

Exercise 5.-

Consider the following Java code snippet:

```
public int divide (int a, int b){
    int c= -1;

    try{
        c = a/b;
    }
    catch(Exception e){
        System.err.print("Exception ");
    }
    finally{
        System.err.println("Finally ");
    }
    return c;
}
```

What will our code print when we call divide (4,0)?

Pick one of the choices:

- Exception Finally
- Finally Exception
- Exception

Explicacion:

El método divide intenta dividir el valor de a por b. En este caso, b es 0, lo que provoca una excepción de división por cero (ArithmeticException). El bloque catch(Exception e) captura cualquier excepción y ejecuta la instrucción System.err.print("Exception "), que imprime "Exception". Luego, el bloque finally se ejecuta independientemente de si se lanzó una excepción o no, e imprime "Finally". Por lo tanto, el resultado final será "Exception Finally".

Exercise 6.-

The feature which allows different methods to have the same name and arguments type, but the different implementation is called?

Pick one of the choices:

- Overloading(SobreCarga)
- Overriding (SobreEscritura @Override)
- Java does not permit methods with same and type signature
- None of the above

Explicacion:

Si buscamos que dos métodos tengan el mismo nombre y argumentos, pero implementados diferentes estamos hablando de SobreEscritura.

Exercise 7.-

What does the following for loop output?

```
for (int i=10, j=1, i>j; --i, ++j)
System.out.print(j %i);|
```

Pick one of the choices:

- 12321
- 12345
- 11111
- 00000

Explicacion:

Obviando el hecho de que no compila por que falta un ;, ya que no es una opcion en las respuestas, al ejecutar el loop, este nos da 1 2 3 4 5, ya que después de terminar el loop es cuando ejecuta la parte de la derecha de el for, ocasionando que no tengamos que tomar en cuenta realmente el que la resta se haga previo por el --i.

Exercise 11.-

We perform the following sequence of actions:

1. Insert the following elements into a set: 1,2,9,1,2,3,1,4,1,5,7.
2. Convert the set into a list and sort it in ascending order.

Which option denotes the sorted list?

Pick one of the choices:

- {1, 2, 3, 4, 5, 7, 9}
- {9, 7, 5, 4, 3, 2, 1}
- {1, 1, 1, 1, 2, 2, 3, 4, 5, 7, 9}
- None of the above

Explicación:

Un set no permite tener elementos duplicados por lo que al pasarlo a una lista y ordenarla con sort, se mostrará simplemente una sola vez los números, aunque los agregaramos varias veces al set.

Exercise 14.-

What is the output for the below Java code?

```
public class Test{
    public static void main (String[] args)
    {
        int i = 010;
        int j = 07;
        System.out.println(i);
        System.out.println(j);
    }
}
```

Pick one of the choices:

- 8 7
- 10 7
- Compilation fails with an error at line 3
- Compilation fails with an error at line 5

Explicación:

Cuando se crea un int con un 0 previo al numero, esta haciendo referencia a que pertenece al sistema octal, y por esto mismo, el valor 10 equivale al 8, y el 7 se conserva como tal simplemente quitando el 0.

Exercise 15.-

A public data member with the same name is provided in both base as well as derived classes. Which of the following is true?

Pick one of the choices:

- It is a compiler error to provide a field with the same name in both base and derived class
- The program will compile and this feature is called overloading
- The program will compile and this feature is called overriding
- The program will compile and this feature is called as hiding or shadowing

Explicación:

Esto se puede realizar, y como dice la respuesta a esta feature se le llama hiding o shadowing, sin embargo es importante mencionar que si se declara una Superclase como variable de referencia y se le pasa una subClase como objeto, al momento de intentar acceder a este dato, nos regresara siempre el del padre, o dicho de otra forma, el de la variable de referencia, a diferencia de si declaramos un método lo mandamos a llamar con el objeto de una subClase.

Exercise 16.-

Given three clases A, B and C.

B is a subclass of A

C is a subclass of B

Which one of these boolean expressions is true only when an object denoted by reference o has actually been instantiated from class B, as opposed to from A or C?

Pick one of the choices:

- (o instanceof B) && (! (o instanceof A))
- (o instanceof B) && (! (o instanceof C))
- (o instanceof B)
- None of the above

Explicacion:

Ya que tanto B como C son instancias de A, al ser sus subClases, al realizar la primera, nunca entrara a la validación y siempre retornara false, sin embargo, si mencionamos que sea instancia de B, pero no de C que es su su clase hija, entonces resolvemos el problema.

