

# Assignment 2

Barletta Alessio (5686474) Sacco Daniele (5616921) Scaffai Daniele (5658260) Vaccarecci Lorenzo Livio (5462843)

## 1 Sonarlint

### 1.1 Cart.java

Porzione di codice:

```
1 public void addProduct(Product product, int quantity) {
2     if (quantity > 0)
3         products.put(product, products.getDefault(product, 0) + quantity);
4     else
5         products.put(product, products.getDefault(product, 0));
6
7 }
```

Tipologia: Maintainability

Correzione:

```
1 public void addProduct(Product product, int quantity) {
2     if (quantity > 0)
3         products.put(product, products.getDefault(product, 0) + quantity);
4     else
5         products.put(product, products.getDefault(product, 0));
6 }
```

---

Porzione di codice:

```
1 public void updateProductQuantity(Product product, int quantity) {
2     if (products.containsKey(product)==true)
3         if (quantity <= 0)
4             removeProduct(product);
5     else
6         products.put(product, quantity);
7 }
```

Tipologia: Maintainability

Correzione:

```
1 public void updateProductQuantity(Product product, int quantity) {
2     if (products.containsKey(product)) {
3         if (quantity <= 0)
4             removeProduct(product);
5     }
6     else {
7         products.put(product, quantity);
8     }
9 }
```

---

Porzione di codice:

```
1 public Map<Product, Integer> getProducts() {  
2     return new HashMap<Product, Integer>(products);  
3 }
```

Tipologia: Maintainability

Correzione:

```
1 public Map<Product, Integer> getProducts() {  
2     return new HashMap<>(products);  
3 }
```

---

Porzione di codice:

```
1 public boolean calc(Cart cart1, Cart cart2) {  
2     if(this.calculateTotal() >= cart1.calculateTotal() && this.calculateTotal() >=  
3         cart2.calculateTotal())  
4         return true;  
5     else  
6         return false;  
7 }
```

Tipologia: Maintainability

Correzione:

```
1 public boolean calc(Cart cart1, Cart cart2) {  
2     return this.calculateTotal() >= cart1.calculateTotal() && this.calculateTotal()  
3         >= cart2.calculateTotal();  
4 }
```

---

Porzione di codice:

```
1 public void calcHigher(Cart cart1, Cart cart2) {  
2     calc(cart2, cart1);  
3 }
```

Tipologia: Maintainability

Correzione:

```
1 public void calcHigher(Cart cart1, Cart cart2) {  
2     calc(cart1, cart2);  
3 }
```

## 1.2 Product.java

Porzione di codice:

```
1 public void setDescription(String description) {  
2 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setDescription(String description) {  
2     this.description = description;  
3 }
```

---

Porzione di codice:

```
1 public void setUnitPrice(double unitPrice) {  
2 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setUnitPrice(double unitPrice) {  
2     this.unitPrice = unitPrice;  
3 }
```

---

Porzione di codice:

```
1 public void setUnitPrice(double unitPrice) {  
2 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setUnitPrice(double unitPrice) {  
2     this.unitPrice = unitPrice;  
3 }
```

---

Porzione di codice:

```
1 public void setCategory(int category) {  
2 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setCategory(int category) {  
2     this.category = category;  
3 }
```

---

Porzione di codice:

```
1 public void setBrand(String brand) {  
2 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setBrand(String brand) {  
2     this.brand = brand;  
3 }
```

---

Porzione di codice:

```
1 public void setStockQuantity(int stockQuantity) {  
2 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setStockQuantity(int stockQuantity) {
2     this.stockQuantity = stockQuantity;
3 }
```

---

Porzione di codice:

```
1 public boolean IsAvailable() {
2     return isAvailable;
3 }
```

Tipologia: Maintainability

Correzione:

```
1 public boolean isAvailable() {
2     return isAvailable;
3 }
```

---

Porzione di codice:

```
1 public void setIsAvailable(boolean isAvailable) {
2 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setIsAvailable(boolean isAvailable) {
2     this.isAvailable = isAvailable;
3 }
```

---

Porzione di codice:

```
1 public void setFullDetailedProduct(String id, String name, String description,
2     double unitPrice, int category, String brand, int stockQuantity, boolean
3     isAvailable) {
4     this.id = id;
5     this.name = name;
6     this.description = description;
7     this.unitPrice = unitPrice;
8     this.category = category;
9     this.brand = brand;
10    this.stockQuantity = stockQuantity;
11    this.isAvailable = isAvailable;
12 }
```

Tipologia: Maintainability

Correzione:

```
1 public void setFullDetailedProduct(Product product) {
2     this.id = product.getId();
3     this.name = product.getName();
4     this.description = product.getDescription();
5     this.unitPrice = product.getUnitPrice();
6     this.category = product.getCategory();
7     this.brand = product.getBrand();
8     this.stockQuantity = product.getStockQuantity();
9     this.isAvailable = product.isAvailable();
10 }
```

### 1.3 User.java

Porzione di codice:

```
1 private List<String> titles = new ArrayList<String>();
```

