Bünyamin AKTAŞ
21/06/2020

ASSIGNMENT – 10

Q1 **The answer is E.**

Since *sb* is initialized empty, *indexOf()* method used in *for* loop will be attempting to reach a nonexisting index, which will cause *StringIndexOutOfBoundsException* at runtime.

Q2 **The answers are C, E.**

The table below shows the operator precedence order.

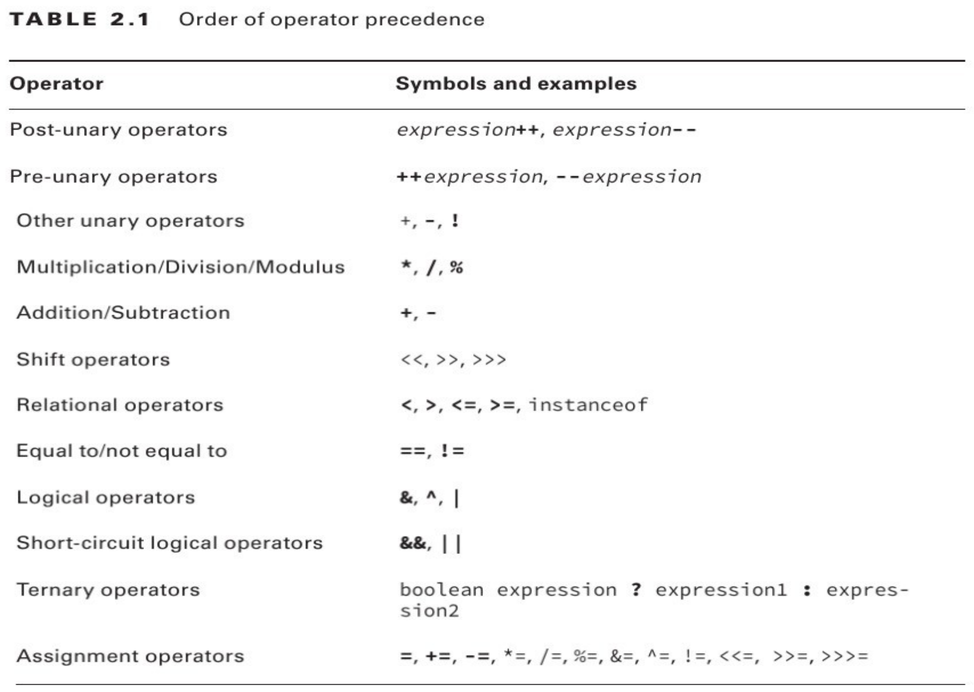

TABLE 2.1 Order of operator precedence

Operator	Symbols and examples
Post-unary operators	*expression*++, *expression*--
Pre-unary operators	++*expression*, --*expression*
Other unary operators	+, -, !
Multiplication/Division/Modulus	*, /, %
Addition/Subtraction	+, -
Shift operators	<<, >>, >>>
Relational operators	<, >, <=, >=, instanceof
Equal to/not equal to	==, !=
Logical operators	&, ^, |
Short-circuit logical operators	&&, ||
Ternary operators	boolean expression ? expression1 : expression2
Assignment operators	=, +=, -=, *=, /=, %=, &=, ^=, !=, <<=, >>=, >>>=

Oracle Certified Associate Java Programmer I Study Guide *page: 52-53*

A – Question asks for operators sorted in increasing order. Assignment operators have the least precedence so they must be at the beginning of the list if they will be used but here = comes after addition / subtraction operators so this option is wrong.

B – This option lists operators in decreasing order.

C – This option lists operators in increasing order.

D – This option lists operators in decreasing order, too.

E – This option lists operators in increasing order.

Q3 **The answers are B, C, F.**

JavaBean naming conventions are:

1 – Properties are private: *private int numEggs;*

2 – Getter methods begin with *is* if the property is a *boolean*: *public boolean isHappy() { return happy; }*

3 – Getter methods begin with *get* if the property is not a *boolean*: *public int getNumEggs() { return numEggs; }*

4 – Setter methods begin with *set*: *public void setHappy(boolean happy) { this.happy = happy; }*
5 – The method name must have a prefix of *set/get/is*, followed by the first letter of the property in uppercase, followed by the rest of the property name: *public void setNumEggs(int num) { numEggs = num; }*

According to these rules **the options B, C,** and **F** are correct.