Exercise 17.-

Which statement is true?

Pick one of the choices:

- Non-static member classes must have either default or public accessibility
- All nested classes can declare static member classes
- Methods in all nested classes can be declared static
- Static member classes can contain non-static methods

Explicacion:

Las clases miembro estáticas pueden contener métodos no estáticos. De hecho, pueden contener tanto métodos estáticos como no estáticos.

Exercise 18.-

A constructor is called whenever

Pick one of the choices:

- An object is declared
- An object is used
- A class is declared
- A class is used

Explicacion:

Al declarar un objeto por defecto lo hacemos mediante el constructor del mismo, por lo que decir que el constructor es llamado cuando el objeto es declarado es correcto.

Exercise 19.-

You are implementing a student registration and student information retrieval system for a school using a simple class roster in Java. When a student is registered, the system must assign an integer ID(enrollmentNumber), starting at 1 and adding 1 as each student is registered. The student's name is stored with the assigned enrollmentNumber. The retrieval request should return a student's registration information.

The student class should implement:

- The constructor: Student (String name)
- The method String toString() to return the string "(enrollmentNumber):(name)"

The locked stub code in the editor validates the implementation of the Student class.

After each student is registered, the code stub requests and prints the student's information to test your code.

Constraints

- $1 \leq \text{numberOfStudents} \leq 10^3$

Sample Input For Custom Testing

```
3
Erica
Bob
Maria
```

Sample Output

```
1: Erica
2: Bob
3: Maria
```

Explanation

The three students are registered in the following order:

- The first student to be registered is Erica so she is assigned 1 as the enrollment number by the portal.
- Bob is second, so he is assigned the number 2.
- Maria is third, so she is assigned the number 3.

Now, the information of all the students is printed in the order in which they are registered.

```

public class Student {
    private static int enrollmentCounter = 1;
    private int enrollmentNumber;
    private String name;

    public Student(String name) {
        this.name = name;
        this.enrollmentNumber = enrollmentCounter++;
    }

    @Override
    public String toString() {
        return enrollmentNumber + ": " + name;
    }

    public static void main(String[] args) {
        // Ejemplo de cómo se utilizaría la clase
        Student s1 = new Student("Erica");
        Student s2 = new Student("Bob");
        Student s3 = new Student("Maria");

        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);
    }
}

```

Exercise 20.-

Which of the following data types in Java are primitive?

Pick one of the choices:

- String
- Struct
- Boolean
- char

Explicación:

La forma más sencilla de saber si una variable es primitiva o no, es mediante la primera letra, ya que los primitivos comienzan en minúscula, y los Wrappers así como las clases comienzan con Mayúscula.

Exercise 21.-

Which of the following are true for Java Classes?

Pick one of the choices:

- The Void class extends the Class class
- The Float class extends the Double class
- The System class extends the Runtime class
- The Integer class extends the Number class

Explicación:

La mayoría de Wrappers en Java (entiéndase por wrappers como la versión Clase de los primitivos en Java), extienden de la clase Number, entre ellos Integer.

Exercise 22.-

The following code snippet is a demonstration of a particular design pattern. Which design pattern is it?

```
public class Mystery{
    private static Mystery instance = null;
    protected Mystery(){
        public static Mystery getInstance(){
            if(instance == null){
                instance = new Mystery();
            }
            return instance;
        }
    }
}
```

Pick one of the choices:

- Factory Design Pattern
- Strategy Pattern
- Singleton
- Facade Design Pattern

Explicación:

Si nos fijamos al realizar el código solamente se realizará una instancia de Mystery, y posterior a esto, siempre que lo solicitamos nos devolverá esta única instancia, por lo que encaja con lo que es el patrón Singleton.

Exercise 23.-

Which of the following Java declaration of the String array is correct?

Pick one of the choices:

- `String temp [] = new String {"j", "a", "z"};`
- `String temp [] = {"j" "b" "c"};`
- `String temp = {"a", "b", "c"};`
- `String temp [] = {"a", "b", "c"};`

Explicación:

Si queremos crear un array de un String, necesitamos declararlo con comillas y entre brackets después del igual, así como separar por comas, además es necesario decirle a Java que será un array colocando [] en cualquier lugar entre frases después del String y antes del =.

