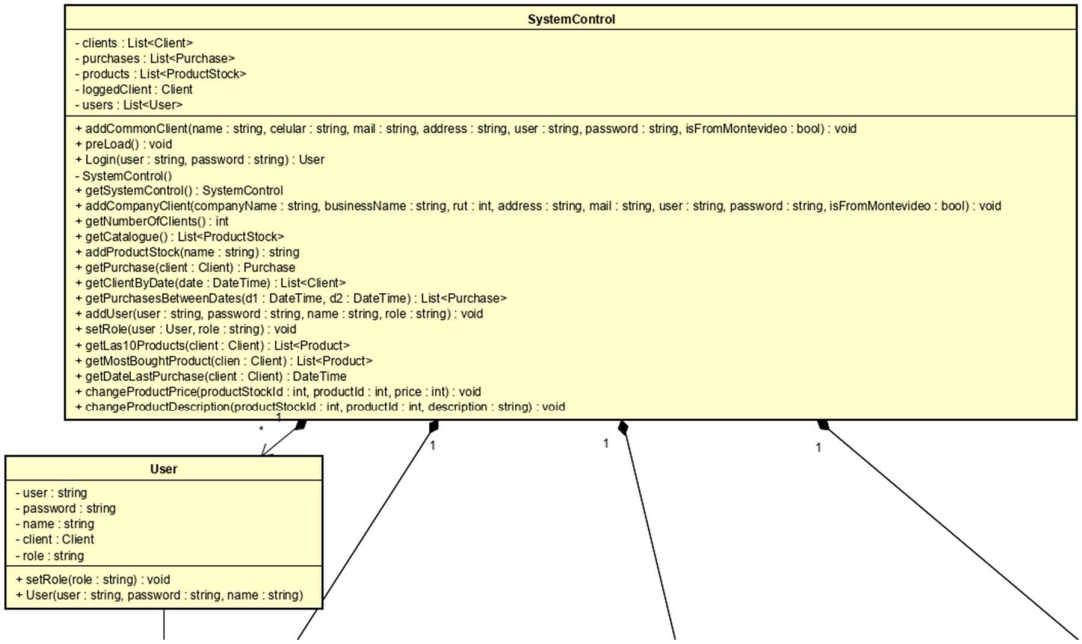
Federico García – 242015

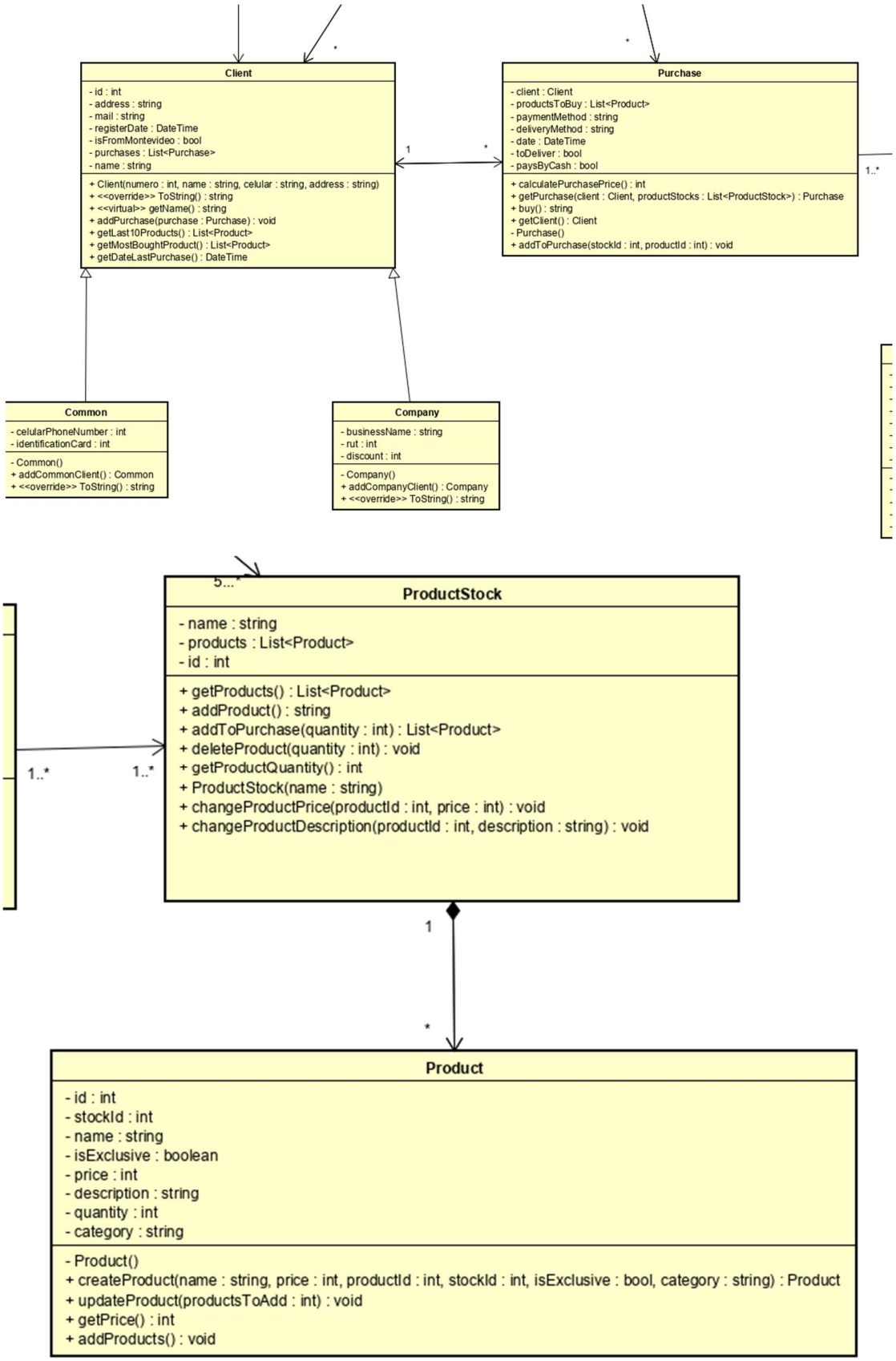
Andrés Valle Lisboa – 246109

Web:obligatorio2p2avlfg.somee.com

Profesor: Luis Dentone

UML




Client
- id : int
- address : string
- mail : string
- registerDate : DateTime
- isFromMontevideo : bool
- purchases : List<Purchase>
- name : string
+ Client(numero : int, name : string, celular : string, address : string)
+ <<override>> ToString() : string
+ <<virtual>> getName() : string
+ addPurchase(purchase : Purchase) : void
+ getLast10Products() : List<Product>
+ getMostBoughtProduct() : List<Product>
+ getDateLastPurchase() : DateTime
Purchase
- client : Client
- productsToBuy : List<Product>
- paymentMethod : string
- deliveryMethod : string
- date : DateTime
- toDeliver : bool
- paysByCash : bool
+ calculatePurchasePrice() : int
+ getPurchase(client : Client, productStocks : List<ProductStock>) : Purchase
+ buy() : string
+ getClient() : Client
- Purchase()
+ addToPurchase(stockId : int, productId : int) : void
1
*
1..*
Common
- celularPhoneNumber : int
- identificationCard : int
- Common()
+ addCommonClient() : Common
+ <<override>> ToString() : string
Company
- businessName : string
- rut : int
- discount : int
- Company()
+ addCompanyClient() : Company
+ <<override>> ToString() : string
5...*
ProductStock
- name : string
- products : List<Product>
- id : int
+ getProducts() : List<Product>
+ addProduct() : string
+ addToPurchase(quantity : int) : List<Product>
+ deleteProduct(quantity : int) : void
+ getProductQuantity() : int
+ ProductStock(name : string)
+ changeProductPrice(productId : int, price : int) : void
+ changeProductDescription(productId : int, description : string) : void
1..*
1..*
1
*
Product
- id : int
- stockId : int
- name : string
- isExclusive : boolean
- price : int
- description : string
- quantity : int
- category : string
- Product()
+ createProduct(name : string, price : int, productId : int, stockId : int, isExclusive : bool, category : string) : Product
+ updateProduct(productsToAdd : int) : void
+ getPrice() : int
+ addProducts() : void

