Ejercicio 1

Enunciado

Caso: El cliente desea devolver latas, botellas y envases.

El proceso comienza cuando el cliente presiona el "botón inicio" en el panel del cliente. Los sensores incorporados en el panel se activan.

El cliente puede devolver elementos (botella, envase, etc.) a través del panel. Los sensores informan al sistema que un objeto ha sido insertado, calibran el elemento depositado y devuelven el resultado al sistema. El sistema utiliza el resultado medido para determinar el tipo de elemento devuelto.

El total diario de elementos depositados se incrementa, así como el total de elementos que el cliente ha depositado.

Cuando el cliente ha depositado todos los elementos a devolver, solicita el recibo presionando el "botón recibo". El sistema procesa la información que debe imprimirse en el recibo. Por cada tipo de elemento depositado extrae su valor de devolución y el número de elementos depositados por el cliente actual. Se imprime el recibo con el detalle y el total de los elementos devueltos y el proceso finaliza.

Diagrama

Sistema

totalDiarioElemsDepositados: int totalElemsDepositadosCliente: int enProcesoDeDevolucion: boolean valorTotalDevuelto: boolean

botonInicio(): void botonRecibo(): int

incrementarDepositosDiarios(): void incrementarDepositosCliente(): void devolverElemento(Botella): void devolverElemento(Lata): void devolverElemento(Envase) : void

Recibo

totalElementosDepositados: int valorTotal: double

getTotalElementosDepositados(): int getValorTotal(): double

Botella

tipoElemento: String valorDevolucion: double

getTipo(): String

getValorDevolucion(): double

Envase

tipoElemento: String valorDevolucion: double

getTipo(): String

getValorDevolucion(): double

Lata

tipoElemento: String valorDevolucion: double

getTipo(): String

getValorDevolucion(): double

Ejercicio 2

Enunciado

Caso: Gestionar todos los trámites académicos de los alumnos.

Se debe desarrollar una aplicación para realizar los trámites de altas, bajas, modificaciones y consultas de los datos de los mismos.

Debemos asignar un número de trámite único en el sistema y una fecha de inicio. La información mínima que se debe guardar de un alumno son el DNI, nombre, dirección, la carrera en la que está matriculado, así como las asignaturas que está cursando actualmente. También se debe almacenar su historial académico, donde deben aparecer todas las asignaturas cursadas y sus respectivas notas y exámenes finales.

Debe tenerse en cuenta que una carrera sólo se da en una sede determinada. Se tienen que gestionar las asignaturas que se imparten en una carrera, teniendo en cuenta que una asignatura sólo se puede dar en un único curso. Algunos de los datos que pueden consultarse de una asignatura son: nombre, cantidad de horas, cuatrimestre en el que se dicta y tipo (obligatoria, transversal, optativa). Debe guardarse la información de los profesores que dictan las distintas asignaturas de la carrera, como mínimo: DNI, nombre, dirección y departamento al que pertenece. También se podrá consultar las distintas asignaturas que imparte.

Diagrama

Alumno

dni: String
nombre: String
direccion: String
carrera: Carrera
asignaturasEnCurso: List<Asignatura>
historialAcademico: HistorialAcademico

Profesor

dni: String nombre: String direccion: String departamento: String asignaturasImpartidas: List<Asignatura>

Curso

codigoCurso: String profesorAsignado: Profesor

Universidad

alumnos: Map<String, Alumno> profesores: Map<String, Profesor>

tramite: Tramite

darAltaAlumno(Alumno): void darBajaAlumno(String): void modificarAlumno(String, Alumno): void consultarAlumno(String): Alumno consultarAsignaturasProfesor(String): List<Asignatura>

getNroTramite(): int getFechalnicio(): Date

Asignatura

nombre: String cantHoras: int cuatrimestre: String tipo: String notaCursada: double notaFinal: double cursoAsignado: Curso

Carrera

nombreCarrera: String

asignaturas|mpartidas: List<Asignatura>

Tramite

nroTramite: int fechalnicio: Date

HistorialAcademico

asignaturasCursadas: List<Asignatura>

Sede

nombreSede: String ubicacion: String

Ejercicio 3

Enunciado

Caso: Gestionar productos asociados a los clientes de un banco.

Los productos que gestiona el banco son: cuentas bancarias, fondos de inversión y carteras de valores.

Las cuentas deben tener: número de cuenta, fecha de apertura, saldo y tipo de interés y los datos de sus clientes. El banco tiene dos tipos de cuentas: corrientes y cajas de ahorro.

Las cuentas corrientes pueden tener tarjetas de crédito asociadas. Solo estas cuentas pueden tener el resto de los productos asociados.

Las cajas de ahorro deben tener el número de meses que estarán abiertas. De los clientes se debe almacenar la siguiente información: DNI, nombre, dirección y teléfono.

Los fondos de inversión deben tener un nombre, importe, rentabilidad y la fecha de apertura y vencimiento.

Las carteras de valores están compuestas por los valores asociados, almacenando el nombre del valor, el número de títulos y el precio de cotización.

Las tarjetas de crédito deben almacenar el tipo (Visa, MasterCard, etc.), el número, el titular y la fecha de caducidad.

Diagrama

Banco

cuentasCorrientes: Map<String, CuentaCorriente> cajasAhorro: Map<String, CajaAhorro> clientes: Map<String, Cliente>

altaCliente(Cliente): void bajaCliente(String): void

modificarCliente(String, Cliente): void consultarCliente(String): Cliente

consultarCuentaCorriente(String): CuentaCorriente consultarCajaAhorro(String): CajaAhorro

CarteraValor

valores: List<Valor>

agregarValor(Valor): void quitarValor(Valor): void

Valor

nombre: String nroTitulos: int precio: double

TarjetaCredito

tipo: String numero: String titular: Cliente fechaCaducidad: Date

CuentaCorriente

nroCuenta: String fechaApertura: Date saldo: double tipoInteres: String cliente: Cliente

tarjetasCredito: List<TarjetaCredito> fondosDeInversion: List<FondoInversion> carterasDeValores: List<CarteraValor>

extraerSaldo(double): void depositarSaldo(double):void altaTarjeta(TarjetaCredito): void bajaTarjeta(TarjetaCredito): void altaFondo(FondoInversion): void bajaFondo(FondoInversion): void

CajaAhorro

nroCuenta: String fechaApertura: Date saldo: double tipoInteres: String cliente: Cliente nroMesesAbierta: int

Cliente

dni: String nombre: String direccion: String telefono: String

FondoInversion

nombre: String importe: double rentabilidad: double fechaApertura: Date fechaVencimiento: Date