04 Java
Loop Constructs

Problema 1

What will the following code print when compiled and run?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        A: for(int i = 0; i < 2; i++){
            B: for(int j = 0; j < 2; j++){
                C: for(int k = 0; k < 3; k++){
                    c++;
                    if(k>j) break;
                }
            }
        }
        System.out.println(c);
    }
}
```

You have to select 1 option:

- 7
- 8
- 9
- [✓ 10]
- 11

Explanation:
The point to note here is that a break without any label breaks the innermost outer loop. So in this case, whenever k>j, the C loop breaks. You should run the program and follow it step by step to understand how it progresses.

Problema 2

Which of the following code snippets will compile without any errors?
(Assume that the statement int x = 0; exists prior to the statements below.)

You have to select 3 options:

- `while (false) { x=3; }`
- `if (false) { x=3; }`
- `do{ x = 3; } while(false);`
    - In a do- while, the block is ALWAYS executed at least once because the condition check is done after the block is executed. Unlike a while loop, where the condition is checked before the execution of the block.
- `for( int i = 0; i< 0; i++) x = 3;`

Explanation:
while (false) { x=3; } is a compile-time error because the statement x=3; is not reachable;
Similarly, for(int i = 0; false; i++) x = 3; is also a compile time error because x= 3 is unreachable. In if(false){ x=3; }, although the body of the condition is unreachable, this is not an error because the JLS explicitly defines this as an exception to the rule. It allows this construct to support optimizations through the conditional compilation. For example, if(DEBUG){ System.out.println("beginning task 1"); } Here, the DEBUG variable can be set to false in the code while generating the production version of the class file, which will allow the compiler to optimize the code by removing the whole if statement entirely from the class file.

Problema 3

What will the following code print when compiled and run?

```
public class TestClass{
   public static void main(String[] args){
        int[] arr = { 1, 2, 3, 4, 5, 6 };
        int counter = 0;
        for (int value : arr) {
           if (counter >= 5) {
               break;
           } else {
            continue;
           }
           if (value > 4) {
             arr[counter] = value + 1;
           }
           counter++;
        }
        System.out.println(arr[counter]);
   }
}
```

You have to select 1 option

- It will not compile.
    - Observe that the line if (value > 4) { and the rest of the code in the for loop will not execute in any case. It is therefore unreachable code and the compiler will complain about it.
- It will throw an exception at run time.
- 5
- 6
- 7
- 8

Problema 4

Consider the following code:

```
public static void main(String[] args) {
   int[] values = { 10, 30, 50 };
        for( int val : values ){
            int x = 0;
            while(x<values.length){
             System.out.println(x+" "+val);
              x++;
            }
        }
}
```

How many times is 2 printed out in the output?
You have to select 1 option

- 0
- 1
- 2
- [✓ 3]

This is a simple while loop nested inside a for loop.
The for loop loops three times - once for each value in values array.
Since, values.length is 3, x is incremented two times for each for loop iteration before the condition x<values.length returns false.
Therefore, it prints:

```
0 10
1 10
2 10
0 30
1 30
2 30
0 50
1 50
2 50
```

Problema 5

What will the following program print?

```
class LoopTest{
    public static void main(String args[]) {
        int counter = 0;
        outer:
        for (int i = 0; i < 3; i++) {
            middle:
            for (int j = 0; j < 3; j++) {
                inner:
                for (int k = 0; k < 3; k++) {
                    if (k - j > 0) {
                        break middle;
                    }
                    counter++;
                }
            }
        }
        System.out.println(counter);
    }
}
```

You have to select 1 option:

- 2
- [✓ 3]
- 6
- 7
- 9

Explanation:
To understand how this loop works let us put some extra print statements in the innermost loop:
System.out.println("i="+i+" j="+j+" k="+k);

```
if(k-j>0){
     System.out.println("breaking middle "+j);
     break middle;
}
counter++;
```

```
This is what it prints:   breaking middle 0     breaking middle 0     breaking middle 0
i=0 j=0 k=0               i=1 j=0 k=0           i=2 j=0 k=0           3
i=0 j=0 k=1               i=1 j=0 k=1           i=2 j=0 k=1
```

The key is that the middle loop is broken as soon as k-j becomes > 0. This happens on every second iteration of inner loop when k is 1 and j is 0. Now, when middle is broken inner cannot continue. So the next iteration of outer starts.

Problema 6

Which of the following code fragments compile without any error?
Assume that Math.random() returns a double between 0.0 and 1.0 (not including 1.0).
You have to select 3 options

```
for(;Math.random()<0.5;){
    System.out.println("true");
}
```

The second expression in a for loop must return a boolean, which is happening here. So this is valid.

