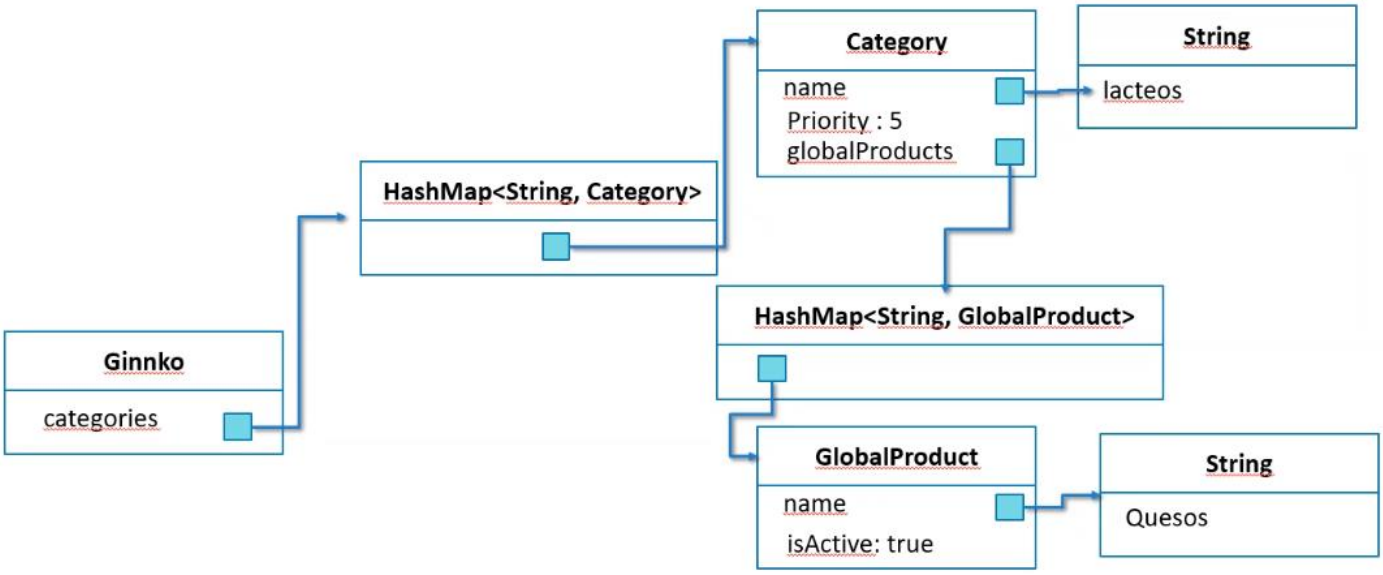
