Object getContents() {
return contents;
}
protected void setContents(Object contents) {
this.contents = contents;
}
public void showPresent() {
System.out.print("Your gift: "+contents);
}
public static void main(String[] treats) {
Gift gift = new Gift();
gift.setContents(gift);
gift.showPresent();
}
}
```

- (●) Your gift: wrap.Gift@29ca2745
- ( ) Your gift: Your gift:
- ( ) **It does not compile.**
- ( ) It compiles but throws an exception at runtime.

Incorrect

The code contains a compilation problem in regard to the contents instance variable. The contents instance variable is marked final, but there is a setContents() instance method that can change the value of the variable. Since these two are incompatible, the code does not compile, and Option C is correct. If the final modifier was removed from the contents variable declaration, then the expected output would be of the form shown in Option A.

44. Question

Which of the following statements are true?

I. You can always change a method signature from call(String[] arg) to call(String... arg) without causing a compiler error in the calling code.

II. You can always change a method signature from call(String... arg) to call(String[] arg) without causing a compiler error in the existing code

- I
- II
- Both I and II
- Neither I nor II

Incorrect

From within a method, an array parameter and a varargs parameter are treated the same. From the caller, an array parameter is more restrictive. Both types can receive an array. However, only a varargs parameter is allowed to automatically turn individual parameters into an array. Therefore, statement I is correct and the answer is Option A.

45. Question

Given that FileNotFoundException is a subclass of IOException, what is the output of the following application?

```
package edu;
import java.io.*;
class School {
public int getNumberOfStudentsPerClassroom(String... students)
throws IOException {
return 3;
}
public int getNumberOfStudentsPerClassroom() throws IOException {
return 9;
}
}
public class HighSchool extends School {
public int getNumberOfStudentsPerClassroom() throws FileNotFoundException {
return 2;
}
public static void main(String[] students) throws IOException {
```

```
School school = new HighSchool();
System.out.print(school.getNumberOfStudentsPerClassroom());
}
}
```

- 2
- 3
- 9
- The code does not compile

Incorrect

The code compiles without issue, so Option D is incorrect. The rule for overriding a method with exceptions is that the subclass cannot throw any new or broader checked exceptions. Since FileNotFoundException is a subclass of IOException, it is considered a narrower exception, and therefore the overridden method is allowed. Due to polymorphism, the overridden version of the method in HighSchool is used, regardless of the reference type, and 2 is printed, making Option A the correct answer. Note that the version of the method that takes the varargs is not used in this application.

46. Question

What is the correct extension for a Java bytecode compiled file?

- .java
- **.class**
- .bytecode
- None of the above

Correct

When you compile a Java class, the translated version of the class— its bytecode—is placed in a file whose name is the name of the class followed by .class

47. Question

Consider the given situation:

X and Z are two classes, A and B are two interfaces, E is an abstract class.

Which one of the following is a correct declaration? (Select two)

- [ ] public class Y extends X, Y{ }
- [ ] public class Y implements E, A{ }
- [x] **public class Y implements B, A{ }**
- [ ] public class Y extends A{ }
- [x] **public class Y extends X implements A{ }**

Correct

We can extend single class only as a inheritance relation. We use the keyword 'implements' to implement an interface to a class. Multiple interfaces can be implemented by a class or interface but multiple classes can not be extended.

48. Question

What is the output of this code-block, if user input is 49?

```
int message = sc.nextInt();
    switch (message) {
      case 11:
        System.out.println("hi");
        break;
      case 21:
        System.out.println("bye");
        break;
      default:
        System.out.println("what?");
        break;
    }
```

- ( ) bye
- ( ) hi
- ( ) no output
- (•) **what?**

Correct

When user type 11, output will be 'hi'. For 21, output is 'bye'. For any other values, output is a default case i.e. 'what?'

49. Question

Which of the following is not true about Java object-oriented programming?

- An object can take many forms via casting
- Each object is a distinct individual
- Object can perform actions via methods
- Object can hold data
- **Object is like a procedure, separate from data**

Correct

First four options are each characteristics of Java object-oriented programming. So answer is very last one.

50. Question

What is the output of the following code?

```
public static void main(String[] args) {

int []x = {0, 1, 2, 3, 4};
int i = x.length – 1;

while(i >=0 ){
System.out.print(x[i]);
i -= 1;
}
}
```

- 1234
- **43210**
- Code will not run due to compilation error
- 4

Correct

The length of the array ‘x’ is 5, so the value of the variable ‘i’ is 4. The while loop will print the array elements in reverse order because the variable ‘i’ has an initial value of 4. So printing of the elements starts with 4 (x[4]

= 4) and runs until the value of variable ‘i’ equals 0. Therefore the first element of the array ‘x’ (x[0]) prints as a last element.

51. Question

Which one of the following has no syntax error?

- ○ int x; x = 5; x = "5"; y = 2.3;
- ○ int x; x = 5; x = "5"; double y = 2.3;
- ◉ **int x; x = 5; x = 15; double y = 2.3;**
- ○ int x; x = 5; x = 5.0; double y = 2.3;

Correct

In the 1st and 2nd options, x = “5” has syntax error because “5” is String type, but x is integer. In the 4th option, 5.0 is again float type, but x is integer.

52. Question

What does this code output?

