

Trabajo Práctico Integrador: Sistema para un comercio “Almacén Granate”

Defensas: En clase por grupos se exponen los avances del proyecto.

- Comprar digitalmente un carrito
- Gestionar turnos de retiro de la compra
- Gestionar entrega a domicilio
- Gestionar ofertas

```

classDiagram
    class Turno {
        -dia: LocalDate
        -hora: LocalTime
        -ocupado: boolean
    }
    class Comercio {
        -nombreComercio: String
        -cuit: long
        -costoFijo: double
        -costoPorKm: double
        -diaDescuento: int
        -porcentajeDescuentoDia: int
        -porcentajeDescuentoEfectivo: int
        -lstDiaRetiro: DiaRetiro
        -lstArticulo: Articulo
        -lstCarrito: Carrito
    }
    class Cliente {
        -apellido: String
        -nombres: String
        -dni: long
        -sexo: char
    }
    class Contacto {
        -email: String
        -celular: String
        -ubicacion: Ubicacion
    }
    class Actor {
        #id: int
        #contacto: Contacto
    }
    class Ubicacion {
        -latitud: double
        -longitud: double
    }
    class Entrega {
        #id: int
        #fecha: LocalDate
        #efectivo: boolean
    }
    class Envio {
        -horaHasta: LocalTime
        -horaDesde: LocalTime
        -costo: double
        -ubicacion: Ubicacion
    }
    class RetiroLocal {
        -horaEntrega: LocalTime
    }
    class Carrito {
        -id: int
        -fecha: LocalDate
        -hora: LocalTime
        -cerrado: boolean
        -descuento: double
        -cliente: Cliente
        -lstItemCarrito: ItemCarrito
        -entrega: Entrega
    }
    class DiaRetiro {
        -id: int
        -diaSemana: int
        -horaDesde: LocalTime
        -horaHasta: LocalTime
        -intervalo: int
    }
    class ItemCarrito {
        -articulo: Articulo
        -cantidad: int
    }
    class Articulo {
        -id: int
        -nombre: String
        -codBarras: String
        -precio: double
    }

    Turno ..|> Comercio
    Comercio "1" o-- "1" DiaRetiro
    Comercio "1" o-- "0..*" Carrito
    Comercio "1" o-- "1" Cliente
    Contacto "1" -- "1" Actor
    Actor "1" -- "1" Carrito
    Carrito "1" -- "1" Entrega
    Carrito "1" -- "1..*" ItemCarrito
    ItemCarrito "0..*" -- "1" Articulo
    Entrega <|-- Envio
    Entrega <|-- RetiroLocal
    
```

UML diagram illustrating the structure of a commerce system, showing classes and their relationships:

- Turno** (Class):
 - Attributes: `-dia: LocalDate`, `-hora: LocalTime`, `-ocupado: boolean`
- Comercio** (Class):
 - Attributes: `-nombreComercio: String`, `-cuit: long`, `-costoFijo: double`, `-costoPorKm: double`, `-diaDescuento: int`, `-porcentajeDescuentoDia: int`, `-porcentajeDescuentoEfectivo: int`, `-lstDiaRetiro: DiaRetiro`, `-lstArticulo: Articulo`, `-lstCarrito: Carrito`
- Cliente** (Class):
 - Attributes: `-apellido: String`, `-nombres: String`, `-dni: long`, `-sexo: char`
- Contacto** (Class):
 - Attributes: `-email: String`, `-celular: String`, `-ubicacion: Ubicacion`
- Actor** (Class):
 - Attributes: `#id: int`, `#contacto: Contacto`
- Ubicacion** (Class):
 - Attributes: `-latitud: double`, `-longitud: double`
- Entrega** (Class):
 - Attributes: `#id: int`, `#fecha: LocalDate`, `#efectivo: boolean`
- Envio** (Class):
 - Attributes: `-horaHasta: LocalTime`, `-horaDesde: LocalTime`, `-costo: double`, `-ubicacion: Ubicacion`
- RetiroLocal** (Class):
 - Attributes: `-horaEntrega: LocalTime`
- Carrito** (Class):
 - Attributes: `-id: int`, `-fecha: LocalDate`, `-hora: LocalTime`, `-cerrado: boolean`, `-descuento: double`, `-cliente: Cliente`, `-lstItemCarrito: ItemCarrito`, `-entrega: Entrega`
- DiaRetiro** (Class):
 - Attributes: `-id: int`, `-diaSemana: int`, `-horaDesde: LocalTime`, `-horaHasta: LocalTime`, `-intervalo: int`
- ItemCarrito** (Class):
 - Attributes: `-articulo: Articulo`, `-cantidad: int`
- Articulo** (Class):
 - Attributes: `-id: int`, `-nombre: String`, `-codBarras: String`, `-precio: double`

Relationships:

- Turno** is a specialization of **Comercio** (indicated by a dashed line with a hollow triangle).
- Comercio** has a 1-to-1 relationship with **DiaRetiro**.
- Comercio** has a 1-to-0..* relationship with **Carrito**.
- Comercio** has a 1-to-1 relationship with **Cliente**.
- Contacto** has a 1-to-1 relationship with **Actor**.
- Actor** has a 1-to-1 relationship with **Carrito**.
- Carrito** has a 1-to-1 relationship with **Entrega**.
- Carrito** has a 1-to-1..* relationship with **ItemCarrito**.
- ItemCarrito** has a 0..*-to-1 relationship with **Articulo**.
- Entrega** is a generalization of **Envio** and **RetiroLocal** (indicated by hollow triangle inheritance arrows).