Dominio

Models

Client.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ShopSystem
{
    public class Client
    {
        private int id;
        private string name;
        private string phone;
        private string address;
        private string mail;
        private DateTime registerDate;
        private bool isFromMontevideo;
        private List<Purchase> purchases = new List<Purchase>();

        public class clientValidation
        {
            public clientValidation(bool isMailUsed)
            {
                this.isMailUsed = isMailUsed;
            }
            public bool isMailUsed;
        }

        public int Id { get { return id; } }
        public string Name { get { return name; } }
        public string Phone { get { return phone; } }
        public string Address { get { return address; } }
        public string Mail { get { return mail; } }
        public DateTime RegisterDate { get { return registerDate; } }
        public bool IsFromMontevideo { get { return isFromMontevideo; } }
        public List<Purchase> Purchases { get { return purchases; } }

        internal Client(int id, string name, string address, string mail, string phone, bool isFromMontevideo)
        {
            this.id = id;
            this.name = name;
            this.address = address;
            this.mail = mail;
            this.phone = phone;
            this.isFromMontevideo = isFromMontevideo;
            registerDate = DateTime.Today;
        }

        public static clientValidation isInformationCorrect(List<Client> clients, string user, string mail)
        {
            int id = clients.Count;
            bool isMailUsed = false;
            foreach (Client c in clients)
            {
                if (c.Mail == mail)
                {
                    isMailUsed = true;
                }
                if (isMailUsed) break;
```

```
            }
            clientValidation clientValidation = new clientValidation(isMailUsed);
            return clientValidation;
        }
        public void addPurchase(Purchase purchase)
        {
            purchases.Add(purchase);
        }
        public List<Product> getLast10Products()
        {
            List<Product> last10Products = new List<Product>();
            int productsAddedQuantity = 0;
            foreach (Purchase c in purchases)
            {
                if (productsAddedQuantity == 10) break;
                foreach (Product p in c.ProductsToBuy)
                {
                    if (productsAddedQuantity == 10) break;
                    last10Products.Add(p);
                    productsAddedQuantity++;
                }
            }
            return last10Products;
        }
        public List<Product> getMostBoughtProduct()
        {
            List<Product> mostBoughtProducts = new List<Product>();
            int mostBoughtProductQuantity = 0;
            foreach (Purchase c in purchases)
            {
                foreach (var p in c.GetDataProductsToBuy)
                {
                    if (mostBoughtProductQuantity < p.quantity)
                    {
                        mostBoughtProductQuantity = p.quantity;
                        mostBoughtProducts.Clear();
                        foreach (Product _p in c.ProductsToBuy)
                        {
                            if (p.productId == _p.Id) mostBoughtProducts.Add(_p);
                        }
                    }
                    else if (mostBoughtProductQuantity == p.quantity)
                    {
                        foreach (Product _p in c.ProductsToBuy)
                        {
                            if (p.productId == _p.Id)
                            {
                                mostBoughtProducts.Add(_p);
                                break;
                            }
                        }
                    }
                }
            }
            return mostBoughtProducts;
```

```
        }

        public DateTime getDateLastPurchase()
        {
            DateTime lastPurchaseDate = new DateTime();
            int count = purchases.Count;
            for (int i = 0; i < count; i++)
            {
                if(i == 0) lastPurchaseDate = purchases[i].Date;
                else
                {
                    int result = DateTime.Compare(purchases[i].Date, purchases[i - 1].Date);
                    if (result > 0) lastPurchaseDate = purchases[i].Date;
                }
            }
            return lastPurchaseDate;
        }

        public override string ToString()
        {
            return base.ToString();
        }
    }
}

```

Commons.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ShopSystem
{
    public class Common : Client
    {
        private string address;
        private int identificationCard;
        private Common(int id, string name, int identificationCard, string phone, string address, string mail, bool isFromMontevideo) : base(id, name, address, mail, phone, isFromMontevideo)
        {
            this.address = address;
        }

        public int IdentificationCard { get { return identificationCard; } }

        public class commonValidation
        {
            public commonValidation(bool isIdentificationCardUsed)
            {
                this.isIdentificationCardUsed = isIdentificationCardUsed;
            }
            public bool isIdentificationCardUsed;
        }

        public static commonValidation isInformationCorrect(List<Common> clients, int identificationCard)
        {
            int id = clients.Count;
            bool isIdentificationCardUsed = false; ;
            foreach (Common c in clients)
            {
                if (c.GetType() == typeof(Common))
                {

                    if (c.IdentificationCard == identificationCard)
                    {
                        isIdentificationCardUsed = true;
                        break;
                    }
                }

            }
            commonValidation commonValidation = new commonValidation(isIdentificationCardUsed);
            return commonValidation;
        }

        public static Common AddCommonClient(int id, string name, int identificationCard, string phone, string address, string mail, string user, string password, bool isFromMontevideo)
        {
            return new Common(id, name, identificationCard, phone, address, mail, isFromMontevideo);
        }

        public override string ToString()
        {
            return "Common";
        }

    }
}
```

