**Programación ll – Trabajo Práctico Integrador**

**2do. Cuatrimestre 2023**

**Primera Parte**

**Integrantes Grupo 6:**

**Nombre y Apellido**: Joaquin Hernández Durand

**DNI**: 44131754

**Mail**: [joacohernandez157@gmail.com](mailto:joacohernandez157@gmail.com)

**COM:** 03

**Nombre** **y** **Apellido**: Joaquin Flores

**DNI**: 43663411

**Mail**: [joaco5flores@gmail.com](mailto:joaco5flores@gmail.com)

**COM:** 03

**Fecha de entrega:** 16/10/2023

**TAD EmpresaAmazing**

**Datos:**

* HashMap<Integer, Pedido> pedidos;
* HashMap <String, Transporte>transportes;
* Double facturacionTotalPedidosCerrados;
* String cuit;

**Operaciones:**

* crear(String cuit): void;
* registrarAutomovil(String patente, int volMax, int valorViaje, int maxPaq): void;
* registrarUtilitario(String patente, int volMax, int valorViaje, int valorExtra): void;
* registrarCamion(String patente, int volMax, int valorViaje, int adicXPaq): void;
* registrarPedido(String cliente, String direccion, int dni) : int;
* agregarPaquete(int codPedido, int volumen, int precio, int costoEnvio): int;
* agregarPaquete(int codPedido, int volumen, int precio, int porcentaje, int adicional): int;
* quitarPaquete(int codPaquete): boolean;
* cerrarPedido(int codPedido): double;
* cargarTransporte(String patente): String;
* costoEntrega(String patente): double;
* facturacionTotalPedidosCerrados(): double;
* Map<Integer, String> pedidosNoEntregados(): Map<Integer, String>;
* hayTransportesIdenticos(): boolean;
* toString(): String;

**IREP:**

* registrarAutomovil, registrarUtilitario y registrar Camión, toman una patente, si esa patente ya está en el sistema genera una excepción.
* agregarPaquete genera una excepción si el pedido no esta registrado o si el pedido ya está finalizado.
* quitarPaquete genera una excepción si no encuentra el paquete en el pedido.
* cerrarPedido genera una excepción si no existe el pedido o si el pedido ya fue cerrado anteriormente.
* cargarTransporte genera una excepción si no encuentra el transporte con la patente dada o si el transporte está lleno.
* costoEntrega genera una excepción si no existe el transporte con la patente dada o si el transporte está vacío.

**TAD PaqueteEspecial**

**Datos:**

* int idUnico;
* int volumen;
* double precio;
* boolean entregado;
* String direccion;
* double porcentajeAdicional;
* int valorAdicional;

**Operaciones:**

* crear(int volumen, double precio, String direccion): void;
* fueEntregado(): boolean;
* paqueteEntregado(): boolean;
* adicionalAPagar (): int;
* costoFinal(): double;
* toString(): String;

**IREP:**

* idUnico deber ser un entero no repetido.
* Volumen debe ser un numero entero mayor que 0.
* porcentajeAdicional debe ser un double mayor que 0.
* valorAdicional debe ser un entero mayor que 0.

**TAD PaqueteOrdinario**

**Datos:**

* int idUnico;
* int volumen;
* double precio;
* String direccion;
* boolean entregado;
* double costoEnvio;

**Operaciones:**

* crear(int volumen, double precio, String direccion, double costoEnvio): void;
* paqueteEntregado(): void;
* fueEntregado(): boolean;
* costoFinal(): double;
* toString(): String;

**IREP:**

* idUnico deber ser un entero no repetido.
* volumen debe ser un entero positivo.
* costoEnvio debe ser un double mayor o igual que 0.

**TAD TransporteComun**

**Datos:**

* String patente;
* Int volumenMaximo;
* Int volumenActual;
* double valorQueCobra;
* Int limitePaquetes;
* List<Paquete> paquetesCargados;

**Operaciones:**

* crear(String patente, int volumenMaximo, int limitePaquetes, double valorQueCobra): void;
* aumentarVolumen(int volumenPaquete): void;
* costoEntrega(): double;
* seCumplenCondiciones(Paquete p): boolean;
* cargarPaquete(Paquete paquete): void;
* transporteVacio(): boolean;
* List<Paquete> listadoPaquetesAEntregar: List<Paquete>;
* superaLimite(): boolean;
* transporteLLeno(): boolean;
* transporteEstaLLeno(): void;
* toString(): String;

**IREP:**

* Patente debe ser una combinación de enteros y números no nulos.
* volumenMaximo, y limitesPaquetes deben ser enteros positivos.
* valorqueCobra debe ser un double mayor que 0.
* volumenActual no puede superar volumenMaximo.
* aumentarVolumen genera excepción si recibe un número menor que 0.
* transporteEstaLLeno genera excepción si el transporte esta lleno.