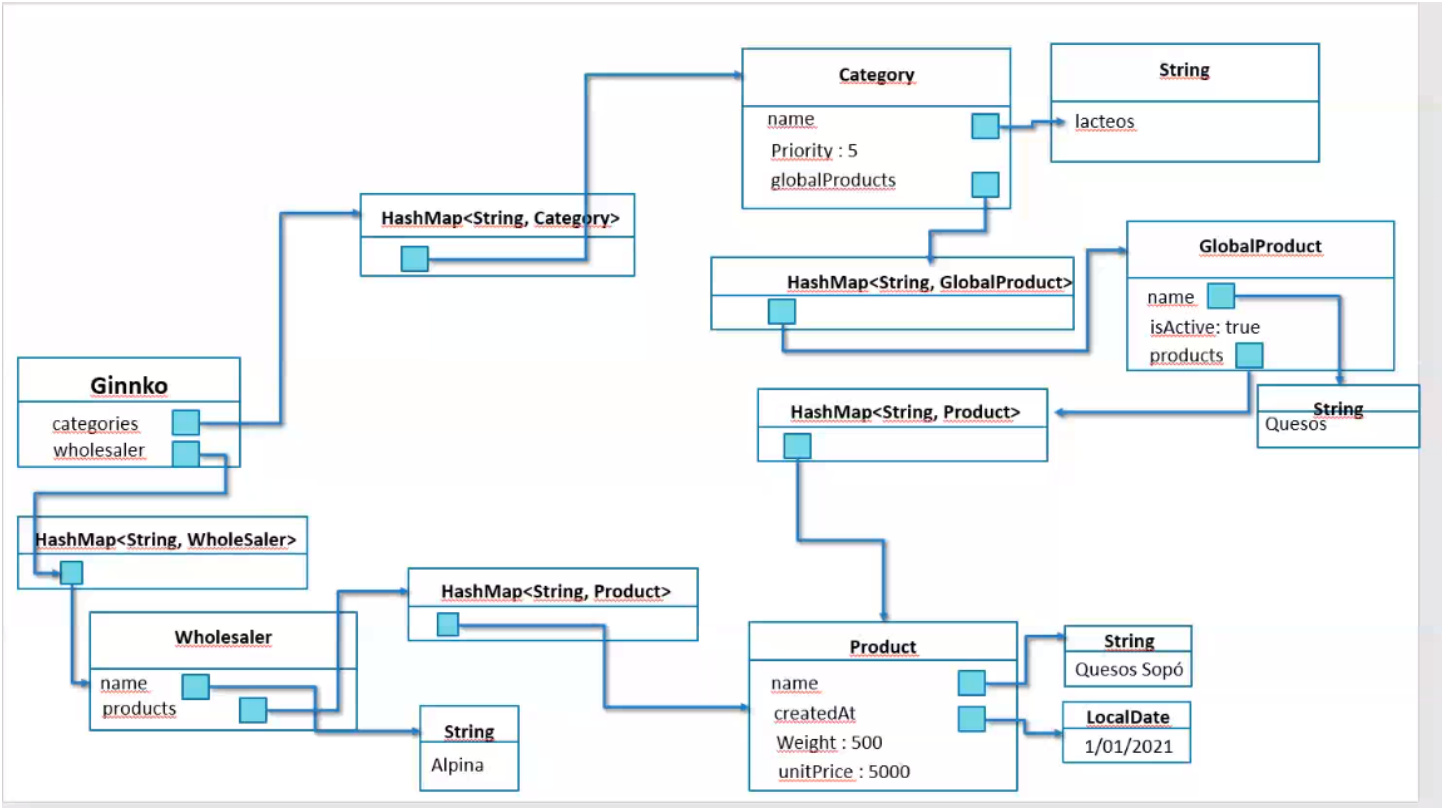
Solución Preparcial