Company.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ShopSystem
{
    public class Company : Client
    {
        private string bussinesName;
        private int rut;
        private int discount;

        public string BussinesName { get { return bussinesName; } }
        public int Rut { get { return rut; } }
        public int Discount { get { return discount; } }

        public class companyValidation
        {
            public companyValidation(bool isBussinesNameUsed, bool isRutUsed)
            {
                this.isBussinesNameUsed = isBussinesNameUsed;
                this.isRutUsed = isRutUsed;
            }
            public bool isBussinesNameUsed;
            public bool isRutUsed;
        }

        public static companyValidation isInformationCorrect(List<Company> clients, string
        {
            int id = clients.Count;
            bool isBussinesNameUsed = false;
            bool isRutUsed = false;
            foreach (Company c in clients)
            {
                if (c.GetType() == typeof(Company))
                {
                    if (c.bussinesName == bussinesName)
                    {
                        isBussinesNameUsed = true;
                        break;
                    }
                    if (c.rut == rut)
                    {
                        isRutUsed = true;
                        break;
                    }
                }

```

```
            }
            companyValidation companyValidation = new companyValidation(isBussinesNameUsed, isRutUsed);
            return companyValidation;
        }

        private Company(int id, string companyName, string bussinesName, int rut, string address, string mail, string phone, string user, string password, bool isFromMontevideo, int discount) : base(id, companyName, address, mail, phone, isFromMontevideo)
        {
            this.bussinesName = bussinesName;
            this.rut = rut;
        }

        public static Company AddCompanyClient(int id, string companyName, string bussinesName, int rut, string address, string mail, string phone, string user, string password, bool isFromMontevideo, int discount)
        {
            return new Company(id, companyName, bussinesName, rut, address, mail, phone, user, password, isFromMontevideo, discount);
        }

        public override string ToString()
        {
            return "Company";
        }
    }
}
```

Product.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ShopSystem
{
    public class Product
    {
        private int id;
        private int stockId;
        private string name;
        private int price;
        private string description;
        private bool isExclusive;
        private int quantity;

        public int Id { get { return id; } set { id = value; } }
        public int StockId { get { return stockId; } }
        public string Name { get { return name; } }
        public int Price { get { return price; } set { price = value; } }
        public string Description { get { return description; } set { description = value; } }
        public int Quantity { get { return quantity; } }
        public bool IsExclusive { get { return isExclusive; } }

        private Product(string name, int id, int stockId, int price, string description, bool isExclusive)
        {
            this.id = id;
            this.stockId = stockId;
            this.name = name;
            this.price = price;
            this.description = description;
            this.isExclusive = isExclusive;
            this.quantity = 0;
        }

        public void addProducts(int quantity)
        {
            this.quantity += quantity;
        }

        public static Product createProduct(string name, int id, int stockId, int price, string description, bool isExclusive)
        {
            Product product = new Product(name, id, stockId, price, description, isExclusive);
            return product;
        }

        public void removeProducts(int quantity)
        {
            this.quantity -= quantity;
        }
    }
}
```

ProductStock.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ShopSystem
{
    public class ProductStock
    {
        private List<Product> products = new List<Product>();
        private int stockId;
        private string name;
        public class _product
        {
            public int id;
            public int quantity;
            public int price;
            public bool wasAdded;
        }

        public int StockId { get { return stockId; } }
        public string Name { get { return name; } }
        public List<Product> Products { get { return products; } }
        public int ProductsQuantity { get { return products.Count; } }

        public string addProduct(string name, int price, string description, bool isExclusive, int quantity) //agregas productos
        {
            int id = products.Count;
            bool productExists = false;
            if (name == "" || name == " " || name.Length < 4) return "The name must be valid";
            else
            {
                if (price > 0)
                {
                    foreach (Product p in products)
                    {
                        if (p.Name == name) productExists = true;
                        break;
                    }
                    if (!productExists)
                    {
                        products.Add(Product.createProduct(name, id, stockId, price, description, isExclusive));
                        products[products.Count - 1].addProducts(quantity);
                        return "The product was added correctly";
                    }
                    else return "The product already exists";
                }
```

