





Reto 2 – Alquiler de Bicicletas

Objetivo:

El objetivo de este reto es que el estudiante reconozca y aplique los elementos básicos del paradigma de la programación orientada a objetos en un escenario abstraído de la cotidianidad.

Contexto:

La compañía de renta de bicicletas ha logrado atraer a una gran cantidad de clientes interesados en transportarse por la ciudad haciendo uso de sus bicicletas, lo que le ha permitido instalar nuevas estaciones para que las personas puedan encontrar las bicicletas más fácilmente.

Es normal que ahora que las bicicletas pasan más tiempo en servicio también requieran mantenimiento más frecuentemente, y la empresa en los últimos días ha estado recibiendo varios informes de que hay bicicletas que no están en condiciones aceptables. Sin embargo, por la naturaleza del servicio, se ha dificultado la labor de encontrar estas bicicletas pues suelen estas cambiando de estación constantemente.

Para afrontar la situación se ha propuesto modificar el sistema actual de la siguiente manera:

Estación	Bicicleta		
- ld: int - Ubicacion: String - Bicicletas: Bicicleta[]	- ld: String - EnServicio: boolean		
+ BicicletasEnServicio(): Bicicleta[] + BicicletasDisponibles(): Bicicleta[] + EncontrarBicicleta(String id): boolean + ConsultarBicicleta(String id): Bicicleta	+ getEnServicio(): boolean + getId(): String		







Reto:

- 1. Refactorice la clase **Bicicleta** como se muestra en el diagrama, implementa un constructor y sus respectivos **getters**.
- 2. Implemente la clase **Estacion** como se muestra en el diagrama, implementa un constructor y las funciones descritas a continuación.
- 3. Refactorice las funciones BicicletasEnServicio y BicicletasDisponibles implementadas en el reto anterior. Note que ahora no reciben argumentos, ahora deben utilizar el miembro dato Bicicletas de la instancia. Ambas funciones aún retornan un arreglo de tipo Bicicleta con las bicicletas que están en servicio o disponibles respectivamente.
 - Cabe destacar que el arreglo que retornan estas dos funciones debe tener un tamaño igual a la cantidad de bicicletas que cumplen la condición encontradas. Es decir, si en la función **BicicletasDisponibles** se encuentran 6 bicicletas que no están en servicio, el arreglo retornado debe ser de tamaño 6, ni más ni menos.
- 4. Implemente la función **EncontrarBicicleta**, que toma como argumento un string que representa el id de la bicicleta que se está buscando. La función debe buscar en el arreglo de **Bicicletas** de la instancia si alguna de las bicicletas tiene el mismo Id que se busca. En caso de encontrarlo retorna verdadero, si no lo encuentra retorna falso.
- 5. Implemente la función **ConsultarBicicleta**, que toma como argumento un string que representa el id de la bicicleta que se está buscando. La función debe buscar en el arreglo de **Bicicletas** de la instancia si alguna de las bicicletas tiene el mismo Id que se busca. En caso de encontrarla retorna esa instancia de bicicleta, si no se encuentra retorna **null**.







Casos de Prueba:

Para validar el correcto funcionamiento del programa considere los siguientes escenarios:

Datos de Entrada:

Para una estación que cuenta con arreglo de bicicletas como el siguiente:

0	1	2	3	4	5
ld: Bicicleta1	ld: Bycicle2	ld: Bicyclette3	ld: Fahrrad4	ld: Velosipyed5	ld: Biciklo6
EnServicio: false	EnServicio: false	EnServicio: true	EnServicio: false	EnServicio: true	EnServicio: false

Caso de Prueba	Salida Esperada					
1. BicicletasEnServicio()		0		ld: Velosipyed5 EnServicio: true		
	ld: Bicyclette3 EnServicio: true					
2. BicicletasDisponibles()	0		1	2		3
	ld: Bicicleta1 EnServicio: false		ycicle2 ervicio: false	ld: Fahrrad4 EnServicio: false		iciklo6 ervicio: false
3.EncontrarBicicleta("Velosipyed5")	True					
4. EncontrarBicicleta("Esta Bicicleta No Existe")	False					
5. ConsultarBicicleta("Bicyclette3")		Bicicleta				
			ld: Bicyclette3 EnServicio: true			
6. ConsultarBicicleta("Esta Bicicleta No Existe")	null					







ENTREGA:

- 1. Los archivos que suba a la plataforma para su calificación deben llamarse **exactamente** *Bicicleta.java* y *Estacion.java*, de lo contrario no se calificará.
- 2. Los nombres de las clases, miembros dato y funciones deben llamarse exactamente como se muestran en los diagramas mostrados al comienzo del reto, las firmas de sus clases deben ser cómo se muestra en las siguientes imágenes:

```
public class Estacion {
    private int Id;
    private String Ubicacion;
    private Bicicleta[] Bicicletas;

    public Estacion(int id, String ubicacion, Bicicleta[] bicicletas) {
        //Implementación
    }

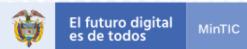
    public Bicicleta[] BicicletasEnServicio(){
        //Implementación
    }

    public Bicicleta[] BicicletasDisponibles(){
        //Implementación
    }

    public boolean EncontrarBicicleta(String id){
        //Implementación
    }

    public Bicicleta ConsultarBicicleta(String id){
        //Implementación
    }

}
```







```
public class Bicicleta {
    private String Id;
    private boolean EnServicio;

public Bicicleta(String _Id, boolean _EnServicio){
    //Implementación
    }

public String getId() {
    //Implementación
    }

public boolean getEnServicio(){
    //Implementación
    }
}
```