Memoria

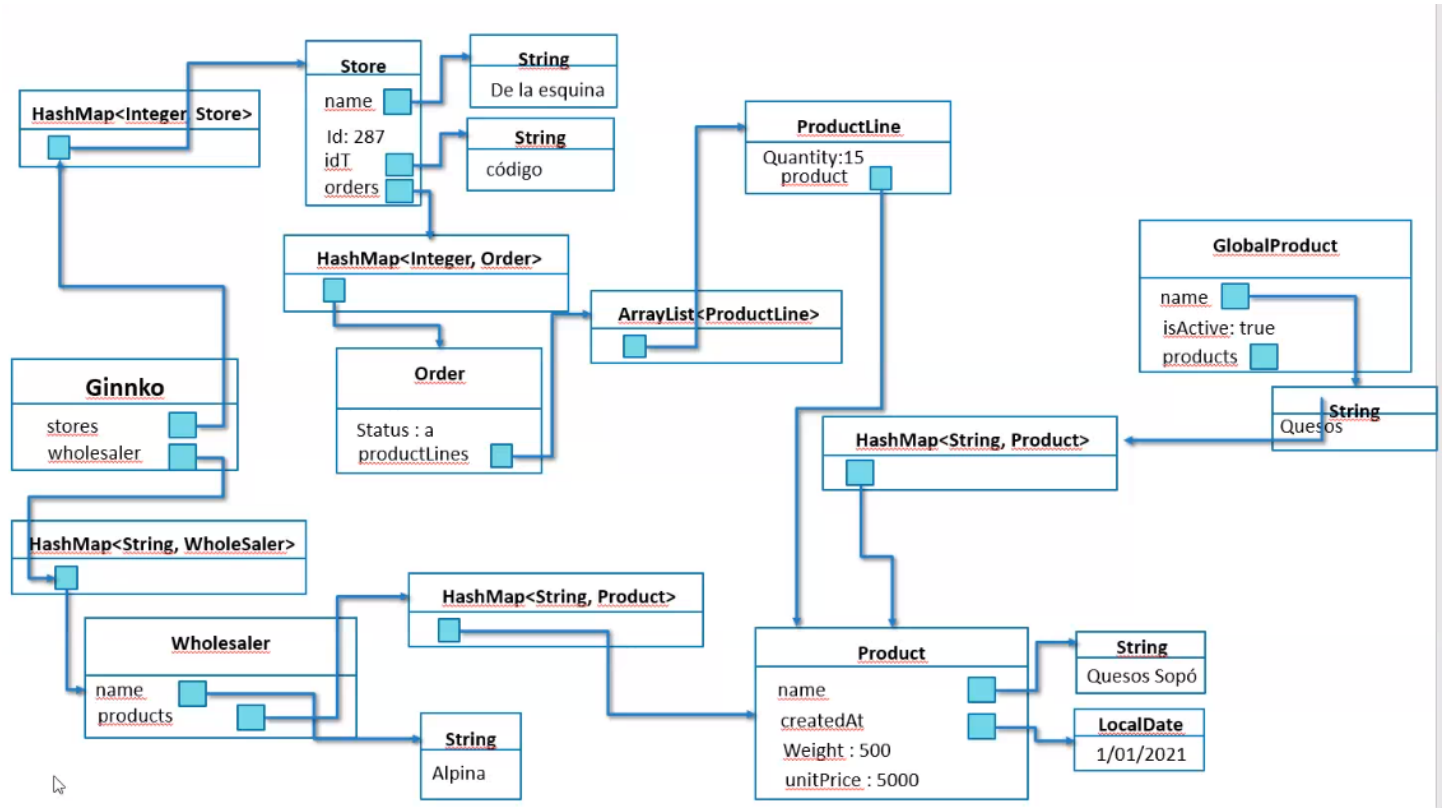
1.