```
47                  else return "The price must be greater than 0";
48              }
49          }
50
51          public _product addToPurchase(int quantity, int productId)
52          {
53              int count = products.Count;
54              int price = 0;
55              bool wasFounded = false;
56              foreach (Product p in products)
57              {
58                  if (p.Id == productId)
59                  {
60                      wasFounded = true;
61                      price = p.Price;
62                      break;
63                  }
64              }
65              if (wasFounded)
66              {
67                  _product productAdded = new _product();
68                  productAdded.id = productId;
69                  productAdded.quantity = quantity;
70                  productAdded.price = price;
71                  productAdded.wasAdded = true;
72                  return productAdded;
73              }
74              else
75              {
76                  _product productAdded = new _product();
77                  productAdded.wasAdded = false;
78                  return productAdded;
79              }
80          }
81
82          public ProductStock(string name, int id)
83          {
84              this.name = name;
85              this.stockId = id;
86          }
87
88          public string removeProduct(int id, int quantity)
89          {
90              int productsNumber = Products.Count;
91              if (quantity > productsNumber) return "There are not enough products";
92              else
```

```
93                  {
94                      for (int i = 0; i < productsNumber; i++)
95                      {
96                          if (Products[i].Id == id) Products.RemoveAt(i);
97                      }
98                      return "Products added";
99                  }
100             }
101
102             public void changeProductPrice(int productId, int productPrice)
103             {
104                 products[productId].Price = productPrice;
105             }
106
107             public void changeProductDescription(int productId, string description)
108             {
109                 products[productId].Description = description;
110             }
111         }
112     }
113
```

Purchase.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ShopSystem
{
    public class Purchase
    {
        private Client client;
        private List<ProductStock> productStocks = new List<ProductStock>();
        private List<DataProductsToBuy> dataProductsToBuy = new List<DataProductsToBuy>();
        private List<Product> productsToBuy = new List<Product>();
        private int totalPrice = 0;
        private int id;
        private DateTime date;
        private bool toDeliver;
        private bool paysByCash;
        private Purchase(Client client, List<ProductStock> productStocks, int id)
        {
            this.client = client;
            this.productStocks = productStocks;
            this.date = DateTime.Today;
            this.id = id;
        }
        public class DataProductsToBuy
        {
            public int productId;
            public int stockId;
            public int quantity;
        }

        public Client Client { get { return client; } }
        public int TotalPrice { get { return totalPrice; } }
        public bool ToDeliver { get { return toDeliver; } set { toDeliver = value; } }
        public bool PaysByCash { get { return paysByCash; } set { paysByCash = value; } }
        public DateTime Date { get { return date; } }
        public int ProductsQuantity { get { return dataProductsToBuy.Count; } }
        public List<Product> ProductsToBuy { get { return productsToBuy; } }
        public List<DataProductsToBuy> GetDataProductsToBuy { get { return dataProductsToBuy; } }
        public int Id { get { return id; } }

        private int calculatePurchasePrice()
        {
            return totalPrice;
        }

        public static Purchase getPurchase(Client client, List<ProductStock> productStocks, int id)
```

```
        {
            return new Purchase(client, productStocks, id);
        }

        public string buy()
        {
            int productsToBuyNumber = dataProductsToBuy.Count;
            int discount = 0;
            for (int i = 0; i < productsToBuyNumber; i++)
            {
                int productId = dataProductsToBuy[i].productId;
                int stockId = dataProductsToBuy[i].stockId;
                int quantity = dataProductsToBuy[i].quantity;
                productStocks[stockId].removeProduct(productId, quantity);
            }
            if (paysByCash && totalPrice > 5000) discount += 4;
            if (((client.RegisterDate - DateTime.Today).TotalDays / 365) > 2) discount += 5;
            if (client.GetType() == typeof(Common) && !(client.IsFromMontevideo)) discount += 5;
            if (client.GetType() == typeof(Company) && ((client.RegisterDate - DateTime.Today).TotalDays / 365) > 5) discount += ((Company)client).Discount * 2;
            else if (client.GetType() == typeof(Company)) discount += ((Company)client).Discount;
            totalPrice = (100 - discount) * totalPrice / 100;
            if (!(client.IsFromMontevideo) && toDeliver) totalPrice += 1000;
            return "Debe pagar $" + totalPrice;
        }

        public string addToPurchase(int stockId, int productId, int quantity)
        {
            var _product = productStocks[stockId].addToPurchase(quantity, productId);
            productsToBuy.Add(productStocks[stockId].Products[productId]);
            if (_product.wasAdded)
            {
                DataProductsToBuy p = new DataProductsToBuy();
                p.productId = _product.id;
                p.stockId = stockId;
                p.quantity = _product.quantity;
                dataProductsToBuy.Add(p);
                if (productStocks[stockId].Products[productId].IsExclusive && _product.quantity > 1)
                {
                    if (_product.quantity > 1) totalPrice += (_product.price) * (_product.quantity - 1);
                }
                else totalPrice += (_product.price) * (_product.quantity);
                return "The products were added correctly";
            }
            else return "The product could not be added";
        }
```

```
    }
}
```