Exercise 24.-

Which is true of the following program?

```

1 package exam.java;
2
3 public class TestFirstApp {
4     static void doIt(int x, int y, int m) {
5         if(x==5) m=y;
6         else m=x;
7     }
8
9     public static void main(String[] args) {
10         int i=6, j=4, k=9;
11         TestFirstApp.doIt(i, j, k);
12         System.out.println(k);
13     }
14 }
```

Pick one of the choices

- Doesn't matter what the values of *i* and *j* are, the output will always be 5.
- Doesn't matter what the values of *k* and *j* are, the output will always be 5.
- Doesn't matter what the values of *i* and *j* are, the output will always be 9.
- Doesn't matter what the values of *k* and *j* are, the output will always be 9.

Explicación:

A pesar de que el código funciona correctamente, y en efecto esta mandando a llamar a una función, esta no regresa nada, y no está cambiando el valor de las variables que le pasamos en el scope del main, por lo que *k* nunca fue modificada e imprimirá el valor que le pasamos originalmente, el cual es 9.

Exercise 25.-

Which of the following statements are correct. Select the correct answer.

- Each Java file must have exactly one package statement to specify where the class is stored.
- If a java file has both import and package statement, the import statement must come before package statement.
- A java file has at least one class defined
- If a java file has a package statement, it must be the first statement (except comments).

Explicación:

Aunque tendría ninguna función, es posible contar con un programa Java que no tenga la declaración de package, sin embargo, tampoco tendría otra cosa, por lo que no sería útil, sin embargo si lo tuviera debe ir en efecto como la primera línea de código que aparezca, sin contar los comentarios.

Exercise 26.-

What is the output for the following program:

```
class Constructor
{
    static String str;
    public void Constructor() {
        System.out.println("In constructor");
        str="Hello World";
    }

    public static void main(String[] args) {

        Constructor c= new Constructor();
        System.out.println(str);
    }
}
```

Pick one of the choices

- In Constructor
- Null
- Compilation Fails
- None of the above

Explicación:

En ningún momento estamos agregando valor a str, ya que "Constructor" es un método y no un constructor como tal, por lo que al mandarlo a llamar desde el método main, nos regresará un null.

Exercise 27.-

Given the following code, what is the most likely result.

```
import java.util.*;

public class Compares {

    public static void main(String[] args) {

        String [] cities= {"Bangalore","Pune","San Francisco","New York City"};
        MySort ms= new MySort();
        Arrays.sort(cities,ms);
        System.out.println(Arrays.binarySearch(cities,"New York City" ));
    }

    static class MySort implements Comparator{
        public int compare(String a, String b){

            return b.compareTo(a);
        }
    }

}
```

Pick one of the choices

- -1
- 1
- 2
- Compilation fails

Explicacion:

El código define una clase MySort que implementa la interfaz Comparator para comparar cadenas (String). En el método compare, las cadenas a y b se comparan usando b.compareTo(a), lo que invierte el orden natural de las cadenas. Por lo tanto, el arreglo cities se ordena en orden descendente: "San Francisco", "Pune", "New York City", "Bangalore".

Después, se realiza una búsqueda binaria (binarySearch) para encontrar la posición de "New York City". Como el arreglo está ordenado en orden descendente, "New York City" se encuentra en la posición 2 (contando desde 0). Sin embargo, como el índice se basa en una comparación en un orden invertido, el método binarySearch devuelve el índice 1 porque ese es el lugar donde se espera encontrar "New York City" si estuviera en orden ascendente.

Por tanto, la respuesta final que se imprime es 1.

Exercise 28.-

Anna has an array of n integers called `num`. She can reduce the array by 1 element by performing a move. Each move consists of the following three steps:

1. Pick two different elements `num[i]` and `num[j]`.
2. Remove the two selected elements from the array.
3. Add the sum of the two selected elements to the end of the array.

Each move has a cost associated with it and this cost is equal to the sum of the two elements removed from the array during the move. Anna wishes to calculate the minimum total cost of reducing her array to one element.

For example, consider the array `num = [4,6,8]`. Anna removes 4 and 6 in her first move at a cost of $4 + 6 = 10$, and the resultant array is `num1 = [10,8]`. She then removes 10 and 8 in her second move at a cost of $10 + 8 = 18$, and the resultant array is `num2 = [18]`. The total cost of reducing this array to one element using this sequence of moves is $10 + 18 = 28$. This is just one set of possible moves. For instance, she could have started with 4 and 8.