```
public static void main(String[] args) {
System.out.println(“x\tx^2\t\n1\t2”);
}
```

- ○ xx^2
- ○ 12
- ○ **x x^2 1 2**
- ◉ x x^2 1 2
- ○ Non of the avove

Incorrect

\t produces tab, \n for new line.

53. Question

What is the result of the following Java class?

```
public class Village {
int house = 10;
public static void main(String[] data) {
int building = 5;
System.out.print(building + house);
}
}
```

- It won't compile
- 15
- buildinghouse
- It will compile, but no output

Incorrect

The main() method is static and does not have access to any class instance variables. The house variable is not static and requires a class instance variable to access. Therefore, the code does not compile when the static method attempts to access a non-static variable without an instance of the class.

54. Question

Identify the error in this code-block. What is the correct code for that line?

```
Scanner input = new Scanner(System.in);
int x;
System.out.print(“x: “);
x = input.nextLine();
System.out.println(x);
```

- String x = input.nextLine();
- System.out.println("x");
- **x = input.nextInt();**
- x = nextInt();

Correct

Since x is integer type, one must use nextInt() instead of nextLine(). Remember nextLine() method is used for getting string value.

55. Question

What is the difference between x++ and ++x?

- x++ increments x but contributes the new value of x to any expression while ++X increments x and contributes the original value of x to any expression
- **x++ increments x but contributes the original value of x to any expression while ++X increments x and contributes the new value of x to any expression**
- There is no difference between x++ and ++x
- x++ increases 1 unit, but ++x decreases 1 unit

Incorrect

x++ returns the value of x and then increments. But, ++x increments the value of x and then returns x. For example, for x++ with initial value x=0, the expression read x=0 first and then increase 1 unit. For ++x, first x value updates (x=1) and then expression reads x =1.

56. Question

Which of the following statements will output “Hello Dolly” if a string variable named s contains the exact contents “I’m Dolly”?

- if (s.equals(I'm Dolly)) { System.out.println("Hello Dolly"); }
- if (s = "I'm Dolly") { System.out.println("Hello Dolly"); }
- **if (s.equals("I'm Dolly")) { System.out.println("Hello Dolly"); }**
- String s = "I'm Dolly"; if (s == True) { System.out.println("Hello Dolly"); }

Incorrect

The equals() method compares the “value” inside String instances.

57. Question

Which one of the following statements is correct? (Select two)

- [ ] A primitive type variable holds multiple values
- [ ] String is a primitive type variable
- [x] **String is a reference type variable**
- [x] **long number = 22514554; is a primitive type variable declaration**
- [ ] There are hundreds of primitive variables in Java
- [ ] boolean is a reference type variable

Correct

There are 8 primitive types variables in Java. These are short, int, long, boolean, char, byte, double,float. String is reference type. Primitive variables hold single value while reference type may have multiple values.

58. Question

Which of the following Java declarations will not compile?

- ( ) int num = 2, number = 4;
- ( ) String name, birthday;
- (•) **double number, int value = 5;**
- ( ) long number = 100, bigNumber = 5552452;

Correct

Java does not allow declaring different types as part of the same declaration. The other three options show various legal combinations of combining multiple variables in the same declarations.

59. Question

Which of the following is not a valid name for declaring variable? (select two)

- [ ] _name
- [ ] $name
- [x] **5name**
- [ ] name$
- [x] **name&**

Correct

Variable declaration cannot start with number, but number can be used as subsequent character. $ sign is valid for variable declaration but other special characters (%,&,*,#) are not.

60. Question

What is the result while running this code?

```
public class Values {
integer x = Integer.valueOf("7");

public static void main(String[] args) {
integer x = Integer.valueOf("5");
integer y = Integer.valueOf("2");
System.out.println(x*y);
}
}
```

- (●) 10
- ( ) Code will not compile
- ( ) 14
- ( ) The code compiles but throws an exception at runtime

Incorrect

There is no class with name integer. The integer variable type is given by 'int' and Integer is a class. For compiling, you need to swap 'integer' with 'int'.

Use Page numbers below to navigate to other practice tests

Pages: 1 2 3 4 5 6 7 8 9 10 11 12 13 14

← Previous Post

Next Post →

We help you to succeed in your certification exams

We have helped over thousands of working professionals to achieve their certification goals with our practice tests.

Skillcertpro

Quick Links

ABOUT US

FAQ

BROWSE ALL PRACTICE TESTS

CONTACT FORM

Important Links

REFUND POLICY

REFUND REQUEST

TERMS & CONDITIONS

PRIVACY POLICY

Privacy Policy