SystemControl.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ShopSystem
{
    public class SystemControl
    {
        private SystemControl() { }                                                     //Private constructor
        private static SystemControl _systemControl = new SystemControl();              //Just one instance
        public static SystemControl getSystemControl() { return _systemControl; }      //Get the instance
        private List<Client> clients = new List<Client>();                              //Clients list
        private List<User> users = new List<User>();
        private List<ProductStock> catalogue = new List<ProductStock>();                //Lista de productStocks(las categorías). Para cada categoría hay un productStock
        private List<Purchase> purchases = new List<Purchase>();                        //Purchases list
        private User loggedUser;

        public int NumberOfClients { get { return clients.Count; } }
        public List<ProductStock> Catalogue { get { return catalogue; } }
        public List<Purchase> Purchases { get { return purchases;} }
        public List<User> Users { get { return users; } }

        public class registerStatus
        {
            public registerStatus(bool wasRegisterSuccessful, string message)
            {
                this.wasRegisterSuccessful = wasRegisterSuccessful;
                this.message = message;
            }
            public bool wasRegisterSuccessful;
            public string message;
        }

        public string setRole(User user, string role)
        {
            string message;
            if(role == "Client" && user.Client == null)
            {
                return "Debe asignarle un cliente para poder cambiar su rol a cliente";
            }
            else if(role == "Client" && user.Client != null)
            {
                message = user.setRole("Client");
            }
            else
            {
                message = user.setRole(role);
```

```
        }
        return message;
    }

    public string setRole(User user, string role, Client client)
    {
        string message;
        if(role == "Client" && client == null)
        {
            message = user.setRole(role);
            user.setClient(client);
        }
        else if(role == "Client" && client != null)
        {
            message = "El usuario ya tiene un cliente asignado";
        }
        else
        {
            message = "El usuario debe tener rol cliente para asignarle uno";
        }
        return message;
    }

    public registerStatus addUser(string user, string password, string name, string role)
    {
        int id = users.Count;
        var isUserInformationCorrect = User.isInformationCorrect(users, user);
        bool isUserUsed = isUserInformationCorrect.isUserUsed;
        if (!isUserUsed)
        {
            User _user = new User(user, password, name, role, id);
            users.Add(_user);
            return new registerStatus(true, "El usuario fue registrado correctamente");
        }
        else return new registerStatus(false, "El usuario no fue registrado");
    }

    public registerStatus addCommonClient(string name, int identificationCard, string celular, string mail, string address, string user, string password, bool isFromMontevideo)
    {
        int id = clients.Count;
        var isClientInformationCorrect = Client.isInformationCorrect(clients, user, mail);
        bool isMailUsed = isClientInformationCorrect.isMailUsed;

        List<Common> commonClientList = new List<Common>();
        foreach (Client c in clients)
        {
            if (c.GetType() == typeof(Common))
            {
                commonClientList.Add((Common)c);
            }
        }
        var isCommonClientInformationCorrect = Common.isInformationCorrect(commonClientList, identificationCard);
        bool isIdentificationCardUsed = isCommonClientInformationCorrect.isIdentificationCardUsed;
        if (!isMailUsed && !isIdentificationCardUsed)
        {
            Client _client = Common.AddCommonClient(id, name, identificationCard, celular, address, mail, user, password, isFromMontevideo);
            clients.Add(_client);
            return new registerStatus(true, "El cliente fue registrado correctamente");
        }
        else return new registerStatus(false, "El cliente no fue registrado");
    }

    public registerStatus addCompanyClient(string companyName, string bussinesName, int rut, string mail, string phone, string address, string user, string password, bool isFromMontevideo, int discount)
    {
        int id = clients.Count;
        var isClientInformationCorrect = Client.isInformationCorrect(clients, user, mail);
        bool isMailUsed = isClientInformationCorrect.isMailUsed;