Function Description

Complete the function `reductionCost` in the editor below. The function must return the minimum total cost of reducing the input array to one element

`reductionCost` has the following parameter(s):

`num[num[0],...num[n-1]]`: an array of integers

Constraints

- $2 \leq n \leq 10^4$
- $0 \leq \text{num}[i] \leq 10^5$

```

3 import java.util.PriorityQueue;
4
5 public class Solution {
6     1 usage
7     public static int reductionCost(int[] num) {
8         // Crear un min-heap (PriorityQueue en Java)
9         PriorityQueue<Integer> minHeap = new PriorityQueue<>();
10
11         // Añadir todos los elementos del array al min-heap
12         for (int n : num) {
13             minHeap.add(n);
14         }
15
16         int totalCost = 0;
17
18         // Continuar hasta que solo quede un elemento en el heap
19         while (minHeap.size() > 1) {
20             // Extraer los dos elementos más pequeños
21             int firstMin = minHeap.poll();
22             int secondMin = minHeap.poll();
23
24             // Calcular el costo y añadirlo al total
25             int cost = firstMin + secondMin;
26             totalCost += cost;
27
28             // Insertar el resultado de la suma nuevamente en el heap
29             minHeap.add(cost);
30         }
31
32         return totalCost;
33     }
34
35     public static void main(String[] args) {
36         // Ejemplo de uso
37         int[] num = {4, 6, 8};
38         System.out.println(reductionCost(num)); // Debería imprimir 28
39     }
40 }

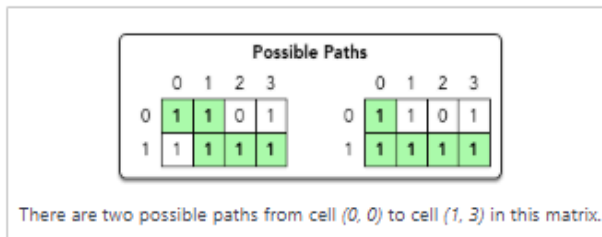
```

Exercise 29.-

Consider a maze mapped to a matrix with an upper left corner at coordinates (row, column) = (0, 0). Any movement must be in increasing row or column direction. Determine the number of distinct paths through the maze. Always start at position (0, 0), the top left. and end up at (max(row), max(column)), the bottom right.

As an example, consider the following diagram where 1 indicates an open cell and 0 indicates blocked. It is only possible to travel through open cells, so no path can go through the cell at (0, 2). There are two distinct paths to the goal.

Possible Paths



Function Description

Complete the function `numberOfPaths` in the editor below. The function must return the number of paths through the matrix, modulo $(10^9 + 7)$.

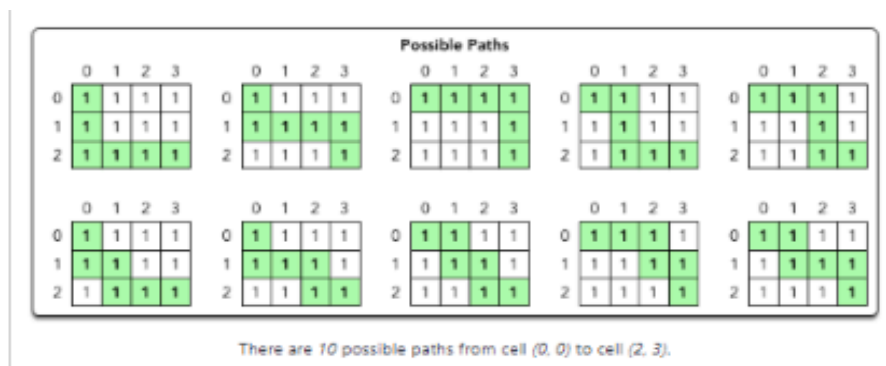
`numberOfPaths` has the following parameter(s):

`a[a[0],...a[n-1]]`: an array of strings, each a row of the matrix where each character represents a column

Constraints

- $1 \leq n, m \leq 1000$
- Each cell in matrix `a` contains either a 0 or a 1.

