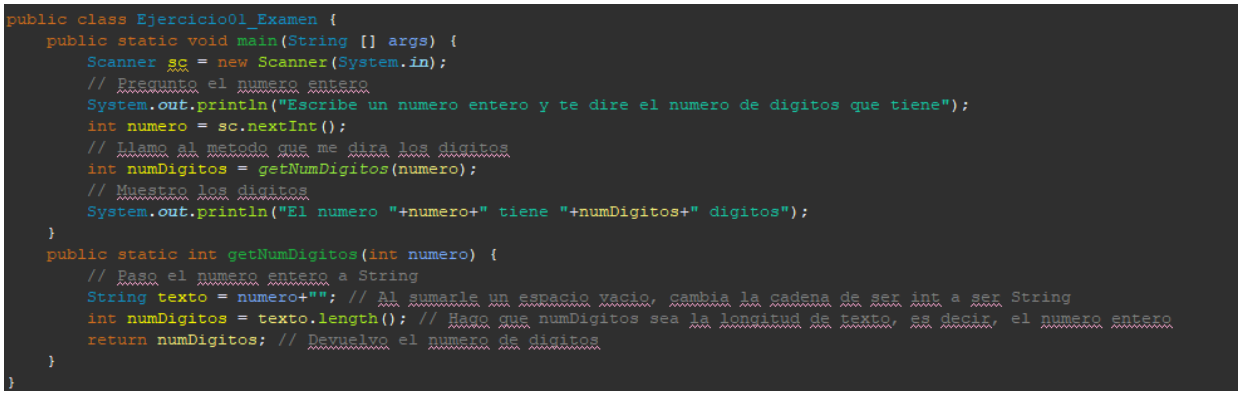
Ejercicio 01



```
public class Ejercicio01_Examen {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        // Pregunto el numero entero
        System.out.println("Escribe un numero entero y te dire el numero de digitos que tiene");
        int numero = sc.nextInt();
        // Llamo al metodo que me dira los digitos
        int numDigitos = getNumDigitos(numero);
        // Muestro los digitos
        System.out.println("El numero "+numero+" tiene "+numDigitos+" digitos");
    }
    public static int getNumDigitos(int numero) {
        // Paso el numero entero a String
        String texto = numero+""; // Al sumarle un espacio vacio, cambia la cadena de ser int a ser String
        int numDigitos = texto.length(); // Hago que numDigitos sea la longitud de texto, es decir, el numero entero
        return numDigitos; // Devuelvo el numero de digitos
    }
}
```

public class Ejercicio01_Examen {

public static void main(String [] args) {

Scanner sc = new Scanner(System.***in***);

// Pregunto el numero entero

System.***out***.println("Escribe un numero entero y te dire el numero de digitos que tiene");

int numero = sc.nextInt();

// Llamo al metodo que me dira los digitos

int numDigitos = *getNumDigitos*(numero);

// Muestro los digitos

System.***out***.println("El numero "+numero+" tiene "+numDigitos+" digitos");

}

public static int getNumDigitos(int numero) {

// Paso el numero entero a String

String texto = numero+""; // Al sumarle un espacio vacio, cambia la cadena de ser int a ser String

int numDigitos = texto.length(); // Hago que numDigitos sea la longitud de texto, es decir, el numero entero

return numDigitos; // Devuelvo el numero de digitos

}

}

Cabecera →

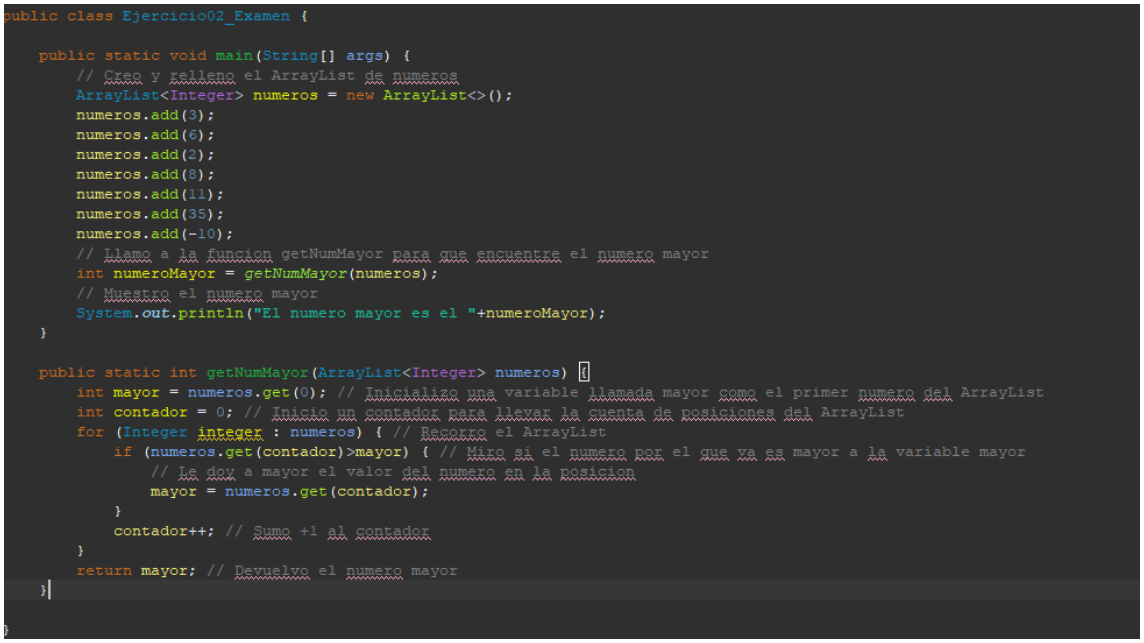
La estructura general de una función es la siguiente:

void / tipoDevuelto nombreMetodo([lista parámetros])