        List<Company> companyClientList = new List<Company>();
        foreach (Client c in clients)
        {
            if (c.GetType() == typeof(Company))
            {
                companyClientList.Add((Company)c);
            }
        }
        var isCompanyClientInformationCorrect = Company.isInformationCorrect(companyClientList, companyName, bussinesName, rut);
        bool isBussinesNameUsed = isCompanyClientInformationCorrect.isBussinesNameUsed;
        bool isRutUsed = isCompanyClientInformationCorrect.isRutUsed;
        if (!isMailUsed && !isBussinesNameUsed && !isRutUsed)
        {
            Client _client = Company.AddCompanyClient(id, companyName, bussinesName, rut, address, mail, phone, user, password, isFromMontevideo, discount);
            clients.Add(_client);
            return new registerStatus(true, "El cliente fue registrado correctamente");
        }
        else return new registerStatus(false, "El cliente no fue registrado");
    }
```

```
        return message;
    }

    public string setRole(User user, string role, Client client)
    {
        string message;
        if(role == "Client" && client == null)
        {
            message = user.setRole(role);
            user.setClient(client);
        }
        else if(role == "Client" && client != null)
        {
            message = "El usuario ya tiene un cliente asignado";
        }
        else
        {
            message = "El usuario debe tener rol cliente para asignarle uno";
        }
        return message;
    }

    public registerStatus addUser(string user, string password, string name, string role)
    {
        int id = users.Count;
        var isUserInformationCorrect = User.isInformationCorrect(users, user);
        bool isUserUsed = isUserInformationCorrect.isUserUsed;
        if (!isUserUsed)
        {
            User _user = new User(user, password, name, role, id);
            users.Add(_user);
            return new registerStatus(true, "El usuario fue registrado correctamente");
        }
        else return new registerStatus(false, "El usuario no fue registrado");
    }

    public registerStatus addCommonClient(string name, int identificationCard, string celular, string mail, string address, string user, string password, bool isFromMontevideo)
    {
        int id = clients.Count;
        var isClientInformationCorrect = Client.isInformationCorrect(clients, user, mail);
        bool isMailUsed = isClientInformationCorrect.isMailUsed;

        List<Common> commonClientList = new List<Common>();
        foreach (Client c in clients)
        {
            if (c.GetType() == typeof(Common))
            {
                commonClientList.Add((Common)c);
            }
        }
        var isCommonClientInformationCorrect = Common.isInformationCorrect(commonClientList, identificationCard);
        bool isIdentificationCardUsed = isCommonClientInformationCorrect.isIdentificationCardUsed;
        if (!isMailUsed && !isIdentificationCardUsed)
        {
            Client _client = Common.AddCommonClient(id, name, identificationCard, celular, address, mail, user, password, isFromMontevideo);
            clients.Add(_client);
            return new registerStatus(true, "El cliente fue registrado correctamente");
        }
        else return new registerStatus(false, "El cliente no fue registrado");
    }

    public registerStatus addCompanyClient(string companyName, string bussinesName, int rut, string mail, string phone, string address, string user, string password, bool isFromMontevideo, int discount)
    {
        int id = clients.Count;
        var isClientInformationCorrect = Client.isInformationCorrect(clients, user, mail);
        bool isMailUsed = isClientInformationCorrect.isMailUsed;

        List<Company> companyClientList = new List<Company>();
        foreach (Client c in clients)
        {
            if (c.GetType() == typeof(Company))
            {
                companyClientList.Add((Company)c);
            }
        }
        var isCompanyClientInformationCorrect = Company.isInformationCorrect(companyClientList, companyName, bussinesName, rut);
        bool isBussinesNameUsed = isCompanyClientInformationCorrect.isBussinesNameUsed;
        bool isRutUsed = isCompanyClientInformationCorrect.isRutUsed;
        if (!isMailUsed && !isBussinesNameUsed && !isRutUsed)
        {
            Client _client = Company.AddCompanyClient(id, companyName, bussinesName, rut, address, mail, phone, user, password, isFromMontevideo, discount);
            clients.Add(_client);
            return new registerStatus(true, "El cliente fue registrado correctamente");
        }
        else return new registerStatus(false, "El cliente no fue registrado");
    }
```

