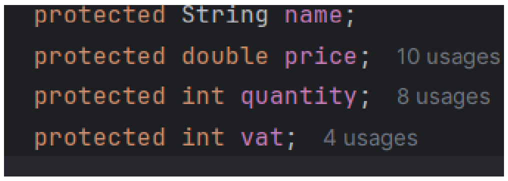
Documentație MIP

Vaida Raluca-Maria
10LF333

În acest document voi enumera elementele folosite la fiecare laborator și modul în care le-am aplicat în realizarea acestei aplicații.

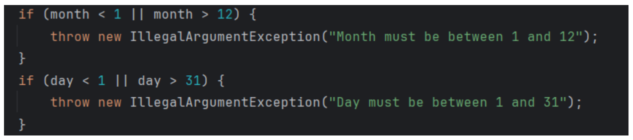
Lab 1 *Introducere în Java (output, tipuri de valori, funcții)*:
In urma acestui laborator am stiut sa declar diferite tipuri de variabile:



```java
protected String name;
protected double price;  10 usages
protected int quantity;  8 usages
protected int vat;  4 usages
```

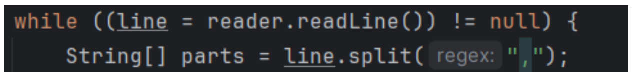
Figure 1: Variabile

Lab 2 *Introducere în Java (input, for, while, switch, if)*:
In urma acestui laborator am putut implementa diverse structuri de control:



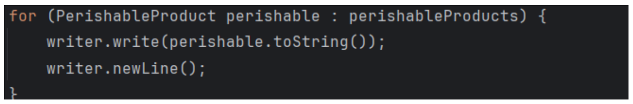
```java
if (month < 1 || month > 12) {
    throw new IllegalArgumentException("Month must be between 1 and 12");
}
if (day < 1 || day > 31) {
    throw new IllegalArgumentException("Day must be between 1 and 31");
}
```

Figure 2: If



```java
while ((line = reader.readLine()) != null) {
    String[] parts = line.split( regex: ",");
```

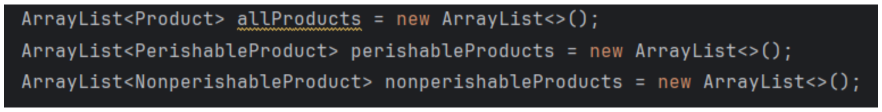
Figure 3: While



```java
for (PerishableProduct perishable : perishableProducts) {
    writer.write(perishable.toString());
    writer.newLine();
}
```

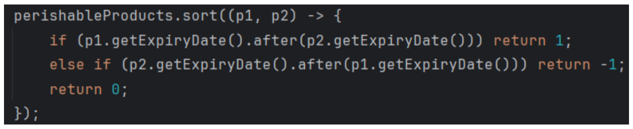
Figure 4: For

Lab 3 *Colecții Java (Array, List, Map)*:
In urma acestui laborator am utilizat ArrayList cu metode specifice:



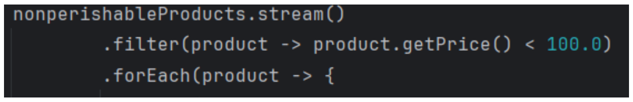
```java
ArrayList<Product> allProducts = new ArrayList<>();
ArrayList<PerishableProduct> perishableProducts = new ArrayList<>();
ArrayList<NonperishableProduct> nonperishableProducts = new ArrayList<>();
```

Figure 5: ArrayList



```java
perishableProducts.sort((p1, p2) -> {
    if (p1.getExpiryDate().after(p2.getExpiryDate())) return 1;
    else if (p2.getExpiryDate().after(p1.getExpiryDate())) return -1;
    return 0;
});
```

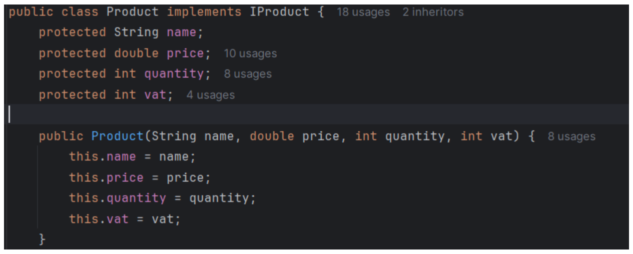
Figure 6: Sort



```java
nonperishableProducts.stream()
        .filter(product -> product.getPrice() < 100.0)
        .forEach(product -> {
```