**TAD TransporteUtilitario**

**Datos:**

* String patente;
* Int volumenMaximo;
* Int volumenActual;
* double valorQueCobra;
* double valorExtra;
* List<Paquete> paquetesCargados;

**Operaciones:**

* crear(String patente, int volumenMaximo, double valorExtra, double valorQueCobra): void;
* aumentarVolumen(int volumenPaquete): void;
* costoEntrega(): double;
* seCumplenCondiciones(Paquete p): boolean;
* cargarPaquete(Paquete paquete): void;
* transporteVacio(): boolean;
* List<Paquete> listadoPaquetesAEntregar: List<Paquete>;
* superaLimite(): boolean;
* transporteEstaLLeno(): void;
* calcularCostoViaje(): void;
* toString(): String;

**IREP:**

* Patente debe ser una combinación de enteros y números no nulos.
* volumenMaximo debe ser un entero positivo.
* valorqueCobra y valorExtra deben ser double mayor que 0.
* volumenActual no puede superar volumenMaximo.
* aumentarVolumen genera excepción si recibe un número menor que 0.
* transporteEstaLLeno genera excepción si el transporte está lleno.

**TAD TransporteCamion**

**Datos:**

* String patente;
* Int volumenMaximo;
* Int volumenActual;
* double valorQueCobra;
* double valorAdicional;
* List<Paquete> paquetesCargados;

**Operaciones:**

* crear(String patente, int volumenMaximo, double valorAdicional, double valorQueCobra): void;
* aumentarVolumen(int volumenPaquete): void;
* costoEntrega(): double;
* seCumplenCondiciones(Paquete p): boolean;
* cargarPaquete(Paquete paquete): void;
* transporteVacio(): boolean;
* List<Paquete> listadoPaquetesAEntregar: List<Paquete>;
* superaLimite(): boolean;
* transporteEstaLLeno(): void;
* calcularCostoViaje(): void;
* toString(): String;

**IREP:**

* Patente debe ser una combinación de enteros y números no nulos.
* volumenMaximo debe ser un entero positivo.
* valorqueCobra y valorAdicional deben ser double mayor que 0.
* volumenActual no puede superar volumenMaximo.
* aumentarVolumen genera excepción si recibe un número menor que 0.
* transporteEstaLLeno genera excepción si el transporte está lleno.

**TAD Pedido**

**Datos:**

* int nroPedido;
* String dirección;
* String nombreDeCliente;
* Int dni;
* boolean estCerrado;
* int contadorPedidos;
* HashMap <Integer,Paquete> carrito;

**Operaciones:**

* crear(int nroPedido, int dni, String dirección, String nombreDeCliente, boolean estaCerrado): void;
* agregarProductoCarrito(int volumen, double precio, int costoEnvio): void;
* agregarProductoOrdinario(Integer id, PaqueteOrdinario p): void;
* agregarProductoCarrito(int volumen, double precio, int porcentaje, int adicional): void;
* agregarProductoEspecial(Integer id, PaqueteEspecial p): void;
* eliminarProductoCarrito(int id): void;
* calcularValorAPagar(): double;
* paqueteEnCarrito (int id): boolean;
* paqueteEntregado(int id): boolean;
* calcularValorPaquete(int id): double;
* obtenerPaquete(int id): Paquete;
* hayNumerosEnString(String s): boolean;
* finalizarPedido(): void;
* toString(): String;

**IREP:**

* nroPedido debe ser un entero único y mayor que 0.
* direccionCliente debe ser un String no null y valido.
* nombreCliente debe ser un String valido.
* agregarProductoOrdinario y agregarProductoEspecial generan excepción si ya existe un paquete con esa id o si el pedido cerro.
* eliminarProductoCarrito genera excepción si el paquete no esta en el carrito o si el pedido está cerrado.