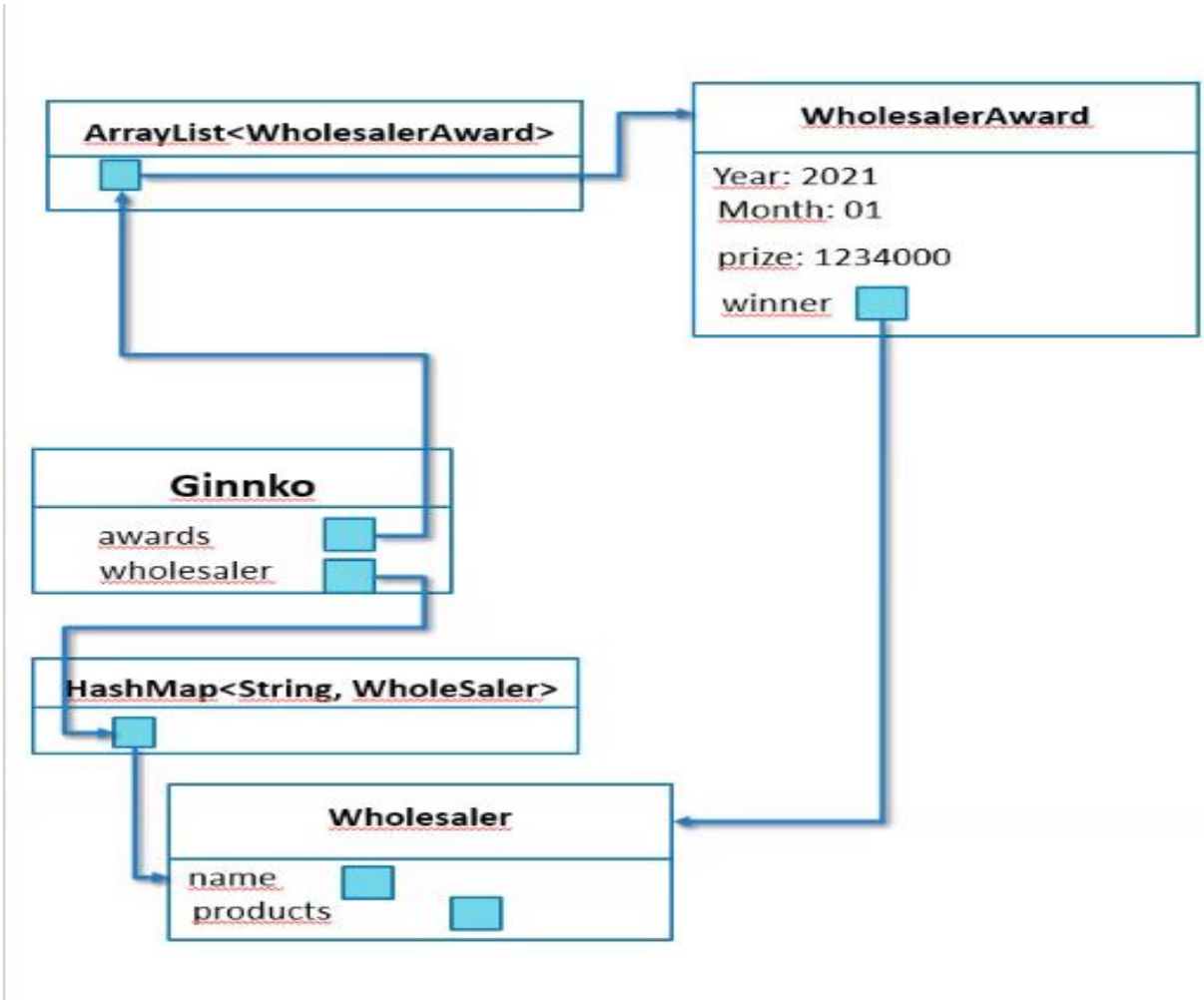
2.



3.