Figure 7: Filter

Lab 4 *Clase Java (clasă cu atribute și metode)* :
In urma acestui laborator am implementat clase precum:



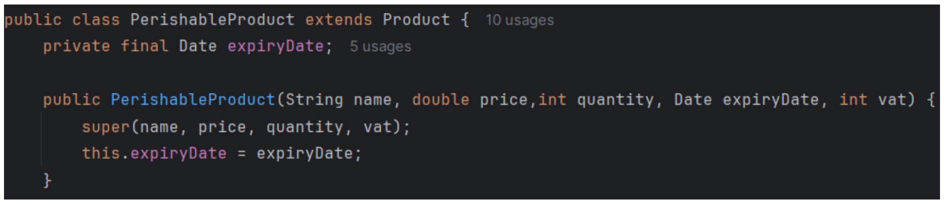
```java
public class Product implements IProduct {  18 usages   2 inheritors
    protected String name;
    protected double price;  10 usages
    protected int quantity;  8 usages
    protected int vat;  4 usages

    public Product(String name, double price, int quantity, int vat) {  8 usages
        this.name = name;
        this.price = price;
        this.quantity = quantity;
        this.vat = vat;
    }
```

Figure 8: Clasa Product

Lab 5 *Moștenire în Java, clase abstracte* :

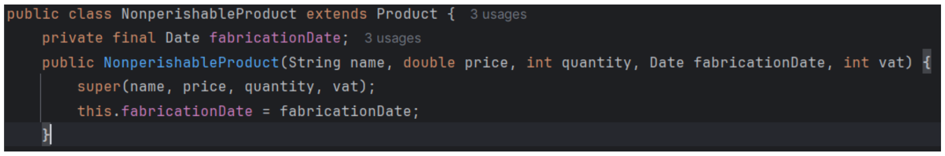
In urma acestui laborator am implementat conceptul de mostenire, precum si clasa abstracta AbstractDate:

- Clasa PerishableProduct mosteneste (extends) Product;
- Clasa NonperishableProduct mosteneste (extends) Product;
- Clasa Date mosteneste (extends) clasa abstracta AbstractDate;



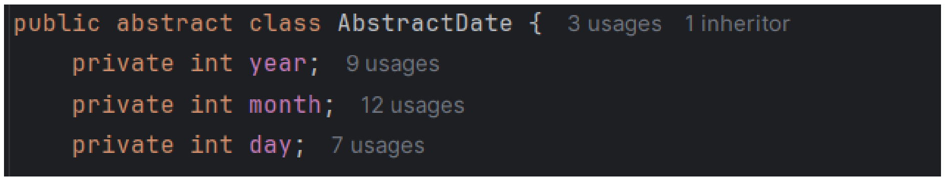
```java
public class PerishableProduct extends Product {  10 usages
    private final Date expiryDate;  5 usages

    public PerishableProduct(String name, double price,int quantity, Date expiryDate, int vat) {
        super(name, price, quantity, vat);
        this.expiryDate = expiryDate;
    }
```

Figure 9: Clasa PerishableProduct



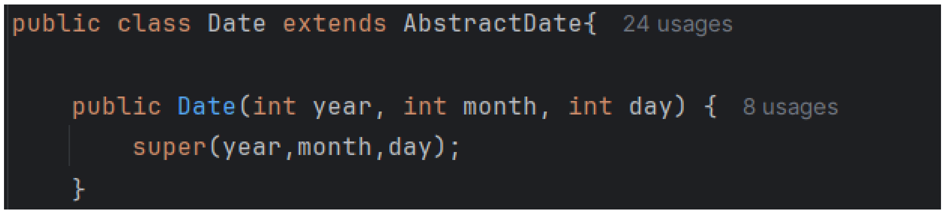
```java
public class NonperishableProduct extends Product {  3 usages
    private final Date fabricationDate;  3 usages
    public NonperishableProduct(String name, double price, int quantity, Date fabricationDate, int vat) {
        super(name, price, quantity, vat);
        this.fabricationDate = fabricationDate;
    }
```

Figure 10: Clasa NonperishableProduct



```java
public abstract class AbstractDate {  3 usages  1 inheritor
    private int year;  9 usages
    private int month;  12 usages
    private int day;  7 usages
```