A – This is wrong since getter methods don't take parameters.

D – This is wrong since method names beginning with "is" must return boolean values.

E – This is wrong since there is no naming convention in JavaBean rules such as "gimme".

Q4 **The answers are A and E.**

This code snippet creates a 2D array of type *int* and assigns the value 'x' to the elements of the array iterating over it. This is legal since *char* values have corresponding *int* values. But the nested *for* loops both iterates until the end of the first dimension of the array so the assignments only affects the half of the second dimension, which makes the option E correct.

Since arrays do not have any method named *size()* the last line will not compile and it must be altered which makes the option A correct.

Q5 **The answers are B, D.**

A – If a resource is unavailable temporarily, it can be handled for example making the program wait for the resource.

B – An application should never catch an Error since Error means a fatal error for the application that it must not continue to work.

C – Error is not a subclass of Exception, but a subclass of Throwable.

D – It is possible to catch and handle an Error although it is not recommended.

E – If the user enters invalid input, the program can ask for it again or throw an exception, not an Error.

Q6 The answers are A, C, D.

import statements allow the classes that belongs to the imported package, to be accessed, not the classes that belongs to the sub-packages of the imported package. So the classes of the packages *forest.ape.bonono* and *savana.sand* and the class *forest.Sloth* cannot be accessed since they are not imported explicitly, **which makes the options B, E and F incorrect.**

Option A and C are correct since they are imported explicitly. Option D is correct, too, since *java.lang* package is imported automatically by Java without needing any explicit import statement.

Q7 The answer is C.

ArrayList and StringBuilder classes are mutable since the methods can change the data stored in the instances of these classes.

String and LocalDateTime classes are immutable since the methods cannot change the data stored in the instances of these classes. **So the answer is two.**

The code compiles since all initializations are correct. LocalDateTime instances cannot be initialized calling a constructor but instead a method named *now()*.

Q8 The answer is D.

The code compiles since there is no syntactic error. *builder* is initialized with the value “Leaves growing”, which is bigger than 5, so the loop may not be causing any exception, too.

The loop calls *delete()* method on *builder* deleting the string between the indexes 0 and 5, including the former index while excluding the latter, until the length of the *builder* becomes less than 5. **So the remaining the value at the end is “wing”, which will be printed.**

Q9 The answer is D.

The code compiles. In this code all String variables points to different objects so that equality operator, ==, returns false since it compares objects, not the values.

But *equals()* method which is overridden by String class will return true since all of the String objects have “up” value. **So the output will be *false false true.***

Q10 **The answer is C.**

Equality operator compares objects not the values and the String methods returns new objects. So the lines 4 and 5 returns false. Line 6 returns false, too, since the former variable is “LOL” and the latter is “lol”.

Line 7 returns true since the method *equals()* compares two same objects with same values. Line 8 and 9 returns true since the *equalsIgnoreCase()* does not care about the cases of the values. So the code outputs three *true*.

Q11 **The answers are A, B, C.**

Since the optional loops are never referenced in this code they can be removed, whicha re on the lines 15 and 17.

Since the inner loop’s boolean condition is broader than the outer loop, it can be removed, too, which makes the options B and C correct.

Q12 **The answers are B, C.**

Since the *long* values cannot have a decimal point, options B and C prevent this line from compiling. Numerals can have underscores between digits. 1234L is a Long value while 1234l is a long value that is autoboxed automatically. So except the options B and C, all other options compile correctly.

Q13 **The answer is A.**

time is set to 1.11 and *while* loop checks if the hour, which is 1, is less than 1. This boolean expression will always return false so no line is printed.

Q14 **The answer is D.**

game is defined as a static field but not initialized so that its size is not determined. Line 6 will throw a NullPointerException since it attempts to access a non-existing index.

