Enunciado

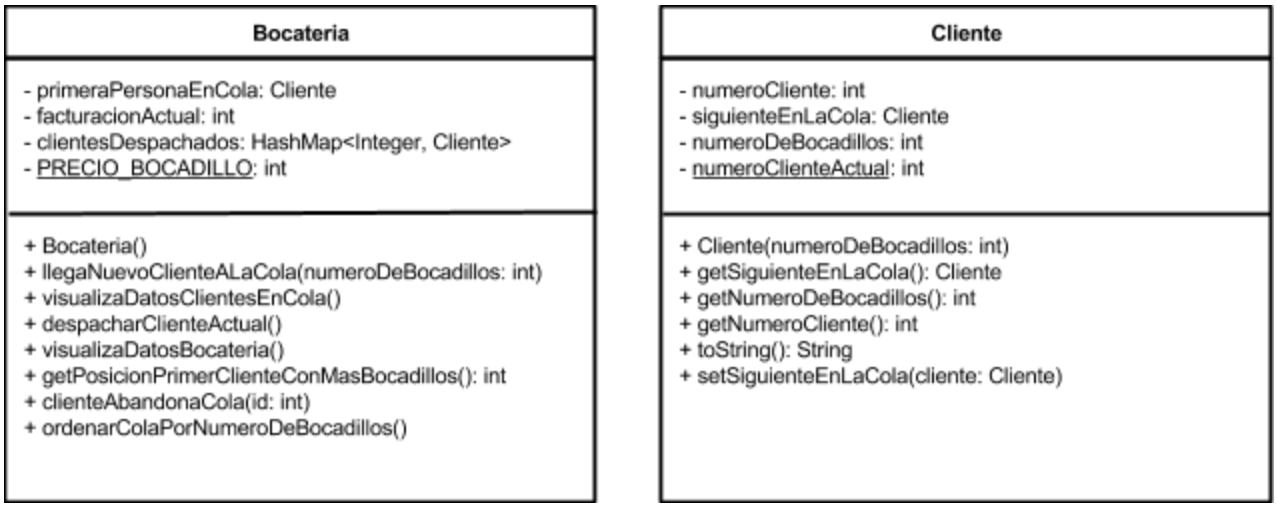
Se desea realizar una aplicación destinada a gestionar una bocatería. En concreto se desea llevar el control de la facturación y de la cola de clientes en la caja.

La bocatería abre a las 9:00 de la mañana y arranca nuestra aplicación. En ese momento no hay nadie en la tienda y por tanto no se ha hecho aun ninguna venta.

Conforme van llegando los clientes estos eligen sus bocadillos de los estantes de la tienda y se ponen a la cola en la caja. Todos los bocadillos tienen el mismo precio: 5 euros.

Nuestra aplicación debe asignar al primer cliente un número de cliente 1, al segundo el número 2, etc.

Se plantean los siguientes diagramas de clases a utilizar:



Pasos a completar

Crea un nuevo proyecto en BlueJ y luego crea un repositorio git en la carpeta correspondiente a dicho proyecto.

Commit 1

20 puntos Codifica el código necesario para:

1. Poder crear objetos de la clase `Bocateria` a través de la interfaz de BlueJ.
2. Implementar el método `llegaNuevoClienteALaCola`. Este método debe crear un nuevo objeto `Cliente` y ponerlo a la cola de clientes que están esperando a ser atendidos. Es importante destacar que los objetos `Cliente` los crea el método, no el usuario a través de la interfaz de BlueJ.

Commit 2

15 puntos Codifica el código necesario para poder visualizar por pantalla los datos de los clientes que están a la cola actualmente por medio del método `visualizaDatosClientesEnCola`.

Un ejemplo de visualización por pantalla de estos datos es:

```
Cliente 1: 3 bocadillo/s (15 euros)
Cliente 2: 10 bocadillo/s (50 euros)
Cliente 3: 2 bocadillo/s (10 euros)
Cliente 4: 5 bocadillo/s (25 euros)
```

Commit 3

15 puntos Codifica el código necesario para despachar al cliente que está el primero de la cola a través del método `despacharClienteActual`. La facturación de la tienda debe verse incrementada, la cola de la caja debe avanzar y el cliente debe pasar al registro de clientes despachados.

En el HashMap utilizado para almacenar a los clientes que ya han sido despachados la clave es el número de cliente y el valor el objeto cliente.

Commit 4

15 puntos Codifica el código necesario para poder ver los datos de la bocateria en el instante actual, lo que incluye la facturacion hasta este momento, la informacion de los clientes de la cola (numero de cliente, número de bocadillos y coste de la compra) y la informacion de los clientes despachados hasta el momento (número de cliente y número de bocadillos).

Un ejemplo de visualización por pantalla de estos datos una vez despachado el cliente numero 1 sería:

```
Facturacion actual: 15 euros
Estado de la cola:
Cliente 2: 10 bocadillo/s (50 euros)
Cliente 3: 2 bocadillo/s (10 euros)
Cliente 4: 5 bocadillo/s (25 euros)
Clientes despachados:
Cliente 1: 3 bocadillos
```

Commit 5

15 puntos Codifica el código necesario para implementar el método `getPosicionPrimerClienteConMasBocadillos` que devuelve la posición en la cola de dicho cliente. Si la cola está vacía el método devuelve -1. La primera posición se indica con el número 1, la segunda con el 2, etc. Si varios clientes están empatados en cuanto a cantidad máxima de bocadillos, devuelve el que esté antes en la cola.

Commit 6

10 puntos Codifica el código necesario para implementar el método `clienteAbandonaCola` en el que se indica como parámetro el numero de cliente que ha abandonado la cola sin llevar a efecto la compra de los bocadillos.

Commit 7

10 puntos Codifica el código necesario para implementar el método `ordenarColaPorNumeroDeBocadillos`. Si varios clientes están empatados en el número de bocadillos no es relevante como los ordene el método.

Observaciones

Es **obligatorio** para que se corrija el examen que el código compile sin errores y que se utilice git de forma correcta para el control de versiones.

No se permite utilizar más de un `return` en los métodos.

Se valora negativamente que el código no esté comentado, la redundancia de código y la indentación incorrecta.

No se pueden implementar en las clases más atributos de los reflejados en el diagrama UML.

Importante

No se permite el uso de ninguna forma de comunicación (verbal, escrita, por Internet) entre alumnos o con cualquier otra persona.

Tampoco se permite ningún mecanismo que permita transmitir el código fuente al resto de alumnos, incluido el simple acceso a repositorios de otros alumnos.

No se permite subir a GitHub el repositorio hasta el momento en que indique el profesor.

En caso de detectar alguna de estas prácticas, el alumno suspende directamente la convocatoria actual.

Entrega

Al finalizar el período del examen y **nunca antes** debe subirse el repositorio a GitHub e indicar a través de un mensaje directo en Slack la URL del último commit.