Tipo de método →

En este caso, tenemos un método que devuelve un tipo (int) por lo que no es void. Por otra parte, le hemos pasado como parámetro el numero entero del que queríamos sacar el número de dígitos. Este tipo de funciones requieren de un return a diferencia de los tipos void, los cuales no deben devolver nada.

Ejercicio 02



```
public class Ejercicio02_Examen {

    public static void main(String[] args) {
        // Creo y relleno el ArrayList de numeros
        ArrayList<Integer> numeros = new ArrayList<>();
        numeros.add(3);
        numeros.add(6);
        numeros.add(2);
        numeros.add(8);
        numeros.add(11);
        numeros.add(35);
        numeros.add(-10);
        // Llamo a la funcion getNumMayor para que encuentre el numero mayor
        int numeroMayor = getNumMayor(numeros);
        // Muestro el numero mayor
        System.out.println("El numero mayor es el "+numeroMayor);
    }

    public static int getNumMayor(ArrayList<Integer> numeros) {
        int mayor = numeros.get(0); // Inicializo una variable llamada mayor como el primer numero del ArrayList
        int contador = 0; // Inicio un contador para llevar la cuenta de posiciones del ArrayList
        for (Integer integer : numeros) { // Recorro el ArrayList
            if (numeros.get(contador)>mayor) { // Miro si el numero por el que va es mayor a la variable mayor
                // Le doy a mayor el valor del numero en la posicion
                mayor = numeros.get(contador);
            }
            contador++; // Sumo +1 al contador
        }
        return mayor; // Devuelvo el numero mayor
    }

}
```

public class Ejercicio02_Examen {

public static void main(String[] args) {

// Creo y relleno el ArrayList de numeros

ArrayList<Integer> numeros = new ArrayList<>();

numeros.add(3);

numeros.add(6);

numeros.add(2);

numeros.add(8);

numeros.add(11);

numeros.add(35);

numeros.add(-10);

// Llamo a la funcion getNumMayor para que encuentre el numero mayor

int numeroMayor = *getNumMayor*(numeros);

// Muestro el numero mayor

System.***out***.println("El numero mayor es el "+numeroMayor);

```
    }

    public static int getNumMayor(ArrayList<Integer> numeros) {

        int mayor = numeros.get(0); // Inicializo una variable llamada mayor como el primer numero del ArrayList

        int contador = 0; // Inicio un contador para llevar la cuenta de posiciones del ArrayList

        for (Integer integer : numeros) { // Recorro el ArrayList

            if (numeros.get(contador)>mayor) { // Miro si el numero por el que va es mayor a la variable mayor

                // Le doy a mayor el valor del numero en la posicion

                mayor = numeros.get(contador);

            }

            contador++; // Sumo +1 al contador

        }

        return mayor; // Devuelvo el numero mayor

    }

}
```

Cabecera →

La estructura general de una función es la siguiente:

void / tipoDevuelto nombreMetodo([lista parámetros])

Tipo de método →

En este caso, tenemos un método que devuelve un tipo (int) por lo que no es void. Por otra parte, le hemos pasado como parámetro el ArrayList de números Integer llamado números del que queríamos sacar el número mayor, ya que los ArrayList no pueden almacenar tipos primitivos. Este tipo de funciones requieren de un return a diferencia de los tipos void, los cuales no deben devolver nada.

Ejercicio 03

```
public class Coche {
    private String nombre;
    private ArrayList<Conductor> lista_conductores = new ArrayList<Conductor>();

    public Coche(String nombre) {
        this.nombre = nombre;
    }

    public String getNombre() {
        return this.nombre;
    }

    public int contar_conductores(){

    public void lista_conductores(){

    public void anyadir_conductor(Conductor c) {
        this.lista_conductores.add(c);
    }
}
```

a) Public class Coche se refiere a la clase.

b) La función public Coche (String nombre) se refiere al constructor del objeto, el cual recibirá un nombre para instanciar la plantilla de la clase, creando un objeto con esos atributos.

c) Los atributos son los siguientes:

private String nombre;

Es un atributo privado el cual se refiere al nombre del coche

private ArrayList <Conductor> lita_conductores = new ArrayList<Conductor>();

Es un atributo también privado el cual se refiere a la lista de conductores de objeto Conductor.

En este caso, está definido el ArrayList.

d) Que un atributo sea private se refiere a que no es accesible a otras clases. Si quisieras acceder al atributo private, lo que deberías de hacer es realizar un método getter o setter, con el objetivo de referirte a el con un this.atributo.

e) La palabra this es una palabra reservada que sirve para referirse a los atributos del objeto directamente.

f) La función añadir_conductor recibe como parámetro un conductor (c) lo añade al ArrayList lista_conductores con (lista_conductores.add(c));