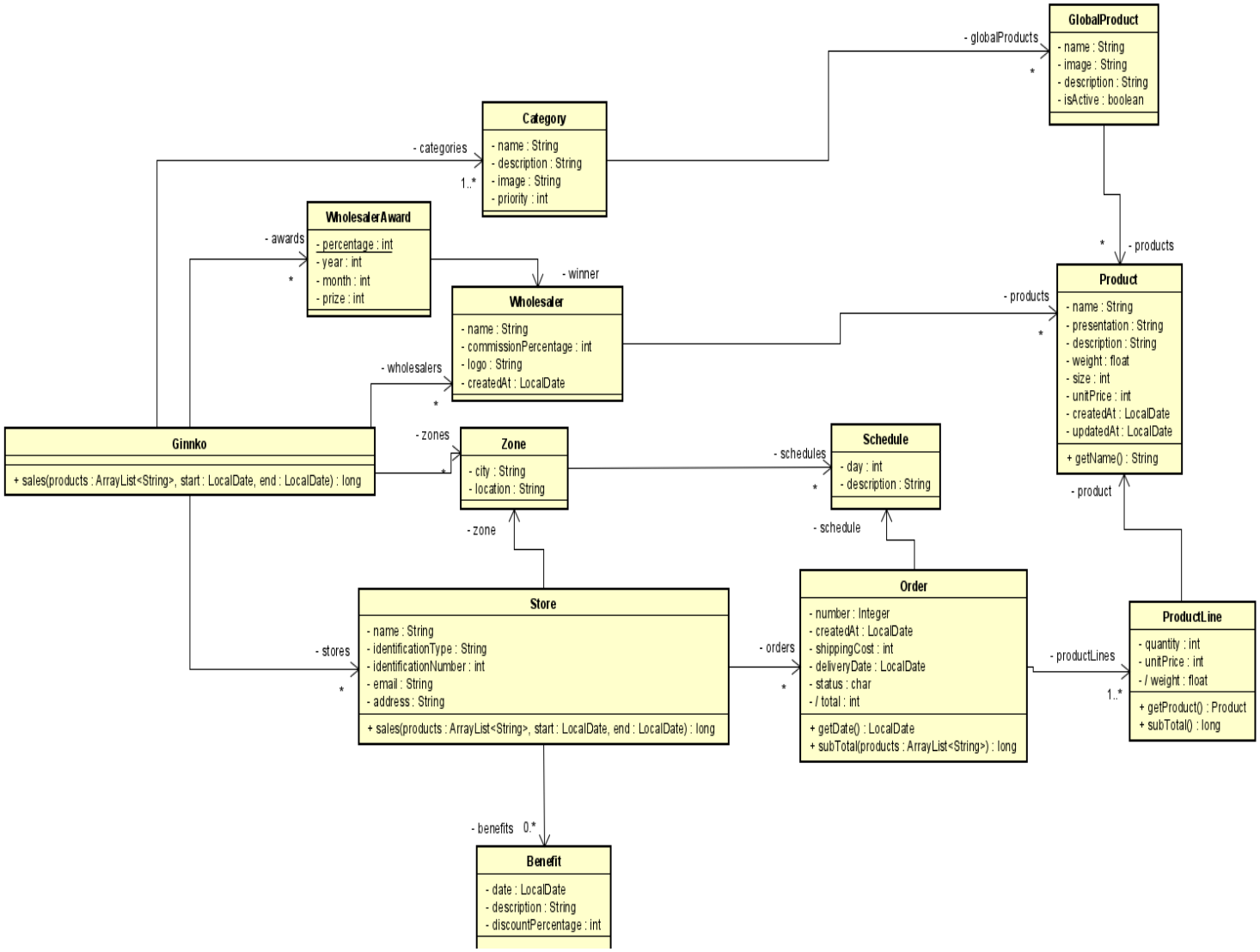

HashMap<Integer, Store>
Store
name
Id: 287
idT
orders
String
De la esquina
String
código
HashMap<Integer, Order>
Order
Status : a
productLines
ArrayList<ProductLine>
ProductLine
Quantity:15
product
GlobalProduct
name
isActive: true
products
String
Quesos
HashMap<String, Product>
Ginnko
stores
wholesaler
HashMap<String, WholeSaler>
Wholesaler
name
products
HashMap<String, Product>
String
Alpina
Product
name
createdAt
Weight : 500
unitPrice : 5000
String
Quesos Sopó
LocalDate
1/01/2021

4.


ArrayList<WholesalerAward>
WholesalerAward
Year: 2021
Month: 01
prize: 1234000
winner
Ginnko
awards
wholesaler
HashMap<String, WholeSaler>
Wholesaler
name
products

CODIGO

2.