Explanation



```

3 ▶ public class Solution {
    1 usage
4     private static final int MOD = 1000000007;
5
    1 usage
6 @ public static int numberOfPaths(String[] a) {
7     int n = a.length;
8     int m = a[0].length();
9
10    // Matriz dp para almacenar el número de caminos hasta cada celda
11    int[][] dp = new int[n][m];
12
13    // Si la primera celda es accesible, inicializamos dp[0][0] a 1
14    if (a[0].charAt(0) == '1') {
15        dp[0][0] = 1;
16    }
17
18    // Llenar la primera fila
19    for (int j = 1; j < m; j++) {
20        if (a[0].charAt(j) == '1') {
21            dp[0][j] = dp[0][j - 1];
22        }
23    }
24
25    // Llenar la primera columna
26    for (int i = 1; i < n; i++) {
27        if (a[i].charAt(0) == '1') {
28            dp[i][0] = dp[i - 1][0];
29        }
30    }
31
32    // Llenar el resto de la matriz dp
33    for (int i = 1; i < n; i++) {
34        for (int j = 1; j < m; j++) {
35            if (a[i].charAt(j) == '1') {
36                dp[i][j] = (dp[i - 1][j] + dp[i][j - 1]) % MOD;
37            }
38        }
39    }
40

```

```

        // El número de caminos al final estará en dp[n-1][m-1]
        return dp[n - 1][m - 1];
    }

    public static void main(String[] args) {
        // Ejemplo de uso
        String[] a = {
            "1011",
            "1111",
            "1111"
        };
        System.out.println(numberOfPaths(a)); // Debería imprimir 10
    }
}

```

Exercise 30.-

Given an array of words representing your dictionary, you test each word to see if it can be made into another word in the dictionary. This will be done by removing characters one at a time. Each word represents its own first element of its string chain, so start with a string chain length of 1. Each time you remove a character, increment your string chain by 1. In order to remove a character, the resulting word must be in your original dictionary. Your goal is to determine the longest string chain achievable for a given dictionary.

For example, given a dictionary [a, and, an, bear], the word and could be reduced to an and then to a. The single character a cannot be reduced any further as the null string is not in the dictionary. This would be the longest string chain, having a length 3. The word bear cannot be reduced at all.

Function Description

Complete the function `longestChain` in the editor below. The function must return a single integer representing the length of the longest string chain.

`longestChain` has the following parameter(s):

`words[words[0]....words[n-1]]`: an array of strings to test

Constraints

- $1 \leq n \leq 50000$
- $1 \leq \text{length}(\text{word}[i]) \leq 60$, where $0 \leq i < n$
- Each `words[i]` is composed of lowercase letters in `ascii[a-z]`.

Sample Input 0

6, a, b, ba, bca, bda, bdca

Sample Output 0

4

Explanation 0

Sample Case 1: words = {"a", "b", "ba", "bca", "bda", "bdca"}

Because 'a' and 'b' are single-character words, we cannot remove any characters from them as that would result in the empty string, so the length for both of these string chains is 1.

The word "ba" can create two different string chains each with a length of 2: ("ba" — "a" and "ba" — 'b'). This means our current longest string chain is 2.

The word "bca" can create two different string chains of length 3: ('bca' --- "ba" — "a" and "bca" — "ba" — 'b'). This means our current longest string chain is now 3.

The word 'bda' can create two different string chains of length 3: ('bda' — 'ba' — "a" and "bda" — 'ba' — 'b'). At this point, our current longest string chain is still 3.

The word 'bdca' can create four different string chains of length 4 ("bdca" — 'bda' — "ba" — "a", 'bdca' — "bda" — "ba" — 'b', 'bdca' — "bca" — "ba" — 'a', and 'bdca' — "bca" — "ba" — 'b'). This means our current longest string chain is now 4.

The longest string chain is 4.


```

3 import java.util.Arrays;
4 import java.util.HashMap;
5 import java.util.Map;
6
7 public class LongestStringChain {
8
9     1 usage
10    public static int longestChain(String[] words) {
11        // Ordenar las palabras por longitud
12        Arrays.sort(words, (a, b) -> a.length() - b.length());
13
14        // Mapa para almacenar la longitud máxima de la cadena para cada palabra
15        Map<String, Integer> dp = new HashMap<>();
16
17        int maxLength = 0;
18
19        for (String word : words) {
20            int currentLength = 1;
21
22            // Intentar eliminar cada carácter de la palabra
23            for (int i = 0; i < word.length(); i++) {
24                String prevWord = word.substring(0, i) + word.substring(beginIndex: i + 1);
25                if (dp.containsKey(prevWord)) {
26                    currentLength = Math.max(currentLength, dp.get(prevWord) + 1);
27                }
28            }
29
30            dp.put(word, currentLength);
31            maxLength = Math.max(maxLength, currentLength);
32        }
33
34        return maxLength;
35    }
36
37    public static void main(String[] args) {
38        String[] words = {"a", "b", "ba", "bca", "bda", "bdca"};
39        System.out.println(longestChain(words)); // Output: 4
40    }
41

```

Exercise 31.-

You are implementing a student registration and student information retrieval system for a school using a simple class roster in Java. When a student is registered, the system must assign an integer ID (enrollmentNumber), starting at 1 and adding 1 as each student is registered. The student's name is stored with the assigned enrollmentNumber. The retrieval request should return a student's registration information.