```
for(;;Math.random()<0.5){
    System.out.println("true");
}
```

Here, the first part (i.e. the init part) and the second part (i.e. the expression/condition part) part of the for loop are empty. Both are valid. (When the expression/condition part is empty, it is interpreted as true.)
The third part (i.e. the update part) of the for loop does not allow every kind of statement. It allows only the following statements here: Assignment, PreIncrementExpression, PreDecrementExpression, PostIncrementExpression, PostDecrementExpression, MethodInvocation, and ClassInstanceCreationExpression. Thus, Math.random()<0.5 is not valid here, and so this will not compile.

```
for(;;Math.random()){
     System.out.println("true");
}
```

This is a valid never ending loop that will keep printing true.

```
for(;;){
    Math.random()<.05? break : continue;
}
```

This is an invalid use of ? : operator. Both sides of : should return some value. Here, break and continue do not return anything. However, the following would have been valid: for(;Math.random()<.05? true : false;){ }

```
for(;;){
    if(Math.random()<.05) break;
}
```

El bucle for(;;) es una forma de crear un bucle infinito en Java. En este caso, no se especifican condiciones de inicio, condición de continuación o expresiones de incremento, por lo que el bucle se ejecutará de forma indefinida hasta que se alcance una condición de salida, que es cuando Math.random() < 0.05 se evalúa como true.

The three parts of a for loop are independent of each other. However, there are certain rules for each part.

Problema 7

What will the following program print?

```
class Test{
    public static void main(String args[]){
        int c = 0;
        boolean flag = true;
        for(int i = 0; i < 3; i++){
            while(flag){
                c++;
                if(i>c || c>5) flag = false;
            }
        }
        System.out.println(c);
    }
}
```

You have to select 1 option

- 3
- 4
- 5
- [✓ 6]
- 7

Explanation:
In the first iteration of for loop, the while loop keeps running till c becomes 6. Now, for all next for loop iteration, the while loop never runs as the flag is false. So final value of c is 6.

Problema 8

What can be inserted in the following code so that it will print exactly 2345 when compiled and run?

```
public class FlowTest {

    static int[] data = {1, 2, 3, 4, 5};

    public static void main(String[] args) {
        for (int i : data) {
            if (i < 2) {
                //insert code1 here
            }
            System.out.print(i);
            if (i == 3) {
                //insert code2 here
            }
        }
    }
}
```

You have to select 2 options

```
break;
and
//nothing is required
```

[✓
```
continue;
and
continue;
```
]

```
break;
and
break;
```

[✓
```
continue;
and
//nothing is required
```
]

```
break;
and
continue;
```

Explanation:
This is a very simple loop to follow if you know what break and continue do. break breaks the nearest outer loop. Once a break is encountered, no further iterations of that loop will execute. continue simply starts the next iteration of the loop. Once a continue is encountered, rest of the statements within that loop are ignored (not executed) and the next iteration is started.

Problema 9

What will the following code print when run?

```
public class TestClass {
    public static void main(String[] args) throws Exception {
        String[] sa = {"a", "b", "c"};
        for(String s :  sa){
            if("b".equals(s)) continue;
            System.out.println(s);
            if("b".equals(s)) break;
            System.out.println(s+" again");
        }
    }
}
```

You have to select 1 option

[✓
```
a
a again
c
c again
```
]

```
a
a again
b
b again
```

```
a
a again
b
```

```
c
c again
```

Explanation:
To determine the output you have to run through the loop one iteration at a time in your mind:

Iteration 1: s is "a". It is not equal to "b" so, it will print "a", and then "a again".

Iteration 2: s is "b". It is equal to "b", so the first if will execute "continue", which mean the rest of the code in the loop will not be executed (thus b and b again will not be printed), and the next iteration will start. Note that the second if is not executed at all because of the continue in the first if.

Iteration 3: s is "c", both the if conditions are not satisfied. So "c" and "c again" will be printed.

Problema 10

What will be the result of attempting to compile and run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int x  = 0;
      labelA:   for (int i=10; i<0; i--){
         int j = 0;
         labelB:
         while (j < 10){
            if (j > i) break labelB;
            if (i == j){
               x++;
               continue labelA;
            }
            j++;
         }
         x--;
      }
      System.out.println(x);
   }
}
```

You have to select 1 option:

- It will not compile.
- It will go in infinite loop when run.
- The program will write 10 to the standard output.
- The program will write 0 to the standard output.
- None of the above.