Tipologia: Maintainability

Correzione:

```
1 private List<String> titles = new ArrayList<>();
```

---

Porzione di codice:

```
1 public boolean isActive(){
2     if(accountActive)
3         return true;
4     return false;
5 }
```

Tipologia: Maintainability

Correzione: metodo cancellato

---

Porzione di codice:

```
1 public boolean deactivateAccount(String id) {
2     if (accountActive && this.userID == id) {
3         accountActive = false;
4         return true;
5     }
6     return false;
7 }
```

Tipologia: Reliability

Correzione:

```
1 public boolean deactivateAccount(String id) {
2     if (accountActive && Objects.equals(this.userID, id)) {
3         accountActive = false;
4         return true;
5     }
6     return false;
7 }
```

---

Porzione di codice:

```
1 public boolean isEqual(User u){
2     return u.userID == this.userID;
3 }
```

Tipologia: Reliability

Correzione:

```
1 public boolean isEqual(User u){
2     return Objects.equals(u.userID, this.userID);
3 }
```

---

Porzione di codice:

```

1 public void printUserInfo() {
2     System.out.println("User Info: " + firstname + " " + lastname + " (Username: " +
        username + ")");
3 }

```

Tipologia: Maintainability

Correzione:

```

1 import java.util.logging.Level;
2 import java.util.logging.Logger;
3
4 private static final Logger LOGGER = Logger.getLogger(User.class.getName());
5 public void printUserInfo() {
6     LOGGER.log(Level.INFO, "User Info: {0} {1} (Username: {2})", new Object[]{
        firstname, lastname, username});
7 }

```

---

Porzione di codice:

```

1 public void linkCart(Cart cart) throws Exception{
2     if(cart == null)
3         throw new Exception();
4     this.cart = cart;
5 }

```

Tipologia: Maintainability

Correzione:

```

1 public void linkCart(Cart cart) {
2     if(cart == null)
3         throw new IllegalArgumentException("Cart cannot be null");
4     this.cart = cart;
5 }

```

---

Porzione di codice:

```

1 public String printAllRoles(){
2     return roles.toString();
3 }

```

Tipologia: Reliability

Correzione:

```

1 public String printAllRoles(){
2     return Arrays.toString(roles);
3 }

```

---

Porzione di codice:

```

1 public void PrintEveryRole(){
2     for (int i = roles.length; i > 0; i++){
3         System.out.println(roles[i]);
4     }
5 }

```

Tipologia: Maintainability

Correzione:

```

1 public void printEveryRole(){
2     for (int i = roles.length; i > 0; i++){
3         System.out.println(roles[i]);
4     }
5 }

```

Porzione di codice:

```

1 public void printEveryRole(){
2     for (int i = roles.length; i > 0; i++){
3         System.out.println(roles[i]);
4     }
5 }

```

Tipologia: Reliability

Correzione:

```

1 public void printEveryRole(){
2     for (int i = 0; i < roles.length; i++){
3         System.out.println(roles[i]);
4     }
5 }

```

Porzione di codice:

```

1 public void PrintEveryRole(){
2     for (int i = roles.length; i > 0; i++){
3         System.out.println(roles[i]);
4     }
5 }

```

Tipologia: Maintainability

Correzione:

```

1 public void printEveryRole(){
2     for (int i = 0; i < roles.length ; i++){
3         LOGGER.log(Level.INFO, roles[i]);
4     }
5 }

```

## 1.4 UserManager.java

Porzione di codice:

```

1 public final static String basicUserID = "User00-";

```

Tipologia: Maintainability

Correzione:

```

1 public final static String BASICUSERID = "User00-";

```

Porzione di codice:

```

1 public final static String BASICUSERID = "User00-";

```

Tipologia: Maintainability

Correzione:

```
1 public static final String BASICUSERID = "User00-";
```

---

Porzione di codice:

```
1 public final static List<User> users = new ArrayList<User>();
```

Tipologia: Maintainability

Correzione:

```
1 public static final List<User> users = new ArrayList<User>();
```

---

Porzione di codice:

```
1 public static final List<User> users = new ArrayList<User>();
```

Tipologia: Maintainability

Correzione:

```
1 protected static final List<User> users = new ArrayList<User>();
```

---

Porzione di codice:

```
1 protected static final List<User> users = new ArrayList<User>();
```

Tipologia: Maintainability

Correzione:

```
1 protected static final List<User> users = new ArrayList<>();
```

---

Porzione di codice:

```
1 public boolean findUserFromDB(String userID) throws SQLException {
2     try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/
mydatabase", BASICUSERID+userID, "password");) {
3         String query = "select firstname, lastname " + "from USERS where username="+
(BASICUSERID+userID);
4         PreparedStatement stmt = conn.prepareStatement(query);
5         ResultSet rs = stmt.executeQuery();
6         while (rs.next())
7             if(rs != null)
8                 return true;
9         return false;
10
11     } catch (SQLException e) {
12         return false;
13     }
14 }
```