Q15 **The answers are C, E.**

Generics of List must have an object type in the left side of a statement. At the right side, it may be left empty but they must written. So the options C and E are correct. Option A compiles but gives a warning.

Q16 **The answers are B, D.**

In this code, shoe1 object at the beginning gets unreferenced after the last three lines so it becomes eligible for garbage collector. But this does not ensure that the Java GC will collect it since it works automatically its own way.

Q17 **The answer is C.**

ClownFish class overrides *getFish()* method and overridden methods can declare narrower exceptions so that this overriding is legal.

However, since main() method assigns a ClownFish object to Fish type reference and *getFish()* method is invoked on this reference, this invoking must handle the possible exception. So the line *f.getFish()* does not compile.

Q18 **The answer is A.**

This code creates an *ArrayList* and adds to it -5, 0, 5, respectively. And then print() method is called giving the list and a lambda expression, *e -> e < 0*, which will print values that is less then 0, as arguments. Since there is only one value that is less than 0 it will be printed which is -5.

Q19 **The answer is F.**

A *try* block must be followed by a *catch* block and/or a *finally* block. None of the options matches this rule.

Q20 **The answer is A.**

The outer loop checks if the result is bigger than 7 and since it is correct it starts to execute its body. The body increments *result* by 1 at first and then enters the inner loop which reduces *result* until 5. Then a *break* statement terminates the outer loop and the result is printed at the end. It is 5.

Q21 **The answer is C.**

new Alligator() increments the static variable *teeth*, which is initialized *to 0*, by 1 and sends it to *snap()* as argument. So the first line of the main() method prints 1 and and the second line prints 2. So the output is 1 2.

Q22 **The answer is A.**

Since String class is immutable, the methods called on String objects does not change the data of the current object but instead returns a new object with a new value. So the *concat()* method return a new object with the value “black” but since it is not assigned to any reference it is lost. The *println()* method prints “b”.

Q23 The answers are A, C, F.

The interface methods are meant to be *public* and *abstract* so that the options B, D and E incorrect. Interfaces can have *default* and *static* methods so that options A and C are correct. Since interface methods are default to *abstract,* the option F is also correct.

Q24 The answer is D.

Since the expression in the definition of the *while* loop is an assignment instead of a boolean comparison, it causes compilation error which makes the option D correct.

If the expression is fixed to boolean expression then the code would compile and makes tie “shoelace” and prints it.

Q25 The answers are B, F.

A – If the file does not contain a package statement the compiler assumes the class is the part of the **default** package.

B – Java assumes every class import *java.lang* package.

C – java.util package must be imported explicitly, if it is to be used.

D – Java does require every class to declare a package statement as understood in **A.**

E - Java does require every class to declare any package statement as understood in **A.**

F – Java assumes every class extends *Object* if no superclass is determined explicitly.

Q26 The answer is F.

Two interfaces implement single default method with the same name which force the class *Tree* to override these methods since *Tree* implements both interfaces. So the line m1 causes compilation error.

If this problem is fixed, then the code compiles and prints the return value of the overridden method. *p* can be cast to *Living* since the object *p* points to is a subclass of *Living.*

Q27 The answer is D.

Identifier names can contain letters,, numbers, $ and _. But a number cannot be used at the beginning. But the println() method attempts to print *profit$$$* which is not initialized explicitly or implicitly since it is a local variable. **So *println()* method does not compile.**

Q28 The answer is A.

Post-unary operators, such as *plan++ or plan--*, increment or decrement values of variables by 1 but output the values before the decrement occurred. Pre-unary operators, such as +*+plan or --plan*, increment or decrement the values of variables by 1 and output the new value.

So the answer is A.

Q29 The answers are B, C, E.

If a super class does not implement a no-argument constructor, its subclasses must call its constructor explicitly.

The first constructor of Trouble does not call super constructor so it causes compilation error. And *public Toruble(String, int…)* and *public Trouble(long deep)* constructors call the no-argument constructor of *Trouble* which causes a compilation error. **So these constructor also causes compilation error which makes options B, C, and E correct.**

Q30 The answers are E, F.