Explanation:
This is just a simple code that is meant to confuse you. Notice the for statement: for(int i=10; i<0; i--). i is being initialized to 10 and the test is i<0, which is false. Therefore, the control will never get inside the for loop, none of the weird code will be executed, and x will remain 0, which is what is printed.

Problema 11

You have been given an array of objects and you need to process this array as follows -
1. Call a method on each object from first to last one by one.
2. Call a method on each object from last to first one by one.
3. Call a method on only those objects at even index (0, 2, 4, 6, etc.)

Which of the following are correct?
You had to select 1 option

- Enhanced for loops can be used for all the three tasks.
- Enhanced for loop can be used for only the first task. For the rest, standard for loops can be used.
- Standard for loops can be used for tasks 1 and 2 but not 3.
- All the tasks can be performed either by using only standard for loops or by using only enhanced for loops.
- Neither standard for loops nor enhanced for loops can be used for all three tasks.

Explanation:
The enhanced for loop is tailor made for processing each element of a collection (or an array) in order. Most importantly, it does not give you an iterating variable that you can manipulate and that makes it impossible to change the order or to skip an element. Therefore, tasks 2 and 3 cannot be done by an enhanced for loop.

The standard for loop is very flexible. It can do pretty much anything. Here is how you can do task 2 and 3 using a standard for loop -
//processing in reverse

```
for(int i=arr.length-1; i>=0; i--){
   arr[i].m1();
}
```

//processing alternate

```
for(int i=0; i<arr.length; i=i+2){
   arr[i].m1();
}
```

Problema 12

How many times will the line marked //1 be called in the following code?

```
int x = 10;
do{
 x--;
 System.out.println(x);  // 1
}while(x<10);
```

You had to select 1 option

- 0
- 1
- 9
- 10
- None of these.

Explanation:
// entra a un bucle infinito ya que por el 'do while' empezará si o si una vez lo que está dentro y entonces x siempre será menor que 10, no saldrá del bucle hasta que se llegue al límite inferior del tipo int y en ese momento pasará al límite superior del int siendo un número positivo y saldrá del bucle. Ese límite es de millones por lo que no es ninguna de las opciones (La opción de ninguna de estas)

A do-while loop is always executed at least once. So in the first iteration, x is decremented and becomes 9. Now the while condition is tested, which returns true because 9 is less than 10. So the loop is executed again with x = 9. In the loop, x is decremented to 8 and the condition is tested again, which again returns true because 8 is less than 10.

As you can see, x keeps on decreasing by one in each iteration and every time the condition x<10 returns true. However, after x reaches -2147483648, which is its MIN_VALUE, it cannot decrease any further and at this time when x-- is executed, the value rolls over to 2147483647, which is Integer.MAX_VALUE. At this time, the condition x<10 fails and the loop terminates.

Problema 13

Consider the following code snippet:

```
    for(int i=INT1; i<INT2; i++){
        System.out.println(i);
    }
```

INT1 and INT2 can be any two integers.

Which of the following will produce the same result?
//El Print algo, es porque se piensa que INT1=1 e INT2=3

- `for(int i=INT1; i<INT2; System.out.println(++i));`
  - Prints: 2 and 3
- `for(int i=INT1; i++<INT2; System.out.println(i));`
  - Prints: 2 and 3
- `int i=INT1; while(i++<INT2) { System.out.println(i); }`
  - Prints: 2 and 3
- `int i=INT1; do { System.out.println(i); }while(i++<INT2);`
  - Prints: 1 2 and 3
- None of these.

Explanation:
In such a question it is best to take a sample data such as INT1=1 and INT2=3 and execute the loops mentally.

Eliminate the wrong options. In this case, the original loop will print:

```
=====ORIGINAL ====
1
2
```

Outputs of all the options are given above (Ignoring the line breaks).

Thus, none of them is same as the original.

Problema 14

What will the following code print when compiled and run?

```
public class DaysTest{

    static String[] days = {"monday", "tuesday", "wednesday", "thursday",
    "friday", "saturday", "sunday" };

    public static void main(String[] args) {

        int index = 0;
        for(String day : days){

            if(index == 3){
                break;
            }else {
                 continue;
            }
            index++;
            if(days[index].length()>3){
                days[index] = day.substring(0,3);
            }
        }
        System.out.println(days[index]);
    }
}
```

You had to select 1 option

- mon
- thu
- fri
- [✓ It will not compile.]
- It will throw an exception at run time.

Explanation:
//nunca entra al else y el compilador te diría que es innecesario, NO COMPILA.
Notice the statement :

```
if(index == 3){
                 break;
             }else {
                  continue;
             }
```

In no situation can the control go beyond this statement in the for loop. Therefore, rest of the statements in the for loop are unreachable and so the code will not compile.