3.

```java
import java.util.*;
import java.time.LocalDate;

/**
 * Representa la operación de la compañía Ginnko enfocada a la comercialización y distribución de productos
 * Invariantes:
 * - No existen categorías repetidas (categorías con el mismo nombre).
 * - No existen mayoristas repetidos (mayorías con el mismo nombre).
 * - No existen zonas repetidas (ubicaciones repetidas).
 * - No existen tiendas repetidas (número de identificación repetida)
 */
public class Ginnko {

    private HashMap<String, Category> categories;
    private ArrayList<WholesalerAward> awards;
    private HashMap<String, Wholesaler> wholesalers;
    private HashMap<String, Zone> zones;
    private HashMap<Integer, Store> stores;

}
```

4.

```java
import java.util.*;
import java.time.LocalDate;

/**
 * Representa la operación de la compañía Ginnko enfocada a la comercialización y distribución de productos
 * Invariantes:
 * - No existen categorías repetidas (categorías con el mismo nombre).
 * - No existen mayoristas repetidos (mayorias con el mismo nombre).
 * - No existen zonas repetidas (ubicaciones repetidas).
 * - No existen tiendas repetidas (número de identificación repetida)
 */
public class Ginnko {

    private HashMap<String, Category> categories;
    private ArrayList<WholesalerAward> awards;
    private HashMap<String, Wholesaler> wholesalers;
    private HashMap<String, Zone> zones;
    private HashMap<Integer, Store> stores;

    /**
     * Returns the total sales of a set of products in a time range
     * @param products - names of the products
     * @param start - start date
     * @param end - end date
     * @return total sales
     */
    public long sales(ArrayList<String> products, LocalDate start, LocalDate end){

        long sales = 0;

        for(Integer store: stores.keySet()){
            sales += stores.get(store).sales(products, start, end);
        }
        return sales;
    }
}
```

```java
import java.util.ArrayList;
import java.util.HashMap;
import java.time.LocalDate;

public class Store {
    private String name;
    private String identificationType;
    private int identificationNumber;
    private String email;
    private String address;
    private HashMap<Integer,Order> orders;
    private ArrayList<Benefit> benefits;
    private Zone zone;

    /**
     * Returns the total sales of a set of products in a time range
     * @param products - names of the products
     * @param start - start date
     * @param end - end date
     * @return total sales
     */
    public long sales(ArrayList<String> products, LocalDate start, LocalDate end){

        long sales = 0;
        LocalDate date;

        for(Integer order : orders.keySet()){
            date = orders.get(order).getDate();

            if(date.compareTo(start) >= 0 && date.compareTo(end) <= 0){
                sales += orders.get(order).subTotal(products);

            }
        }
        return sales;
    }
}
```

```java
import java.util.ArrayList;
import java.time.LocalDate;

public class Order {
    private Integer number;
    private LocalDate createdAt;
    private int shippingCost;
    private LocalDate deliveryDate;
    private char status;
    private Schedule schedule;
    private ArrayList<ProductLine> productLines;

    public LocalDate getDate(){
        return createdAt;
    }

    /**
     * Calcula el subtotal de los productos relacionados en cada #ProductLine
     * @param products lista de productos a validar
     * @return subtotal de la orden
     */
    public long subTotal(ArrayList<String> products){

        long subtotal = 0;
        Product product;
        String productName = "";

        for(ProductLine productLine : productLines){
            product = productLine.getProduct();
            productName = product.getName();

            if(products.contains(productName)){
                subtotal += productLine.subTotal();
            }
        }
        return subtotal;

    }
}
```

