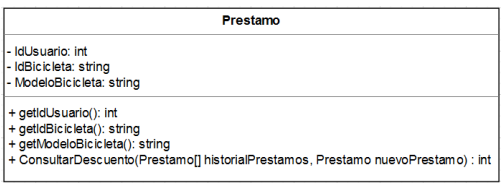
Reto – Préstamos de Bicicletas

**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**:

Una empresa de préstamos de bicicletas quiere atraer nuevos usuarios y afianzar a los clientes regulares con los que cuenta, por lo que ha decidido implementar un nuevo modelo de negocio en el que recompensa a los usuarios más recurrentes con descuentos. En el modelo de negocio actual de la empresa un Prestamo está definido como se muestra en el siguiente diagrama:



La empresa establece una lógica de negocio para aplicar descuentos a los usuarios en base a la cantidad de préstamos que ha hecho el usuario con anterioridad, por lo que ha estado guardando registros de cada vez que un usuario hace un préstamo como se muestran en la siguiente tabla:

Tabla

Descripción generada automáticamente

**Reto 1:**

Debe implementar la clase como se muestra en el diagrama UML de **Prestamo**. Las funciones **getIdUsuario**, **getIdBicicleta** y **getModeloBicicleta** retornan los miembros dato correspondientes.

Para la implementación de la función **ConsultarDescuento**, note que toma dos argumentos, un arreglo de **Prestamo** que representa un historial de préstamos y un objeto tipo **Prestamo** que representa el préstamo que se está haciendo, la función debe retornar un número entero con el descuento que se aplica para el nuevo préstamo, las instrucciones para calcular un descuento son las siguientes:

• Si el usuario del nuevo préstamo cuenta con menos de 3 préstamos no aplica para ningún descuento, y la función retorna 0.

• Si el usuario usuario del nuevo préstamo cuenta con entre 3 y 5 préstamos recibe un descuento del 4%.

• Si el usuario usuario del nuevo préstamo cuenta 6 o más préstamos recibe un descuento del 6%.

Sumado a esto, si el usuario cuenta con 6 o más prestamos, también se aplica un descuento extra dependiendo del modelo de la bicicleta:

Tabla

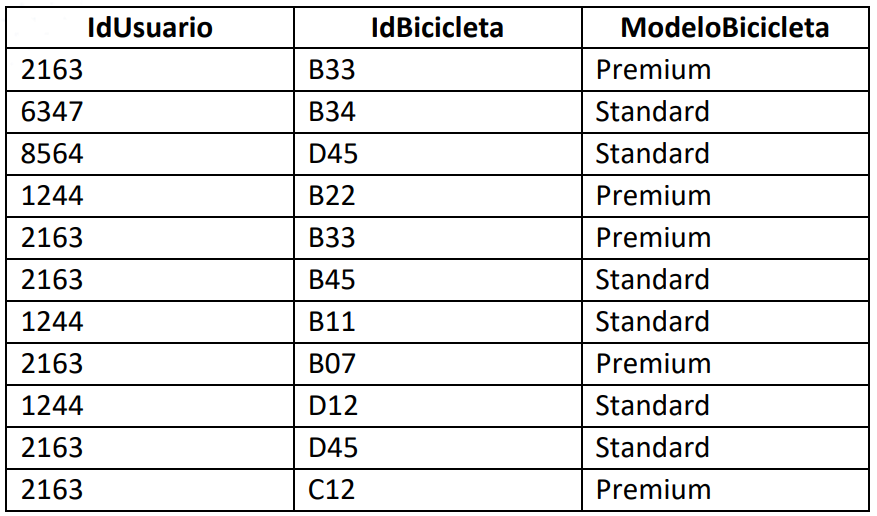
Descripción generada automáticamente

El valor que retorna la función es entonces la suma de ambos descuentos, por ejemplo, recurriendo a la tabla mostrada con anterioridad, si el usuario **3526** hiciera el préstamo de una bicicleta Premium, recibiría un descuento del 4% por tener entre 3 y 5 préstamos, pero no aplicaría al descuento por modelo de bicicleta.