The Student class should implement:

- The constructor `Student(String name)`
- The method `String toString()` to return the string `"{enrollmentNumber}: {name}"`

The locked stub code in the editor validates the implementation of the `Student` class.

After each student is registered, the code stub requests and prints the student's information to test your code.

Constraints

- $1 \leq \text{numberOfStudents} \leq 10^3$

Input Format For Custom Testing

The first line contains the value of `numberOfStudents` describing the total number of students being registered. Each of the next `numberOfStudents` lines contains the student name.

Sample Case 0

Sample Input For Custom Testing

3, Erica, Bob, Maria

Sample Output

1: Erica

2: Bob

3: Maria

```

3 ▶ public class Student {
4     // Atributos de la clase
5     private static int enrollmentCounter = 1; // Contador para asignar el número de matrícula
6     private int enrollmentNumber; // Número de matrícula del estudiante
7     private String name; // Nombre del estudiante
8
9     // Constructor que recibe el nombre del estudiante
10    public Student(String name) {
11        this.name = name;
12        this.enrollmentNumber = enrollmentCounter++; // Asigna el número de matrícula y lo incrementa
13    }
14
15    // Método para representar el objeto Student como cadena de texto
16    @Override
17    public String toString() {
18        return enrollmentNumber + ": " + name;
19    }
20
21    // Método principal para probar la funcionalidad
22    ▶ public static void main(String[] args) {
23        // Ejemplo de entrada
24        Student student1 = new Student(name: "Erica");
25        Student student2 = new Student(name: "Bob");
26        Student student3 = new Student(name: "Maria");
27
28        // Imprimir la información de cada estudiante
29        System.out.println(student1);
30        System.out.println(student2);
31        System.out.println(student3);
32    }
33 }
34

```

Exercise 32.-

To delete all pairs of keys and values in a given HashMap, which of the following methods should be used?

Pick one of the choices

- a) clearAll()
- b) empty()
- c) remove()
- d) clear()

Explicacion:

En Java, el método clear() se utiliza para eliminar todas las entradas (pares clave-valor) de un HashMap. Esto deja el HashMap vacío pero aún existente. Las otras opciones (clearAll(), empty(), remove()) no son métodos válidos de HashMap.

Exercise 33.-

Which pattern do you see in the code below:

```
java.util.Calendar.getInstance();
```

Pick one of the choices

- a) Singleton Pattern
- b) Factory Pattern
- c) Facade Pattern
- d) Adaptor Pattern

Explicacion:

El método getInstance, es un estándar cuando se habla de Singleton Pattern, por lo cual se puede deducir que se esta hablando de este cuando se ejecuta el código proporcionado.

Exercise 34.-

What is the output of the following program:

```
interface BaseI { void method (); }
class BaseC
{
    public void method()
    {
        System.out.println("Inside BaseC::method");
    }
}
class ImplC extends BaseC implements BaseI
{
    public static void main (String []s)
    {
        (new ImplC()).method();
    }
}
```

Pick one of the choices

- a) Null
- b) Compilation fails
- c) Inside BaseC::method
- d) None of the above

Explicacion:

El código es correcto, esa es una forma de llamar a un método de un objeto, sin necesidad de asignarle una variable de referencia al objeto que utilizamos.

Exercise 35.-

Consider the following three classes:

```
class A {}
class B extends A {}
class C extends B {}
```

Consider n object of class B is instantiated, i.e.,

```
B b = new B();
```

Which of the following Boolean expressions evaluates to true:

Pick one of the choices

- a) (b instanceof B)
- b) (b instanceof B) && !(b instanceof A)
- c) (b instanceof B) && !(b instanceof C)
- d) None of the above

Explicacion:

Tanto la opción a como la opción c son correctas. Sin embargo si quisiéramos que solamente fuera correcta si la Clase fuera B, seleccionaremos la c, de lo contrario si quisiéramos que simplemente nos retornará un true, las dos son válidas.

Exercise 36.-

What is the output of the following program:

```
class Constructor
{
    static String str;
    public void Constructor(){
        System.out.println("In constructor");
        str = "Hello World";
    }
    public static void main (String [] args){
        Constructor C = new Constructor ();
        System.out.println(str);
    }
}
```

Pick one of the choices

- a) In Constructor
- b) Null
- c) Compilation fails
- d) None of the above

Explicacion:

En ningún momento se le asigna un valor a str, por lo cual retorna null.

Exercise 37.-

☆ The Robot Class

We need a *Robot* class that can move around on a two dimensional plane. It needs to be able to change its position, report its position and report its last move as described below. Implement a *Robot* class per the following specifications:

Fields		
Data Type	Name	Description
integer	currentX	The robot's current x-coordinate in the 2D plane.
integer	currentY	The robot's current y-coordinate in the 2D plane.
integer	previousX	The robot's x-coordinate in the 2D plane prior to its most recent movement.
integer	previousY	The robot's y-coordinate in the 2D plane prior to its most recent movement.
Note: The robot's initial location is at (x, y) coordinate (0, 5).		

Parametrized constructor

Data Type	Param. Name	Description
integer	x	The value of <i>currentX</i> for the new Robot.
integer	y	The value of <i>currentY</i> for the new Robot.
Note: The robot created by this constructor is considered to have spawned at (0, 5) and moved to (currentX, currentY) so (previousX, previousY) starts as (0, 5).		

Methods				
Return Type	Method Name	Param. Type	Param. Name	Description
void	moveX	integer	dx	Move the robot from current position (x, y) to new position (x + dx, y). Remember to maintain <i>previousX</i> .
void	moveY	integer	dy	Move the robot from current position (x, y) to new position (x, y + dy). Remember to maintain <i>previousY</i> .

void	printCurrentCoordinates	no parameters	Print two space-separated integers describing the robot's current x and y coordinates.
void	printLastCoordinates	no parameters	Print two space-separated integers describing the robot's <i>previousX</i> and <i>previousY</i> coordinates. This will be called after the robot has moved from position $(0, 5)$ at least once.
void	printLastMove	no parameters	Print two space-separated values describing the robot's most recent movement: <ul style="list-style-type: none"> If the last move was <i>moveX(dx)</i>, print $x\ dx$ where x is the actual character x and dx is the distance moved in the x-direction during the last call to <i>moveX</i>. If the last move was

			<ul style="list-style-type: none"> If the last move was <i>moveY(dy)</i>, print $y\ dy$ where y is the actual character y and dy is the distance moved in the y-direction during the last call to <i>moveY</i>.
--	--	--	---

The code provided has a complete definition for a main method that creates *Robot* objects and tests the class' methods. Implement the *Robot* class according to the criteria above to pass all test cases.

Constraints

- $-100 \leq x, y, dx, dy \leq 100$

► Input Format for Custom Testing

▼ Sample Case 0

Sample Input 0

```
2
1
1
1
```

Sample Output 0

```
0 5
2 1
x 1
3 1
3 1
x 2
5 2
5 2
```

Explanation 0

- The *firstRobot* object is initially at position (0, 5), so the call to *firstRobot.printCurrentCoordinates()* prints 0 5.
- For the *secondRobot* object created with the parameterized constructor, it was created and moved from (0, 5) → (2, 1) so *secondRobot.printCurrentCoordinates()* prints 2 1. Next, we call the following sequence of methods on the *secondRobot* object:
 1. *secondRobot.moveX(1)* moves the robot 1 unit from (2, 1) → (3, 1).
 2. *secondRobot.printLastMove()* prints x 1 as its last movement was *moveX(1)*.
 3. *secondRobot.printCurrentCoordinates()* prints 3 1 because it moved from (2, 1) → (3, 1).
 4. *secondRobot.moveY(1)* moves the robot 1 unit from (3, 1) → (3, 2).
 5. *secondRobot.printLastCoordinates()* prints 3 1 because its last movement was (3, 1) → (3, 2), so the coordinates of its last location prior to the movement was (3, 1).
 - At this point, test code adds 1 to $dx \Rightarrow dx = 2$ and subtracts 1 from $dy \Rightarrow dy = 0$.
 6. *secondRobot.moveX(2)* moves the robot from the (3, 2) → (5, 2).
 7. *secondRobot.printLastMove()* prints x 2 as its last movement was *moveX(2)*.
 8. *secondRobot.printCurrentCoordinates()* prints 5 2 because it moved from (3, 2) → (5, 2).
 9. *secondRobot.moveY(0)* moves the robot 0 units from (5, 2) → (5, 2).
 10. *secondRobot.printLastCoordinates()* prints 5 2 because its last movement was (5, 2) → (5, 2).

