**Definiciones**

***-Función:***

* Conjunto de pasos e instrucciones que hacen algo. Opcionalmente puede, recibir variables de entrada; y opcionalmente puede devolver un valor.

***-Objeto:***

* Es la instancia de una clase porque contiene tanto datos (atributos) como comportamiento (métodos), definidos por la clase a partir de la cual fue creado.

***-Instancia:***

* Crea espacio de memoria para albergar los datos de un objeto.

***-Clase:***

* Plantilla donde definimos el objeto real y donde ponemos todas las herramientas para definir las “propiedades” y métodos.

***-Métodos / -constructor:***

* Función que pertenece a una clase, En lugar de escribir el mismo código varias veces, puedes definir un método y luego llamarlo cuando sea necesario (**bloque de código reutilizable)**
* Método especial. **Se trabaja con los datos.** Se utiliza para inicializar un objeto que luego se pueda modificar o no (dependiendo si la clase lo permite).

***-Método estático main:***

* Método especial que actúa como el punto de entrada o iniciador de cualquier aplicación que se ejecuta desde la línea de comandos o un entorno de ejecución.

***-Variables / -propiedades-atributo:***

* Es un almacén donde guardamos los valores en datos en memoria de un tipo al que accedemos desde un nombre y puede cambiar su valor durante la ejecución del programa, (características que estamos definiendo)
* Pertenece a una clase. **Almacena datos.** Es la representación del estado de un objeto de un tipo que accedemos desde la instancia con un nombre y puede cambiar su valor durante la ejecución del programa.

***-Public /-private:***

* Nos permite acceder desde cualquier lugar (de un método, clase o variable) /
* Solo accedemos dentro de la misma clase

***\*This/String/\*toString:***

* Para referirnos a los valores de una instancia (los atributos de una clase) (Recomendable si tenemos un parámetro de método con el mismo nombre que un atributo, podemos usar “this” para aclarar que te refieres al atributo de la clase.
* Clase para trabajar una cadena de texto.
* Método heredado (mecanismo por el cual una clase permite heredar las características de otra clase), y convierte un objeto en cadena de texto.

***Getter/Setter:***

* Nos permite tomar el valor de un atributo
* public String getName(){
* return this.name;
* }
* Nos permite asignar un valor a un atributo
* public void setName(String name){
* this.name = name; //asignacion del valor
* }

**Condición (Sentencia de control):**

***If-else***

* Estructura de control que ejecuta un bloque de código si se cumple una condición.
* Puede tener varios bloques con su condición que se van ejecutando en cascada si no se cumple la condición del bloque anterior.
* Si no se cumple ninguna condición puede ejecutarse un bloque llamado `else`.

*Estructura:*

if (condición) {

    // Si SI cumple la condición

}

// No es obligatorio

else if (condición2) {

    // Si SI cumple la condición2

}

// No es obligatorio

else {

    // Si NO cumple ninguna condición

}

***Switch***

* Estructura de control que ejecuta un bloque de código si se cumple un valor.
* Puede tener varios casos con sus bloques.
* Si no se cumple ninguna condición puede ejecutarse un bloque llamado `default`.
* Después de cada bloque se usa `break` para indicar que termina el bloque (se puede no poner y ejecutaría el bloque siguiente).

*Estructura*

switch(valor) {

    case valor1:

        // Si SI valor == valor1

        break;

    // No es obligatorio

    case valor2:

        // Si SI valor == valor2

        break;

    // No es obligatorio pero recomendable

    default:

        // Si ningún case es igual al valor

}

***Ternary***

* Su estructura de control que se asigna a una variable un valor que, dependiendo de una condición, un valor para el caso verdadero y otro para el caso falso. Se puede anidar.

*Estructura:*

* Estructura de control que asigna a una variable, dependiendo de una condición, un valor para el caso verdadero u otro valor para el caso falso.

variable variableName = (condición) ? valorSiVerdadero : valorSiFalso;

**Bucles**

***For y foreach***

* Se repite un número concreto de veces (se puede calcular desde el principio)
* Repite un bloque de código

***Estructura:*** *For*

for(i = valorInicial; condición; variación) {

    ...

}

***Estructura:*** *Foreach*

for(objetoAIterar : listaIterable) {

    ...

}

***While***

* Se repite un número indeterminado de veces (se suele calcular dentro de si mismo)
* Repite un bloque de código
* Se comprueba la condición antes de ejecutar el bloque

*Estructura*

while (condición) {

    ...

}

***do while***

* Se repite un número indeterminado de veces (se suele calcular dentro de si mismo)
* Repite un bloque de código
* Se comprueba la condición después de ejecutar el bloque

### **Estructura:**

do {

    ...

} while (condición);

***labels***

- break

- continue

- return

**“Ejercicios”**

* Dado una clase “classImportant” píntame el método “myMethod” con instrucciones … que recibe un String “info” y no devuelve nada.

Public class ClassImportant {

    Public void myMethod(String info) {

        …

    }

}

Dado una clase “TrafficLight” píntame el constructor que inicializa el estado de la “luz” (´verde´, ´naranja´, o ´rojo´) y créame su getter.

public class TrafficLight{

    //Su valor solo puede ser: ´verde´, ´naranja´ o ´rojo´

    private String lightState;

    public TrafficLight(String lighState){

        //Comprobar que el valor recibido cumpla lsa restricciones

        this.lightState = lightState;

    }

    public String getLighState(){

        return this.lightState;

    }

}