```
147
148         addUser("company1", "company1", "company1", "client");
149         addCompanyClient("company1", "company1 s.a.", 154684654, "company1@gmail.com", "25842143579", "Luis Alberto de Herrera 154843223", "company1", "company1", true, 3);
150             //nombre empresa, razón social, rut, mail, teléfono, address, usuario, contraseña, esDeMontevideo
151         users[3].setClient(clients[3]);
152         addUser("company2", "company2", "company2", "client");
153         addCompanyClient("company2", "company2 s.a.", 878746845, "company2@gmail.com", "65487651321", "Luis Alberto de Herrera 648948455", "company2", "company2", true, 7);
154         users[4].setClient(clients[4]);
155         addUser("company3", "company3", "company3", "client");
156         addCompanyClient("company3", "company3 s.a.", 346456148, "company3@gmail.com", "324564561551", "Luis Alberto de Herrera 878456456", "company3", "company3", false, 5);
157         users[5].setClient(clients[5]);
158
159         addUser("admin", "admin", "Leonardo", "admin");
160         addUser("invitado", "invitado", "Ralph", "guest");
161
162         addProductStock("Frescos"); addProductStock("Congelados"); addProductStock("Hogar"); addProductStock("Téxtiles"); addProductStock("Tecnología"); //Categorías precargadas(1 ProductStock para cada categoría)
163
164         catalogue[0].addProduct("Escarola", 59, "Precio por Kg", false, 100);          //Nombre, precio, descripción, esDeMontevideo                          //Se  agregan los productos
165         catalogue[0].addProduct("Espinaca", 24, "Precio por Kg", false, 100);
166
167         catalogue[1].addProduct("Croquetas", 99, "Precio por Kg", false, 100);
168         catalogue[1].addProduct("Buñuelo", 40, "Precio por Kg", true, 100);
169
170         catalogue[2].addProduct("Detergente", 60, "Precio por L", false, 100);
171         catalogue[2].addProduct("Jabón de manos", 35, "Precio por unidad", true, 100);
172
173         catalogue[3].addProduct("Toallas", 70, "Precio por unidad", false, 100);
174         catalogue[3].addProduct("Sábanas", 150, "Precio por unidad", false, 100);
175
176         catalogue[4].addProduct("PC Gamer", 12500, "Precio por unidad", false, 100);
177         catalogue[4].addProduct("Televisor Led", 15000, "Precio por unidad", false, 10);
178
179         login("jorge", "jorge");                        //Compras de los clientes
180         var purchase1 = getPurchase();
181         purchase1.addToPurchase(0, 0, 10);
182         purchase1.buy();
183         var purchase2 = getPurchase();
```

```
184             purchase2.buy();
185             var purchase7 = getPurchase();
186             purchase7.addToPurchase(1, 0, 12);
187             purchase7.buy();
188             login("javier", "Javier");
189             var purchase3 = getPurchase();
190             purchase3.addToPurchase(1, 0, 10);
191             purchase3.buy();
192             var purchase4 = getPurchase();
193             purchase4.addToPurchase(1, 1, 10);
194             purchase4.buy();
195             login("juan", "juan");
196             var purchase5 = getPurchase();
197             purchase5.addToPurchase(2, 0, 10);
198             purchase5.buy();
199             var purchase6 = getPurchase();
200             purchase6.addToPurchase(2, 1, 10);
201             purchase6.buy();
202         }
203
204         public User login(string user, string password)
205         {
206             bool wasFounded = false;
207             bool isPasswordCorrect = false;
208             User _userLogged = null;
209             foreach (User _user in users)
210             {
211                 if (_user.UserName == user)
212                 {
213                     wasFounded = true;
214                     if (_user.Password == password) isPasswordCorrect = true;
215                     loggedUser = _user;
216                     _userLogged = _user;
217                     break;
218                 }
219             }
220             return _userLogged;
221         }
222
223         private Purchase getPurchase()//Unicamente se utiliza para la precagarga
224         {
225             if (loggedUser != null && loggedUser.Client != null)
226             {
227                 Purchase _purchase = Purchase.getPurchase(loggedUser.Client, catalogue, purchases.Count);
228                 purchases.Add(_purchase);
229                 loggedUser.Client.addPurchase(_purchase);
230                 return _purchase;
231             }
232             else throw new Exception("There is no logged user");
233         }
234
235         public Purchase getPurchase(int id)//Utilizado por el controlador de la web
236         {
237             Purchase _purchase = Purchase.getPurchase(users[id].Client, catalogue, purchases.Count);
238             purchases.Add(_purchase);
239             users[id].Client.addPurchase(_purchase);
240             return _purchase;
```