Note: NO está marcada la relación entre Cliente y Carrito (uno a muchos) para no cruzar relaciones en el diagrama

El sistema debe implementar la siguiente funcionalidad:

Validar identificador:

DNI: el ente nacional para validación de DNI es el ReNaPer para este TP de laboratorio solo validar que el ingreso sea numérico.

CUIT: según el Algoritmo: verifica un CUIT de la Guía 6 (Métodos Estáticos y Excepciones)

Envío a domicilio

El costo de envío es el resultado de un costo fijo en función de la distancia del local al domicilio.

Para calcular la distancia se propone el siguiente método:

Cálculo de distancia:

```
public double distanciaCoord(double lat1, double lng1, double lat2, double lng2) {  
    double radioTierra = 6371; //en kilómetros  
    double dLat = Math.toRadians(lat2 - lat1);  
    double dLng = Math.toRadians(lng2 - lng1);  
    double sindLat = Math.sin(dLat / 2);  
    double sindLng = Math.sin(dLng / 2);  
    double va1 = Math.pow(sindLat, 2) + Math.pow(sindLng, 2) * Math.cos(Math.toRadians(lat1)) *  
        Math.cos(Math.toRadians(lat2));  
    double va2 = 2 * Math.atan2(Math.sqrt(va1), Math.sqrt(1 - va1));  
    return radioTierra * va2;  
}
```

Retirar en el negocio

Para retirar en el negocio el comercio determina para cada día de la semana de entrega, la franja horaria y el intervalo de tiempo.

Descuento

- 1) El comercio ofrece un día a la semana con la compra de 2 artículos iguales aplica un porcentaje de descuento al precio de la 2° unidad. Por ejemplo si compra 7 artículos iguales tendrá un descuento = $3 * \text{precioArticulo} * \text{porcentajeDeDescuento} / 100$
- 2) El comercio sobre la compra aplica un porcentaje de descuento sobre el costo del carrito algunos días o si el cliente realiza el pago en efectivo o débito.

El CU que determina el descuento sobre el carrito aplicará el mayor de los dos de los descuentos sobre la compra.

Agregar al carrito

Cuando el cliente agrega al carrito un artículo y cantidad se debe verificar si existe un objeto ItemCarrito para ese artículo, en caso que existe se incrementa la cantidad de lo contrario se crea y se agrega a la lista.

Sacar del carrito

Cuando el cliente saca del carrito una cantidad de un artículo, si la cantidad es menor la resta y si es igual elimina el objeto ItemCarrito.

Algunos de los CU que debe implementar el sistema:

- 1) # validarIdentificadorUnico(long identificador) : boolean //valida DNI o CUIT según la sub-clase

- 2) + traerHoraRetiro (LocalDate fecha): LocalTime
- 3) + generarTurnosLibres (LocalDate fecha) : Turno //retorna una lista de objetos Turno libres
- 4) + traerTurnosOcupados(LocalDate fecha):Turno //retorna una lista de objetos Turno dados
- 5) + generarAgenda (LocalDate fecha) : Turno //retorna una lista de objetos Turno indicando si está ocupado o libre.
- 6) + agregarDiaRetiro(int diaSemana, LocalTime horaDesde, LocalTime horaHasta, int intervalo):boolean
- 7) + validarCodBarras(String codBarras):boolean //Se valida por el sistema EAN -13 http://www.alimentosargentinos.gob.ar/contenido/revista/pdfs/07/07_03_codigo.htm
- 8) + agregar(Articulo articulo, int cantidad):boolean
- 9) + calcularSubTotalItem():double
- 10) + calcularTotalCarrito() : doble
- 11) + calcularDescuentoDia (int diaDescuento, double porcentajeDescuentoDia):double
- 12) + calcularDescuentoEfectivo (double porcentajeDescuentoEfectivo): double
- 13) + calcularDescuentoCarrito (int diaDescuento, double porcentajeDescuento, double porcentajeDescuentoEfectivo) : //determina cual es el descuento mayor
- 14) # setDescuento(double descuento): //es optativo definirlo protected si solo se setea por las reglas de negocio de descuento
- 15) + totalAPagarCarrito() : double
- 16) + setCosto(Ubicacion ubicacion, double costoFijo, double costoPorKm):
- 17) + traerUbicacion() : Ubicacion

Capa Test

La capa de testeo debe crear los objetos necesarios en una clase ejecutable para los posibles escenarios según el caso de uso a testear.