Problema 15

Which of these statements are valid when occurring by themselves in a method?
You had to select 3 options

- while ( ) break ;
    - The condition expression in a while header is required.
- [✓ do { break ; } while (true) ;]
- if (true) { break ; } (When not inside a switch block or a loop)
    - You cannot have break or continue in an 'if' or 'else' block without being inside a loop. Note that the problem statement mentions, "...occuring by themselves". This implies that the given statement is not wrapped within any other block. Note: break with a label is possible in an if/else statement without a loop: label: if(true){ System.out.println("break label");   break label; //this is valid   }
- [✓ switch (1) { default : break; }]
    - You can use a constant in switch(...);
- [✓ for (; true ;) break ;]

Explanation:
// Tal cual, se requiere la expresión del while, y el true puede tenerlo pero si está dentro de un ciclo.

It is not possible to break out of an if statement. But if the if statement is placed within a switch statement or a loop construct, the usage of break in option 3 would be valid.

Problema 16

What will the following program print?

```
class Test{
   public static void main(String args[]){
      int var = 20, i=0;
      do{
         while(true){
         if( i++ > var) break;
         }
      }while(i<var--);
      System.out.println(var);
   }
}
```

You had to select 1 option

- [✓ 19]
- 20
- 21
- 22
- It will enter an infinite loop.

Explanation:
// El pedo es ese if,peor la explicación es:
// Cuando la condición del if se evalúa, primero se compara el valor actual de i con var. Luego, i se incrementa en 1 después de la comparación.
// Cuando i=20 => `( i++ > 20 )` => `false` entonces itera otra vez el `while` y suma i = 21
// Cuando i=21 => `( i++ > 20 )` => `true` entonces entra al `if`, sale del `while` infinito y suma
// `i=22`

When the first iteration of outer do-while loop starts, var is 20. Now, the inner loop executes till i becomes 21. Now, the condition for outer do-while is checked, while(22 < 20), [i is 22 because of the last i++>var check], thereby making var 19. And as the condition is false, the outer loop also ends. So, 19 is printed.

Problema 17

Consider the following class :

```
class Test{
   public static void main(String[] args){
      for (int i = 0; i < 10; i++) System.out.print(i + " ");  //1
      for (int i = 10; i > 0; i--) System.out.print(i + " ");  //2
      int i = 20;                  //3
      System.out.print(i + " ");   //4
   }
}
```

Which of the following statements are true?
You had to select 4 options

- [✓ As such, the class will compile and print "20 " (without quotes) at the end of its output.]
- [✓ It will not compile if line 3 is removed.]
    - If //3 is removed, 'i' will be undefined for //4
- [✓ It will not compile if line 3 is removed and placed before line 1.]
- [✓ It will not compile if line 4 is removed and placed before line 3.]
- Only Option 2, 3, and 4 are correct.

Explanation:
The scope of a local variable declared in 'for' statement is the rest of the 'for' statement, including its own initializer.
So, when line 3 is placed before line 1, there is a redeclaration of i in the first for() which is not legal.

As such, the scope of i's declared in for() is just within the 'for' blocks. So placing line 4 before line 3 will not work since 'i' is not in scope there.

Problema 18

Which of these for statements are valid?

```
1. for (int i=5; i=0; i--) { }

2.  int j=5;
      for(int i=0, j+=5;  i<j ; i++) {  j--;  }

3. int i, j;
    for (j=10; i<j; j--) { i += 2; }

4. int i=10;
    for ( ; i>0 ; i--) { }

5. for (int i=0, j=10; i<j; i++, --j) {;}
```

You had to select 1 option

- 1, 2
- 3, 4
- 1, 5
- 1 is not valid.
- [✓ 4, 5]
- 5

No 1. uses '=' instead of '==' for condition which is invalid. The loop condition must be of type boolean.

No 2. uses 'j +=5'. Now, this statement is preceded by 'int i=0,' and that means we are trying to declare variable j. But it is already declared before the for loop. If we remove the int in the initialization part and declare i before the loop then it will work. But if we remove the statement int j = 5; it will not work because compiler will try to do j = j+5 and you can't use the variable before it is initialized. Although the compiler gives a message 'Invalid declaration' for j += 5, it really means the above mentioned thing.

No 3. i is uninitialized.

No 4. is valid. It contains empty initialization part.