Figure 11: Clasa AbstractDate

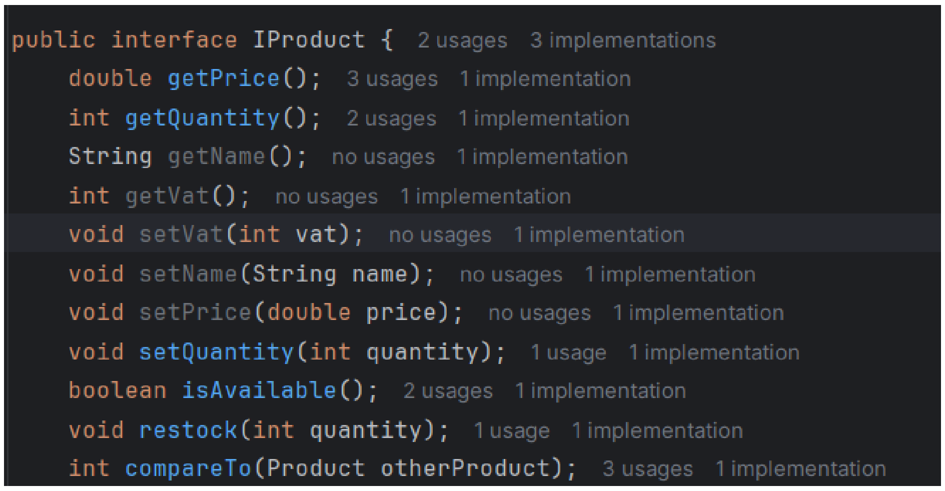


```java
public class Date extends AbstractDate{  24 usages

    public Date(int year, int month, int day) {  8 usages
        super(year,month,day);
    }
```

Figure 12: Clasa Date

Lab 6 *Interfețe în Java*:

In urma acestui laborator am implementat conceptul de interfata IProduct care implementeaza (implements) Product:

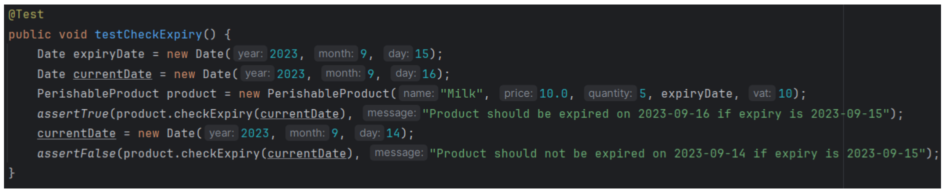


```java
public interface IProduct {  2 usages   3 implementations
    double getPrice();  3 usages   1 implementation
    int getQuantity();  2 usages   1 implementation
    String getName();  no usages   1 implementation
    int getVat();  no usages   1 implementation
    void setVat(int vat);  no usages   1 implementation
    void setName(String name);  no usages   1 implementation
    void setPrice(double price);  no usages   1 implementation
    void setQuantity(int quantity);  1 usage   1 implementation
    boolean isAvailable();  2 usages   1 implementation
    void restock(int quantity);  1 usage   1 implementation
    int compareTo(Product otherProduct);  3 usages   1 implementation
```

Figure 13: Interfata IProduct

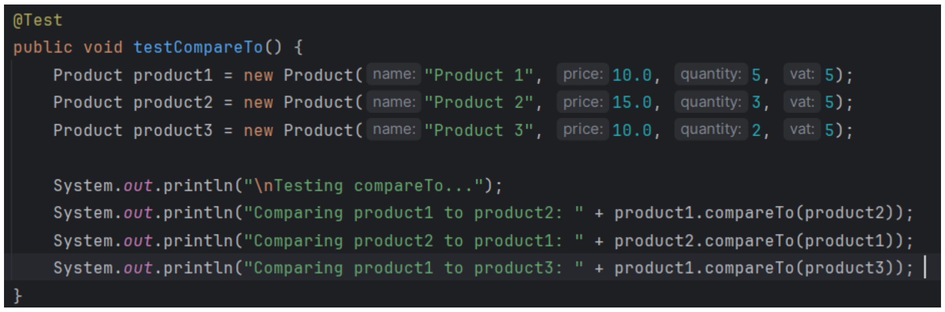
Lab 7 *Unit Testing*:

In urma acestui laborator am testat diverse metode deja implementate prin intermediul claselor PerishableProductTest si ProductTest:



```java
@Test
public void testCheckExpiry() {
    Date expiryDate = new Date( year: 2023,  month: 9,  day: 15);
    Date currentDate = new Date( year: 2023,  month: 9,  day: 16);
    PerishableProduct product = new PerishableProduct( name: "Milk",  price: 10.0,  quantity: 5, expiryDate,  vat: 10);
    assertTrue(product.checkExpiry(currentDate),  message: "Product should be expired on 2023-09-16 if expiry is 2023-09-15");
    currentDate = new Date( year: 2023,  month: 9,  day: 14);
    assertFalse(product.checkExpiry(currentDate),  message: "Product should not be expired on 2023-09-14 if expiry is 2023-09-15");
}
```

