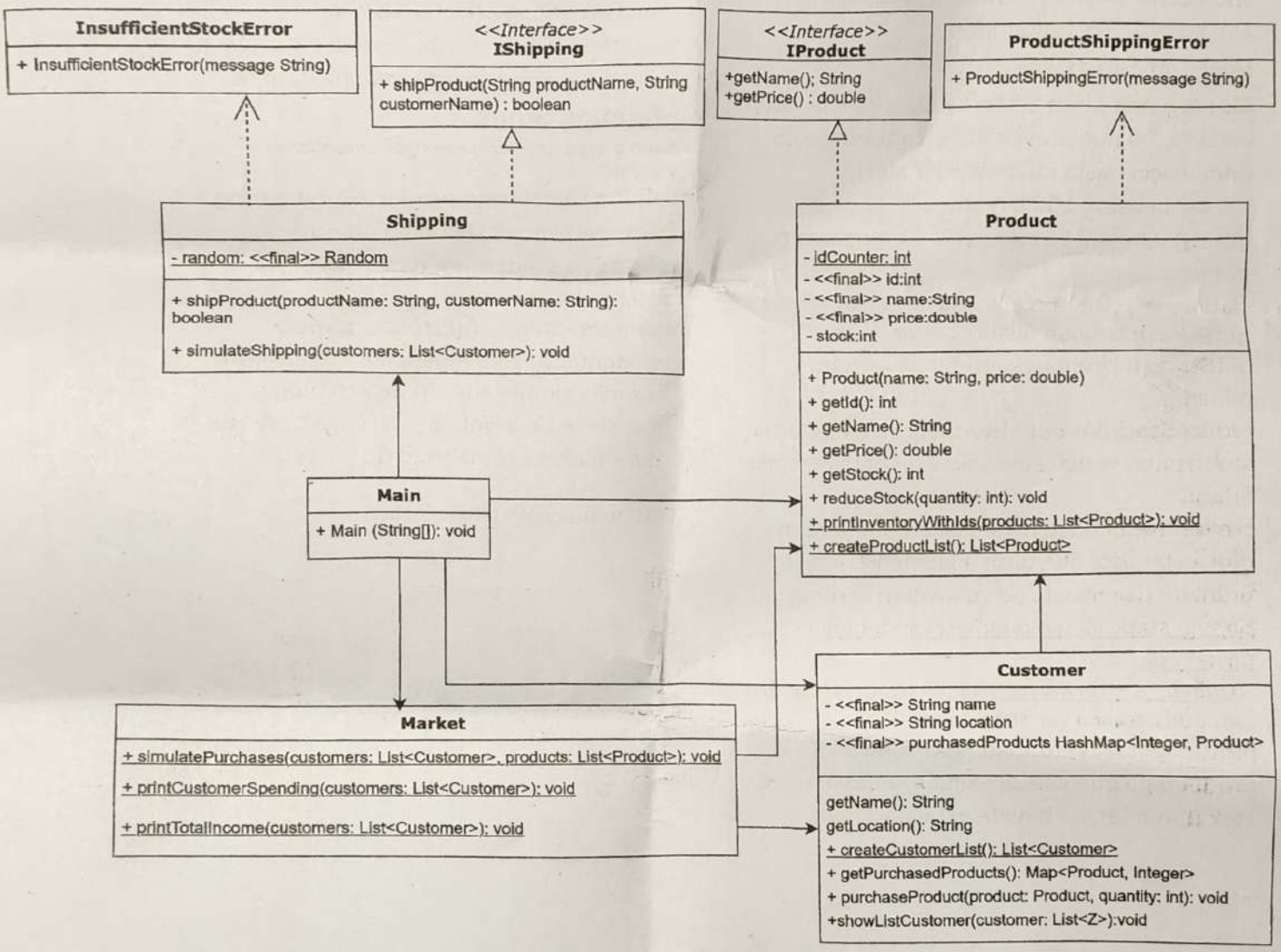
17.12.2024 NYP LAB 3 **Süre:** 80 dk

Bugünki ilgili kodda basit bir **Market Simülasyonu oluşturulacaktır.**

- Main sınıfında görüldüğü üzere öncelikle liste halinde tutulmuş müşteriler ve ürünler yazdırılır. Her ürünün kendine ait; kendi kendine artan static bir id'si, adı ve fiyatı vardır. Müşteri nesnesinde ise müşterinin adı ve lokasyonu tutulmaktadır.
- Her müşteri rastgele 1-3 arası ürün sipariş etmekte, Her ürün için rastgele bir miktar (25-50 arasında) ürün satın alınmaktadır. Eğer satın alınmak istenen ürünün stokları yetersizse, kalan miktar satılır ve stok yetersizliği mesajı gösterilir.
- Shipping Status başlığında Her müşteri için satın alınan ürünlerin sevkiyatı simüle edilir. Ürünlerin sevkiyatı %75 başarı oranıyla gerçekleştirilir. Eğer sevkiyat başarısız olursa, ilgili ürünün sevkiyatında hata mesajı gösterilir. Eğer tüm ürünlerin sevkiyatı başarılıysa, müşteriye "Tüm siparişler başarıyla gönderildi" mesajı gösterilir.
- Market Inventory başlığı altında Her müşterinin toplam harcamaları hesaplanır ve ekrana yazdırılır ve son olarak Total Income başlığı altında ise Marketin tüm müşterilerden elde ettiği toplam gelir hesaplanır ve ekrana yazdırılır.

Aşağıda kodlama için gerekli UML Diyagramı yer almaktadır.



İlgili çalışma içerisinde bulunan metotlara dair detaylar ve istenilenler diğer sayfada yer almaktadır.

Customer Sınıfı:

- **Constructer:** Belirli bir adla ve lokasyonla müşteriyi başlatır.
- **getName():** Müşterinin adını döner.
- **getLocation():** Müşterinin konumunu döner.
- **createCustomerList():** Varsayılan müşterilerden oluşan bir liste oluşturur. *(Müşteri listesi için ihtiyacınız olan bilgi sıkıştırılmış .xlsx dosyasında yer almaktadır.)*
- **getPurchasedProducts():** Müşterinin satın aldığı ürünlerin listesini döner.
- **purchaseProduct(Product product, int quantity):** Bir müşteriye belirtilen üründen belirli miktarda satın alma işlemi ekler.
- **showListCustomer(List<Z> customer):** Müşteri listesini görüntüler.

Product Sınıfı:

- **Constructor:** Ürün adı ve fiyatı ile yeni bir ürün başlatır, "id numarasını" bir sonrakine geçirir (bunun için başta idCounter'ı 1 alın.), Varsayılan stok 100 birimdir.
- **getId():** Ürünün benzersiz kimlik numarasını döner.
- **getName():** Ürünün adını döner.
- **getPrice():** Ürünün fiyatını döner.
- **getStock():** Ürünün mevcut stok miktarını döner.
- **reduceStock(int quantity):**Belirtilen miktarda stok azaltır. Yeterli stok yoksa bir hata mesajını fırlatır.
- **createProductList():** Varsayılan ürünlerden oluşan bir liste oluşturur. Eklemeniz gereken ürünlere dair sırayla ad ve fiyatları verilmiştir. Sırayla static ID tanımlaması için tabloya dikkat edin.
  (Ürün listesi için ihtiyacınız olan bilgi sıkıştırılmış .xlsx dosyasında yer almaktadır.)
- **printInventoryWithIds(List<Product> products):** Ürün listesini kimlik numaraları ve stok durumlarıyla birlikte ekrana yazdırır.

Market Sınıfı:

Satışları ve gelirleri simüle eder, müşterilerin harcamalarını hesaplar.

- **simulatePurchases(List<Customer> customers, List<Product> products):** Müşteriler için rastgele ürün satın alımları simüle eder. Müşteri rastgele 1 ila 3 arası ürün satın alabilir. Satın aldığı ürün adedi (25 ila 50 arasındadır.). try catch bloğu içerisinde metotlar aracılığı ile ürünlerin satışı simüle edilir ve stok yetersizliği durumunda hata mesajı fırlatır.
- **printCustomerSpending(List<Customer> customers):** Her müşterinin toplam harcamasını hesaplar ve ekrana yazdırır.
- **printTotalIncome(List<Customer> customers):** Tüm müşterilerden elde edilen toplam geliri hesaplar ve ekrana yazdırır.

Shipping Sınıfı:

Satın alınan ürünlerin sevkiyat durumlarını yönetir.

- **shipProduct(String productName, String customerName):** Belirtilen ürünü müşteriye göndermeyi dener. %25 başarısızlık ihtimali vardır.
- **simulateShipping(List<Customer> customers):** Tüm müşteriler için sevkiyat işlemini simüle eder. Try catch bloğu içerisinde Sevkiyat başarılı veya başarısız durumlarını ekrana yazdırır.

PS: quantity Miktar demektir.