No 5. This is perfectly valid. You can have any number of comma separated statements in initialization and incrementation part. The condition part must contain a single expression that returns a boolean. All a for loop needs is two semi colons :- for(; ;) {} This is a valid for loop that never ends. A more concise form for the same is : for(; ;);

Problema 19

What will be the output if you run the following program?

```
public class TestClass{
   public static void main(String args[]){
      int i;
      int j;
      for (i = 0, j = 0 ; j < 1 ; ++j , i++){
         System.out.println( i + " " + j );
      }
      System.out.println( i + " " + j );
   }
}
```

You had to select 1 option

- 0 0 will be printed twice.
- 1 1 will be printed once.
- 0 1 will be printed followed by 1 2.
- [✓ 0 0 will be printed followed by 1 1.]
- It will print 0 0 and then 0 1.

Explanation:
j will be less than 1 for only the first iteration. So, first it will print 0, 0. Next, i and j are incremented. Because j is not less than 1 at the start of the loop, the condition fails and it comes out of the loop. Finally, it will print 1,1.

Problema 20

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      boolean b = false;
      int i = 1;
      do{
         i++ ;
      } while (b = !b);
      System.out.println( i );
   }
}
```

You had to select 1 option

- The code will fail to compile, 'while' has an invalid condition expression.
    - It is perfectly valid because b = !b; returns a boolean, which is what is needed for while condition.
- It will compile but will throw an exception at runtime.
- It will print 3.
    - The loop body is executed twice and the program will print 3.
- It will go in an infinite loop.
- It will print 1.

Explanation:
Unlike the 'while(){}' loop, the 'do {} while()' loop executes at least once because the condition is checked after the iteration.

Problema 21

Given:

```
package loops;
public class JustLooping {
    private int j;
    void showJ(){
        while(j<=5){
            for(int j=1; j <= 5;){
                System.out.print(j+" ");
                j++;
            }
            j++;
        }
    }
    public static void main(String[] args) {
        new JustLooping().showJ();
    }
}
```

What is the result?
You had to select 1 option

- It will not compile.
    - There is no problem with the code. The variable j declared in the for loop shadows the instance member j inside the for loop.
- It will print 1 2 3 4 5 five times.
- It will print 1 3 5 five times.
- It will print 1 2 3 4 5 once.
- It will print 1 2 3 4 5 six times.

Explanation:
The point to note here is that the j in for loop is different from the instance member j. Therefore, j++ occuring in the for loop doesn't affect the while loop. The for loop prints 1 2 3 4 5.

The while loop runs for the values 0 to 5 i.e. 6 iterations. Thus, 1 2 3 4 5 is printed 6 times. Note that after the end of the while loop the value of j is 6.

Problema 22

What will be the result of attempting to compile and run the following program?

```
class TestClass{
   public static void main(String args[]){
      int i = 0;
      for (i=1 ;  i<5  ; i++) continue;  //(1)
      for (i=0 ;       ; i++) break;       //(2)
      for (    ; i<5?false:true ;    );     //(3)
   }
}
```

You had to select 1 option

- [✓ The code will compile without error and will terminate without problem when run.]
- The code will fail to compile, since the continue can't be used this way.
- The code will fail to compile, since the break can't be used this way.
- The code will fail to compile, since the for statement in line 2 is invalid.
- The code will compile without error but will never terminate.
    - the condition part is 'false' so the control will never go inside the loop.

Explanation:
A continue statement can occur in and only in loops i.e. for/enhanced-for, while, and do-while loop. A continue statement means: Forget about the rest of the statements in the loop and start the next iteration.
So,

`for (i=1 ;  i<5  ; i++) continue;` just increments the value of i up to 5 because of i++.

`for (i=0 ;       ; i++) break;` iterates only once because of the break so the value of i becomes 0.

`for (    ; i<5?false:true ;    );` never iterates because i is less than 5 (it is 0 because of //2) and the condition expression is false!

At the end of the code, the value of i is 0.

Problema 23

What will the following code print?

```
public class BreakTest{
  public static void main(String[] args){
    int i = 0, j = 5;
    lab1 : for( ; ; i++){
      for( ; ; --j)  if( i >j ) break lab1;
    }
    System.out.println(" i = "+i+", j = "+j);
  }
}
```

You had to select 1 option

- `i = 1, j = -1`
- `i = 1, j = 4`
- `i = 0, j = 4`
- [✓ `i = 0, j = -1`]
- It will not compile.

Explanation:
The values of i and j in the inner most for loop change as follows:

```
i = 0 j = 5
i = 0 j = 4
i = 0 j = 3
i = 0 j = 2
i = 0 j = 1
i = 0 j = 0
i = 0 j = -1
```

Therefore, the final println prints `i = 0, j = -1`