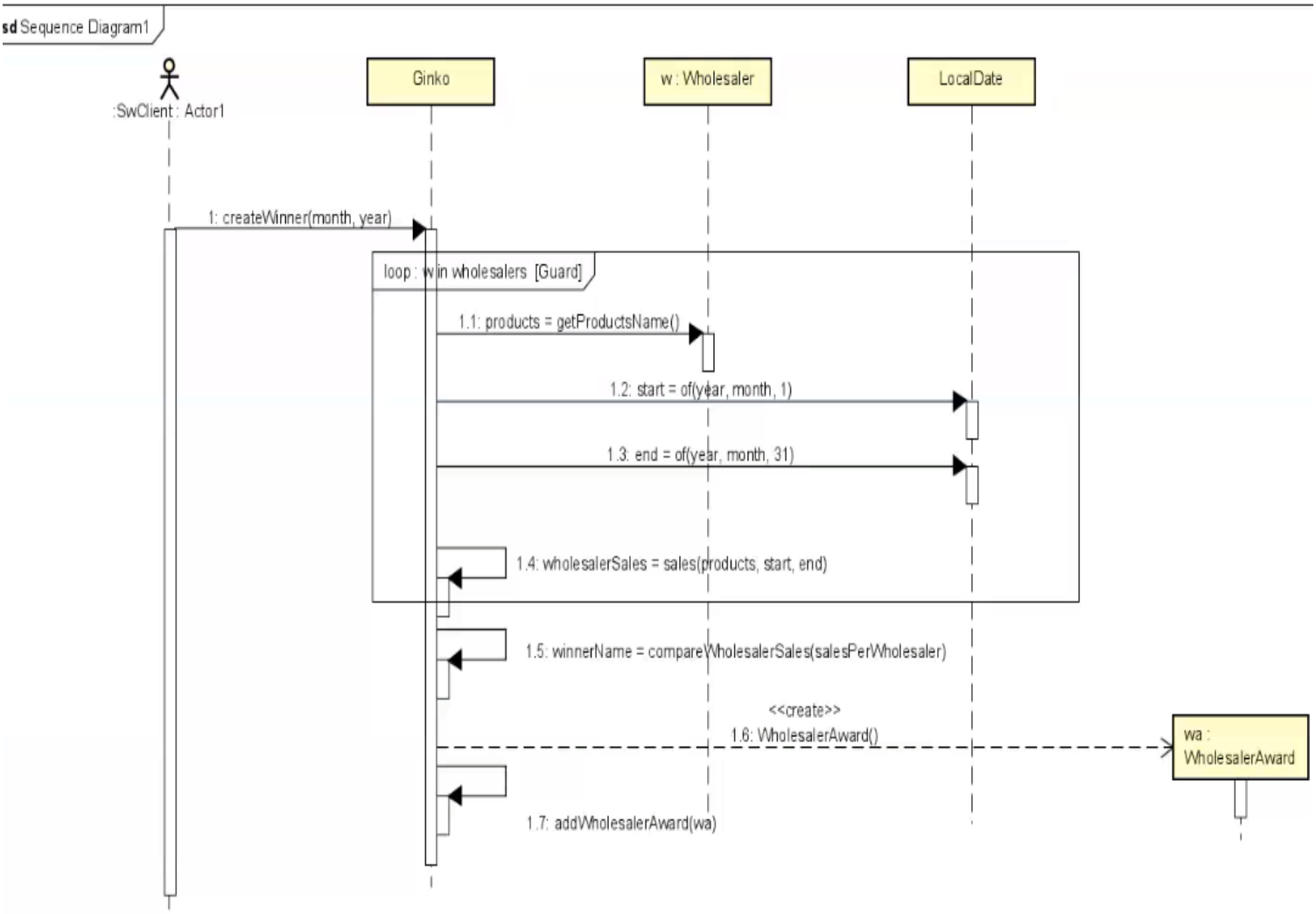
```java
public class ProductLine {
    private int quantity;
    private int unitPrice;
    private Product product;

    public Product getProduct(){
    return product;
}

    /**
     * Calcula el subtotal de la linea de producto
     * @return subtotal (cantidad * precio unitario)
     */
    public long subTotal(){
        return quantity * unitPrice;
    }
}
```

MEMORIA

Diseñe un método de la clase Ginnko (especificación y diagrama de secuencia decorado) que cree el premio de distribuidor para un mes y un año dado. Si hay empate, no hay premio. No olvide considerar las precondiciones.



Conceptos

2. ¿Cuáles son los dos principales preguntas de diseño de un método en OO? ¿Donde se responden?

1.Que objetos tengo—se resuelve con las clases.

2.Que debo hacer o cuál es su objetivo o que objetos interactúan—se responde con el diagrama de secuencia.