**Casos de Prueba**:

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

**Historial de prueba**:



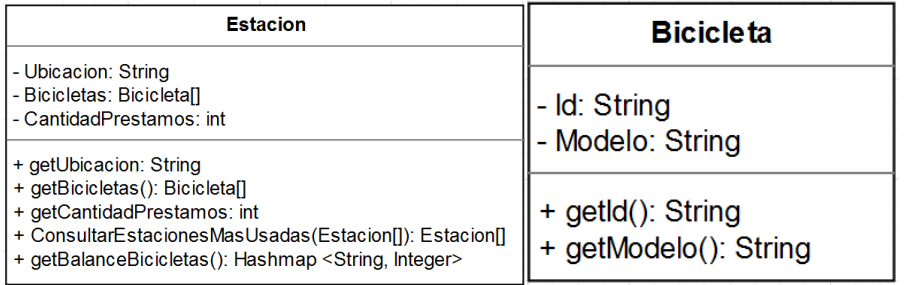
Calendario

Descripción generada automáticamente con confianza baja

**Segunda Parte del proyecto:**

**Contexto**:

La empresa de préstamos de bicicletas ha logrado un gran éxito con el nuevo modelo de negocios, debido a la gran afluencia de nuevos clientes las estaciones de bicicletas actuales no dan abasto y la empresa necesita saber cuáles estaciones son las más usadas para incluirlas en un proyecto de expansión. Además, les gustaría estandarizar el servicio a un solo tipo de bicicleta, por lo que también están le solicitando a usted que implemente un sistema que les permita saber cuántas bicicletas de cada tipo (Premium o Standard) hay en cada estación. Para esto le proponen que implemente un sistema como se muestra a continuación:

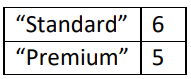


**Reto: 2**.

1. Implemente las clases **Bicicleta** y **Estacion** como se muestra en el diagrama, con sus respectivos constructores y **getters**.

2. Para la implementación del método estático **ConsultarEstacionesMasUsadas**, note que tiene un parámetro, un arreglo de **Estacion** que representa una lista de todas las estaciones con las que cuenta la empresa. La función debe retornar un arreglo con las tres estaciones que tienen la mayor cantidad de préstamos, ordenadas de mayor a menor. En caso de encontrarse dos estaciones con la misma cantidad de préstamos, deben aparecer en el mismo orden en que se encuentran en la lista original.

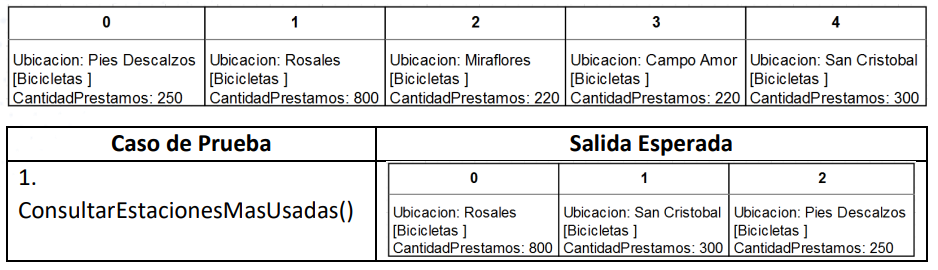
3. Para la implementación de la función **getBalanceBicicletas** debe retornar un diccionario (**Hashmap**) con la cantidad de bicicletas que hay de cada tipo en la estación. Los pares llave valor son un **String** que es el modelo de la bicicleta y un **Integer** con la cantidad de bicicletas de ese tipo. Por ejemplo:



Casos de Prueba:

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

• Para una lista de estaciones como la siguiente:



• Considere la siguiente lista de estaciones:

Tabla

Descripción generada automáticamente

• Para una estación que tiene un arreglo de bicicletas como el siguiente:

