DISCON 2013

Especificació Memòria Codi i Test Errors

Versió 1.8

1- Historial de Revisions

Data	Versió	Descripció	Autor
27/02/2013	0.0	Esquelet Projecte	TOTS
10/04/2013	1.0	Primera revisió d'errors	TOTS
12/04/2013	1.1	Revisió i correcció dels errors	TOTS
15/04/2013	1.2	Creació dels Test	TOTS
20/05/2013	1.3	Modificació del text	TOTS
26/05/2013	1.4	Modificació projecte	TOTS
27/05/2013	1.5	Retocs finals	TOTS
02/06/2013	1.6	Modificació projecte	TOTS
03/06/2013	1.7	Arreglar Test mal implementats	TOTS
04/06/2013	1.8	Implementació d'un test des de 0	TOTS

2- Índex

3- Introducció

La companyia comercial Discon Market treballa des de fa 25 anys en la venta de productes a particulars i disposa d'un establiment de venda al públic. La filosofia de l'empresa es: "Tots els productes que tenim estan a disposició del client", es a dir, que tot el local comercial ha de ser dedicat a la venda, fins i tot el magatzem.
La venda es realitza en el mateix local els dies laborables de 10 : 00 a 14 : 00 i de 17:00 a 22 : 00.
La companyia vol disposar d'un sistema informàtic per millorar la gestió de l'empresa i continuar creixent.
La companyia esta dividida des del punt de vista organitzatiu en quatre departaments:

1. Gestió de vendes.
2. Gestió d'entregues.
3. Gestió logística.
4. Gestió contable.

4- Errors modificats del codi

Classe Catàleg

Descripció general:

Aquesta classe ens mostrarà el catàleg a partir d'un fitxer. A cada línia del fitxer tenim guardat la informació dels productes que tenim al catàleg.

Canvis implementats:

- El primer canvi realitzat ha sigut modificar l'ordre de les posicions de les variables "idProd" i el "nombreProd" en la llista on tenim guardat el catàleg. S'ha de canviar per poder tractar la informació del producte en altres funcions.

A més hem modificat les següents funcions:

- writeProductToTxt()

Aquesta funció modificarà la forma en la que inserim el nostre producte dintre del txt i també la forma en la que el tractem.

- readProductStr()

Llegeix el producte i el passa a String perquè el puguem tractar.

- GetProd()

Hem posat aquest mètode com a "públic" per tal de poder accedir al objecte "producte" a tot el programa.

Report d'errors:

El principal error al executar per primer cop, és que no teníem un document a on llegir i guardar els productes. Una vegada generat aquest document, podem accedir al catàleg i poder modificar l'stock.

Classe Client

Descripció general:

A través de uns paràmetres obtindrem les dades de un determinat client, a més aquesta classe contindrà dades com pot esser el login i el compte bancari.

Canvis implementats:

Hem creat variables privades de la classe que feien hem fet les següents implementacions:

- Inicialització de les variables de la classe.

- Mètodes "Sets" i "Gets":
  Aquestes modificacions es fan a partir de les noves variables que hem creat.

- Mètode "Identificarse()":
  Sense aquest mètode no podem comprovar que el client introduït està guardat en el registre.

Report d'errors:

Al inserir N usuaris el sistema no els reconeix correctament.

Classe Compra

Descripció general:

Classe que permetrà fer tots els passos oportuns per a poder definir una compra ja sigui mirant el carro o guardant les dades en un arxiu seleccionat.

Canvis implementats:

Hem modificat les següents funcions:

- EfectuarCompra().
- RegistrarCompra().
- AfegirProducteCarro().

Amb aquestes funcions podrem modificar el stock que hi ha dintre del nostre .txt de productes i a més podrem guardar al .txt del registre de les compres, les compres realitzades per un usuari.

Report d'errors:

Sense fitxer inicial el numero de compres no es realitzava.

Error en la comptabilitat del producte, per solucionar aquest error es van tenir que canviar alguns paràmetres dintre de la classe.

Classe Constantes

Descripció general:

Classe que conté totes les rutes necessàries tant de fitxers com a variables publiques estàtiques i també variables publiques finals. Això ho fem per poder accedir a elles des de qualsevol part del nostre programa i poder utilitzar-les com si fos un arxiu de configuració.

Canvis implementats:

Per la realització d'aquesta classe hem creat les següents constatns:

```
public class Constantes {
	private static final String Ruta = "src/org/uab/etse/es/ES00/";
	public static final String RutaClient = Ruta + "clients.txt";
	public static final String RutaProductes = Ruta + "productes.txt";
	public static final String RutaRegistros = Ruta + "registro.txt";
}
```

Classe GestionarCompres

Descripció general:

Aquesta classe ens permet modificar i gestionar amb certa facilitat les compres efectuades per un client. A partir de llegir les dades del client i veure els productes que hi ha al catàleg.

Canvis implementats:

- Dins el mètode “RegistrarClient” hem canviat l’ordre de l’incrementador ”pos” del bucle principal del mètode. Amb això ens assegurem de recórrer tota la llista d’usuaris registrats.
- Hem modificat la classe amb la realització del patró "Singleton" de manera que, el constructor ha passat de ser un mètode public a un privat i implementant una funció estàtica amb el nom "getInstance".

Report d’errors:

Mal funcionament de la identificació del client.

Report de versions:

- V.1: Codi realitzat fins al moment.
- V.1.2: Aplicació del model "Singleton".

Classe LiniaCompra

Descripció general:

Classe que guarda la informació, el nombre d'unitats i el preu final d'un producte seleccionat per l'usuari.

Canvis implementats:

Hem implementat totes els mètodes de la classe, ya que no estaven fets.

- Creació del constructor de la classe.
- Creació de les totes les funcions set() i get() de les variables privades de la classe (unitat, IDlinea, producte).
- Creació del mètode GetPreu(), que calcula el preu de la línea de compra (la suma del preu del producte per el número d'unitats que comprem).

Report d'errors:

Creació dels nous mètodes per poder obtenir el preu de la línia de compra i les dades del producte amb els get() i set().

Classe Main

Descripció general:

Classe principal on cridarem seqüencialment funcions del nostre codi per executar el nostre programa. A mes a mes hemimplementat una petita interfície en la que interactuarem amb un usuari que es podrà loguejar, registrar o consultar les nostres opcions.

Canvis implementats:

La funció "DesconectarClient()" la hem inserit dintre d'aquesta classe a la opció 6 del menú perquè no ens doni errors

- Modificació de la variable "gestion". Hem passat la variable de "private static" a "public static", ja que necessitem la variable a tot el programa.
- Modifiquem el tipus de tractat de la condició "col·leccions" per utilitzar arrays.
- Afegim dintre del nostre menú principal el cas 7 perquè ens permeti sortir de l'aplicació.
- Tornem a modificar la variable "gestion" passant-la de tipus "public static" a tipus "private static", ja que ara el sistema te una estructura "singleton" i no necessita que sigui publica. Finalment, això fa que el nostre codi sigui més sòlid.

Report d'errors:

Error al loguejar i desloguejar.
Error dintre de la compra de productes.
Error al actualitzar els txt i comprovar les dades.

Report de versions:

- V.1: Codi realitzant fins al moment.
- V.1.2: Canvi de variables

Classe Productes

Descripció general:

A partir d'un arxiu txt anirem actualitzant i obtenint les dades que tenen els productes, com el seu nom, el stock, el preu entre altres funcions.

Canvis implementats:

- Modificació de la funció "UpdateProdTxt()":
  Canviem l'ordre de les posicions del "Getcost()" i "GetUnitats()" dins de la funció.

- Modificació de la funció "IdProdStr()":
  Hem modificat: ProdDades[**1**] -> ProdDades[**0**]. Així, en retornarà l'ID del nostre producte en comptes del nom del nostre producte.

- Implementació de la funció "rutaFitxer()":
  Utilitzem aquesta funció per obtenir la ruta del fitxer on es troba la llista de productes del catàleg, per poder accedir-hi còmodament des de qualsevol funció.

Report d'errors:

Error en la contabilitat del producte i en el tractament de els dades.

Altres

Per poder generar tot el nostre sistema hem creat 3 fitxers anomenats:

- Clients.txt
- Productes.txt
- Registro.txt

Aquests fitxers s'aniran actualitzant amb les dades noves que poden ser d'un usuari, d'un producte o un registre de compra.

Aquest es un dels primers errors que vam trobar al codi, sense aquests fitxers, el codi no executava res.

5- Test

Classe ConstantesTests

Descripció general:

Classe Test efectuada per comprovar que la classe Constantes no tenia cap error per això hem aplicat uns determinats algorismes.

Canvis implementats:

Aquest test lo que ens farà serà comprovar que hi ha una ruta o un objecte que existeix.

Classe TestCataleg

Descripció general:

Test basat en la classe catàleg que buscarà alguna vulnerabilitat o error dins del nostre codi.

Canvis implementats:

- Hem inicialitzar els vectors "names" i "description" com "Vector<String> i, el vector "prizes" com "Vector<Float>

- Hem afegit els .txt necessaris per comprovar els test i hem creat la classe "ConstantesTests.java" on declarem les rutes dels arxius.

- Hem inicialitzat la variable "catalegFilename" amb la ruta que tenim guardada a la classe "ConstantesTests.java"

- Hem fet una crida a la funció "testAfegirProducte" a la funció principal del test.

- A la funció "testContainsProducte" hem iniciat l'iterador com "Iterator<Producte>".

- Implementem la funció "TestSetProducte", que afegeix a la llista un nou producte amb el nom, la descripció i el preu que l'introduim.

- Modifiquem la funció "testEliminarProductes", a la que afegim el paràmetre "String numero" per poder passar-li a la funció "eliminarProducto" dins de la classe "Cataleg"

Classe TestClient

Descripció general:

Test que comprova si la classe client conte algun error.

Canvis implementats:

- L'únic canvi que hem tingut que realitzar ha sigut en el mètode ”SetUp”, on hem afegit tots els test de la classe (“testDefaultClient” , “testFull”, “testModifiers”, “testIdentificar”, “testCompra”).

Classe TestProducte

Descripció general:

Classe que obtindrà la comprovació que un producte es correcte o esta ben definit, sense cap error.

Canvis implementats:

- S'ha tingut que implementar tots els test dels mètodes de la classe Producte, ja que no estaven fets i comprovem que les dades, que s'introdueixen del producte, són correctes.

- Hem creat la subclasse "ProducteTest" que hereta de la classe "Producte" i hem fet les següents modificacions:

- Hem sobrecarregat el mètode "rutaFitxer()" per a utilitzar un fitxer propi del test i no modificar directament el fitxer "Producte.txt" del programa principal.
- Hem implementat el mètode "comprobarUpdate()" per poder comprovar que cada vegada que fem una modificació en el .txt (amb els mètodes "Set()" de la classe), el que s'ha guardat sigui el mateix que lo que hem afegit.

Classe TestLiniaCompra

Descripció general:

Classe que comprova la correcta implementació de la classe "LiniaCompra" i dels cadascun dels seus mètodes.

Canvis implementats:

Implementació de tota la classe amb els tests que comproven tots els mètodes de la classe a comprovar, en aquest cas "LiniaCompra". Tests implementats:

- "testGetProducte()", comprova que s'ha generat correctament la línia de compra del producte insertat per l'usuari.

- "testSetProducte()", comprova que aquest mètode està ven implementat i que, per tant, podem modificar el producte manualment sense errors.

- "testGetCompra()", comprova que s'ha generat correctament la línia de compra del producte dins del objecte "compra" de l'usuari.

- "testSetCompra()", comprova que aquest mètode està ven implementat i que, per tant, podem modificar la línia de compra del objecte "compra".

- "testGetUnitats()", comprova que s'han generat correctament les unitats desitjades per l'usuari del producte que ha seleccionat.

- "testSetUnitats()", comprova que aquest mètode està ven implementat i que, per tant, podem modificar les unitats del producte seleccionat per l'usuari.

- "testGetIDlineaCompra()", comprova que s'ha generat correctament la ID de la línia de compra d'un producte.

- "testSetIDlineaCompra()", comprova que aquest mètode està ven implementat i que, per tant, podem modificar la ID de la línia de compra d'un producte.

- "testGetPreu()", comprova que aquest mètode està ven implementat i que, per tant, es genera correctament el preu de la línia de compra del producte a partir del preu de cada unitat i de les unitats que s'hagin escogit.

6- Conclusió

Una vegada realitzat la feina d'implementació del codi de l'anterior equip de treball, podem afirmar que el codi estava mol malt implementat i que s'han modificat una gran quantitat de línies del programa. Després d'acabar d'arreglar els errors, hem hagut d'acabar de realitzar els tests per confirma que s'ha fet una bona feina amb el codi i que tot el programa està llest per al seu posterior funcionament.