YOUR ANSWER

```
1 import java.util.Scanner; ...
2
3 /*
4  * Create the class Robot. Do not use the public access modifier in your class declaration.
5  */
6 public class RobotMoves {
7     private static final Scanner scan = new Scanner(System.in);
8
9     public static void main(String[] args) {
10         int x = scan.nextInt();
11         int y = scan.nextInt();
12         int dx = scan.nextInt();
13         int dy = scan.nextInt();
14
15         Robot firstRobot = new Robot();
16         firstRobot.printCurrentCoordinates();
17
18         Robot secondRobot = new Robot(x, y);
19         secondRobot.printCurrentCoordinates();
20
21         for (int i = 1; i < 3; i++) {
22             secondRobot.moveX(dx);
23             secondRobot.printLastMove();
24
25             secondRobot.printCurrentCoordinates();
26             secondRobot.moveY(dy);
27             secondRobot.printLastCoordinates();
28
29             dx += i * i;
30             dy -= i * i;
31         }
32     }
```


Exercise 39.-

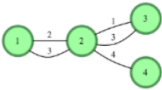
☆ Shared Interests

Given a graph of friends who have various interests, determine which groups of friends have the most interests in common. You will then use a little math to determine a value to return.

The graph will be represented as a series of nodes numbered consecutively from 1 to `friends_nodes`. Friendships have evolved based on interests which will be represented as weights in the graph. Any members who share the same interest are said to be connected by that interest. Once you have determined the node pairs with the maximum number of shared interests, return the maximal product of the node pairs' labels.

For example, you are given a graph with `friends_nodes = 4` and `friends_edges = 5`:

From	To	Weight
1	2	2
1	2	3
2	3	1
2	3	3
2	4	4



If we look at each interest, we have the following connections:

Weight (Interest)	Connections
1	2, 3
2	1, 2
3	1, 2, 3
4	2, 4

In this case, we see the pairs (1, 2) (interests 2 and 3) and (2, 3) (interests 1 and 3) each share two interests which is maximal. We take the products of the node numbers: $1 \times 2 = 2$ and $2 \times 3 = 6$. The maximal value 6 is returned.

Function Description

Complete the function `maxShared` in the editor below. The function must return the maximal integer product of all node pairs sharing the most interests. Indexes align `friends_from`, `friends_to` and `friends_weight`.

maxShared has the following parameter(s):

- `friends_nodes`: integer number of nodes
- `friends_edges`: integer number of edges
- `friends_from[from[0]...from[edges-1]]`: an array of integers
- `friends_to[to[0]...to[edges-1]]`: an array of integers
- `friends_weight[weight[0]...weight[edges-1]]`: an array of integers

Constraints

- $2 \leq \text{friends_nodes} \leq 100$
- $1 \leq \text{friends_edges} \leq \min(200, (\text{friends_nodes} \times \text{friends_nodes} - 1) / 2)$
- $1 \leq \text{friends_weight}[i] \leq 100$
- $1 \leq \text{friends_from}[i], \text{friends_to}[i] \leq \text{friends_nodes}$
- $1 \leq \text{friends_weight}[i] \leq \text{friends_edges}$
- $\text{friends_from}[i] \neq \text{friends_to}[i]$
- Each pair of friends can be connected by zero or more interests.

► Input Format for Custom Testing

▼ Sample Case 0

Sample Input 0

```
4 5
1 2 1
1 2 2
2 3 1
2 3 3
2 4 3
```

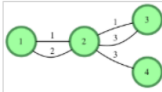
Sample Output 0

```
6
```

Explanation 0

Each pair of `friends_nodes = 4` friends is connected by the following interests:

- Pair (1, 2) shares 2 interests (i.e., interests 1 and 2)
- Pair (1, 3) shares 1 interest (i.e., interest 1)
- Pair (1, 4) shares 0 interests
- Pair (2, 3) shares 2 interests (i.e., interests 1 and 3)
- Pair (2, 4) shares 1 interest (i.e., interest 3)
- Pair (3, 4) shares 1 interest (i.e., interest 3)



The pairs connected by the maximal number of interests are (1, 2) and (2, 3). Their respective products are $1 \times 2 = 2$ and $2 \times 3 = 6$. We then return the largest of these values as our answer, which is 6.