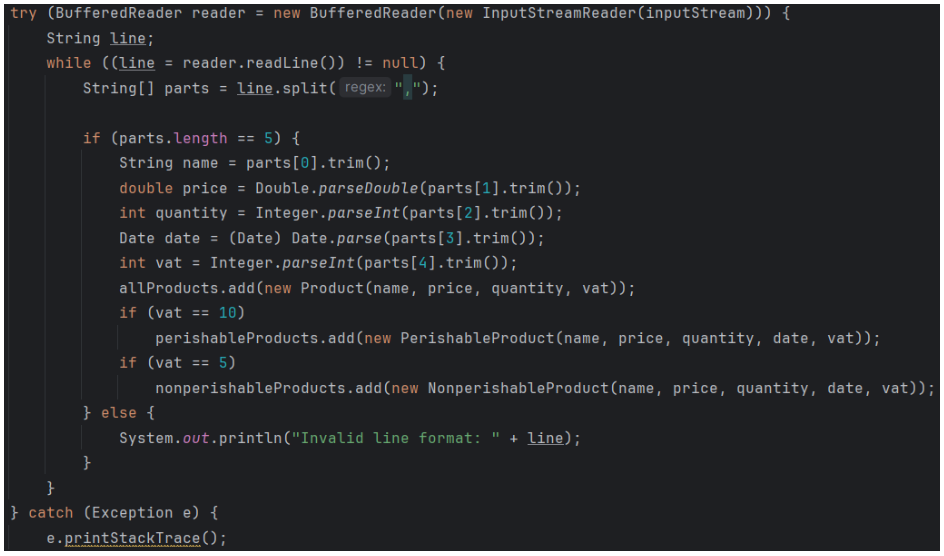
Figure 14: Unit Test-Exemplu 1



```java
@Test
public void testCompareTo() {
    Product product1 = new Product( name: "Product 1",  price: 10.0,  quantity: 5,  vat: 5);
    Product product2 = new Product( name: "Product 2",  price: 15.0,  quantity: 3,  vat: 5);
    Product product3 = new Product( name: "Product 3",  price: 10.0,  quantity: 2,  vat: 5);

    System.out.println("\nTesting compareTo...");
    System.out.println("Comparing product1 to product2: " + product1.compareTo(product2));
    System.out.println("Comparing product2 to product1: " + product2.compareTo(product1));
    System.out.println("Comparing product1 to product3: " + product1.compareTo(product3));
}
```

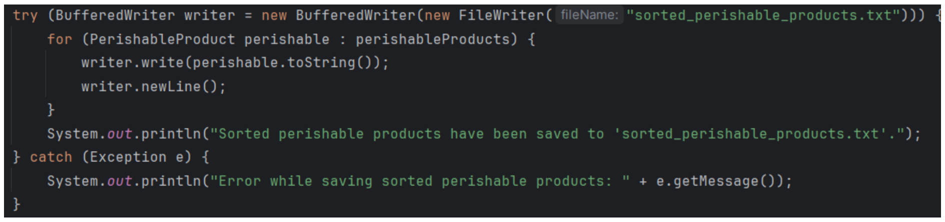
Figure 15: Unit Test-Exemplu 2

Lab 8 *Persistenta Datelor*:
In urma acestui laborator am modificat citirea si scrierea din/in format .txt :



```java
try (BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream))) {
    String line;
    while ((line = reader.readLine()) != null) {
        String[] parts = line.split( regex: ",");

        if (parts.length == 5) {
            String name = parts[0].trim();
            double price = Double.parseDouble(parts[1].trim());
            int quantity = Integer.parseInt(parts[2].trim());
            Date date = (Date) Date.parse(parts[3].trim());
            int vat = Integer.parseInt(parts[4].trim());
            allProducts.add(new Product(name, price, quantity, vat));
            if (vat == 10)
                perishableProducts.add(new PerishableProduct(name, price, quantity, date, vat));
            if (vat == 5)
                nonperishableProducts.add(new NonperishableProduct(name, price, quantity, date, vat));
        } else {
            System.out.println("Invalid line format: " + line);
        }
    }
} catch (Exception e) {
    e.printStackTrace();
```