Since the main() method attempts to access *min* and *max* in a static way they must defined static, which makes the options A, B, C, and D incorrect.

E – Correct since *min* is default to 0, and the difference between *min* and *max* is 100.

F – Correct since it explicitly assigns *min* 0, and *max* 100.

Q31 The answers are B, E.

A – Ternary operators execute only one side.

B – A *switch* statement may have at most one *default* statement.

C – An *if-then* statement can have at most one *else* statement and can have multiple *else-if* statements.

D – I and II does not produce same produce since the latter may not executes the right side of a comparison.

E - ! operator can only be applied to boolean expressions.

Q32 The answer is C.

Line 4 appends “red” to empty object *sb*. Line 5 deletes the first element. Line 6 deletes nothing since it takes the same indexes for the beginning and for the end. So the output will be “ed”.

Q33 The answers are A, D.

Identifier names can contain letters,, numbers, $ and _. But a number cannot be used at the beginning.

A – Contains only underscores which is legal.

B – Names cannot have %.

C – Names cannot have -.

D – This is legal.

E – Names cannot have \\.

F – Names cannot have #.

Q34 The answers are B, C.

A – Use cases of inheritance and static methods are not related to each other.

B – Inheritance allows subclasses to override methods that they inherited from their superclasses.

C – Inheritance allows objects to inherit attributes.

D – It is not possible to create a Java class that does not inherit from any class since Java automatically determines *Object* class as the superclass of the all classes.

E – It depends on the coding.

Q35 The answer is E.

Java separates packages with dot. Java supports functional programming with lambda expressions. Java is object oriented and supports polymorphism, too. So the answer is E.

Q36 **The answer is C.**

This code creates a 2D String array with sizes [3][2], implicitly. So the println() method will print the sizes of the two dimensions, which are 3 and 2, respectively.

Q37 **The answers are A, B, E.**

A switch statement supports the primitive types byte, short, char, and int and their associated wrapper classes Character, Byte, Short, and Integer. It also supports the String class and enumerated types. Double, long and Object types are not supported.

Q38 **The answer is D.**

In main() method, *prepare()* method is invoked taking 45 and an expression, *d* => *d* > *5* || *d* < -*5*, which is illegal since there is no operator such as => in Java. If we change this to ->, this expression becomes a lambda expression which checks if *d* is greater then 5 or less than -5 and since 45 is bigger than 5, it would print *true*.

Q39 **The answers are B, C, E.**

Double slash before a line or lines encapsulated by /* - */ characters are interpreted as comments in Java, which makes the options B, C, and E correct. Multiple * characters can be used.

Q40 **The answers are A, F.**

import static allows us to import static members of a class. But the class name must be written in this statements. This code uses all the static members of the Grass class so they must imported either one by one, or with * character, such as below:

import static food.Grass.getGrass;
import static food.Grass.seeds;

or,

import static food.Grass.;*

Options C and E are incorrect since the keywords are in a wrong order. Option D is incorrect since it misses the *static* keyword.

Q41 **The answer is D.**

Arrays.asList() method creates a fixed size List so that they cannot be altered except changing the values of elements. So the *remove()* method throws an exception at runtime.

Q42 The answers are A, D.

Identifier names can contain letters,, numbers, $ and _. But a number cannot be used at the beginning, which makes the option A correct.

Numerals can have underscores between digits, which makes the option D correct.

int is a reserved type, it can not be changed.

Q43 The answer is B.

args is assigned a new String array object of size 4 and sorted later in alphabetical order. *Arrays.toString()* method prints the array in a neat way, which will be [0, 01, 1, 10].

Q44 The answer is D.

In order to access a variable from anywhere it must be *public*. And in order to access a variable without having an instance of the class, it must be *static*. So the answer is D.

Q45 The answer is A.

The code compiles and prints only one line since there is neither *println()* method, nor a new line character in for loops.

Q46 The answers are C, D.

A – javac command compiles source code into a bytecode, not into a machine code.

B – java command executes .class files.

C – javac command compiles .java files.

D – javac command compiles source code into bytecode.

E – java command does not compile .java files.

F – javac command compiles .java files.

Q47 The answer is