Figure 16: Citire din fisier



```java
try (BufferedWriter writer = new BufferedWriter(new FileWriter( fileName: "sorted_perishable_products.txt"))) {
    for (PerishableProduct perishable : perishableProducts) {
        writer.write(perishable.toString());
        writer.newLine();
    }
    System.out.println("Sorted perishable products have been saved to 'sorted_perishable_products.txt'.");
} catch (Exception e) {
    System.out.println("Error while saving sorted perishable products: " + e.getMessage());
}
```

Figure 17: Afisare in fisier

Lab 9 *Diagrama UML*:
In urma acestui laborator am conceput diagrama UML pentru aplicatia mea:

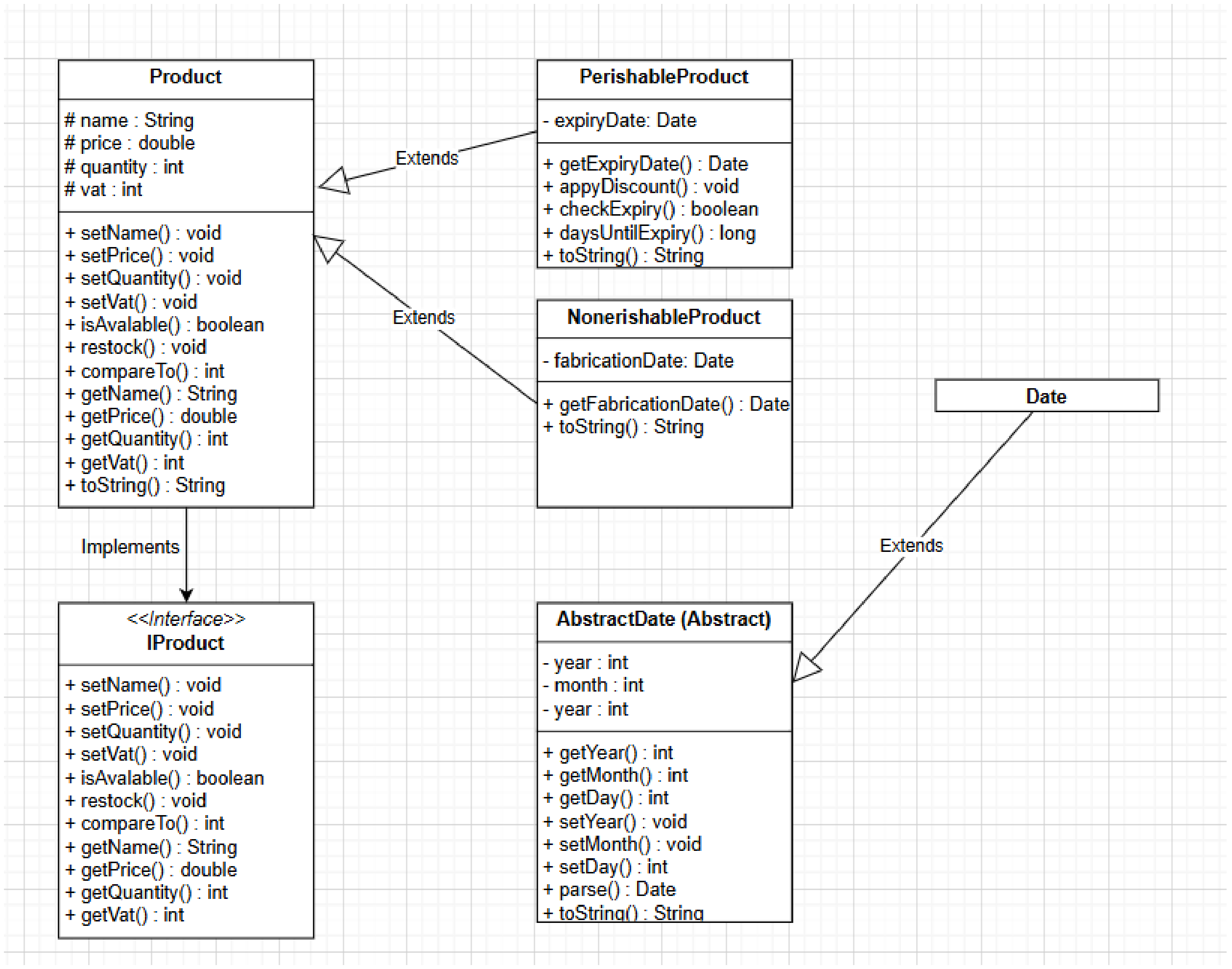


Figure 18: Diagrama UML