Tipologia: Security

Correzione: Creazione di un file .env

```
1 public boolean findUserFromDB(String userID) throws SQLException {
2     try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/
mydatabase", BASICUSERID+userID, System.getenv("DB_PASSWORD"));) {
3         String query = "select firstname, lastname " + "from USERS where username="+
(BASICUSERID+userID);
4         PreparedStatement stmt = conn.prepareStatement(query);
5         ResultSet rs = stmt.executeQuery();
```



```

6         while (rs.next())
7             if(rs != null)
8                 return true;
9         return false;
10
11     } catch (SQLException e) {
12         return false;
13     }
14 }

```

Porzione di codice:

```

1 void removeEmptyTitlesFromUser(User user) {
2     List<String> titles = user.getTitles();
3     for (int i = 0; i < titles.size(); i++) {
4         if (titles.get(i).isEmpty()) {
5             titles.remove(i);
6         }
7     }
8 }

```

Tipologia: Maintainability

Correzione:

```

1 void removeEmptyTitlesFromUser(User user) {
2     List<String> titles = user.getTitles();
3     titles.removeIf(String::isEmpty);
4 }

```

Porzione di codice:

```

1 void addCartToUser(User user, Cart cart) throws Exception{
2     try {
3         user.linkCart(cart);
4     } catch (Exception e) {
5         throw e;
6     }
7 }

```

Tipologia: Maintainability

Correzione:

```

1 void addCartToUser(User user, Cart cart) {
2     try {
3         user.linkCart(cart);
4     } catch (Exception e) {
5         throw new IllegalArgumentException("Error linking cart to user", e);
6     }
7 }

```

## 1.5 Porzioni di codice problematiche

### 1.5.1 Cart.java

La funzione `calcHigher` è inutile in quanto esegue la stessa operazione della funzione `calc` e non ritorna alcun valore, inoltre il nome `calc` non è significativo quindi si potrebbe cambiare in `isTotalHighest`.

### 1.5.2 Product.java

L'id di un prodotto non dovrebbe essere modificabile, quindi il metodo `setId` non dovrebbe esistere, il campo `id` dovrebbe essere `final` e nel costruttore dovremmo mettere almeno per quel campo `requireNonNull`. Il metodo `setFullDetailedProduct` lo si potrebbe togliere in quanto si hanno già i metodi `set` per ogni campo.

### 1.5.3 User.java

Il nome del metodo `printAllRoles` lo si potrebbe modificare in `getRolesAsString`.

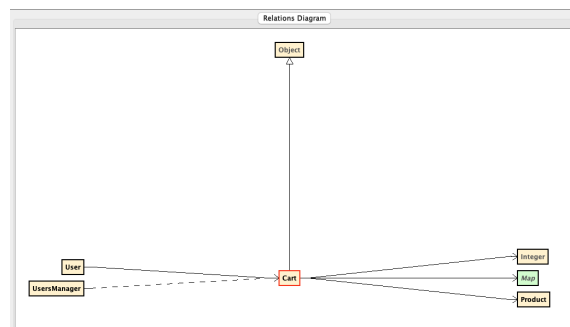
### 1.5.4 UserManager.java

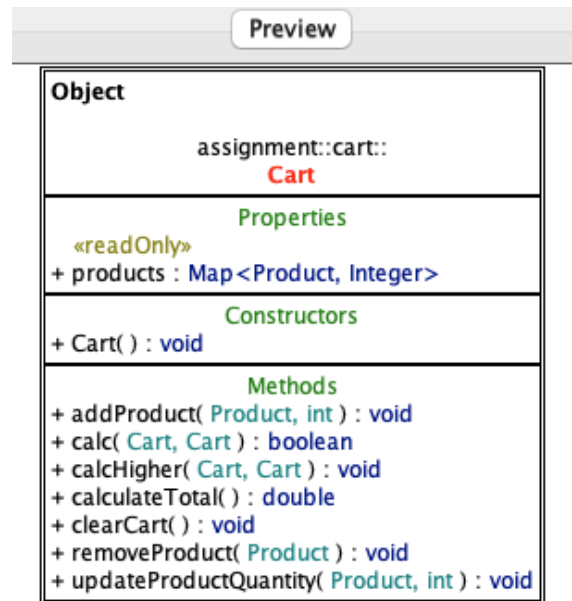
Nel metodo `findUserFromDB` non gestiamo correttamente i parametri `BASICUSERID` e `userID` quando concatenati alla query lasciando la possibilità di SQL injection, una possibile soluzione potrebbe essere usare la funzione `setString` del `PreparedStatement` cambiando la query in `"select firstname, lastname from USERS where username=?"`.

## 2 Class Visualizer

### 2.1 Cart.java

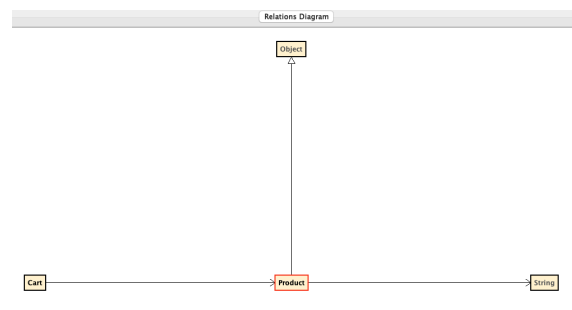
Uses: 3, Used by: 2

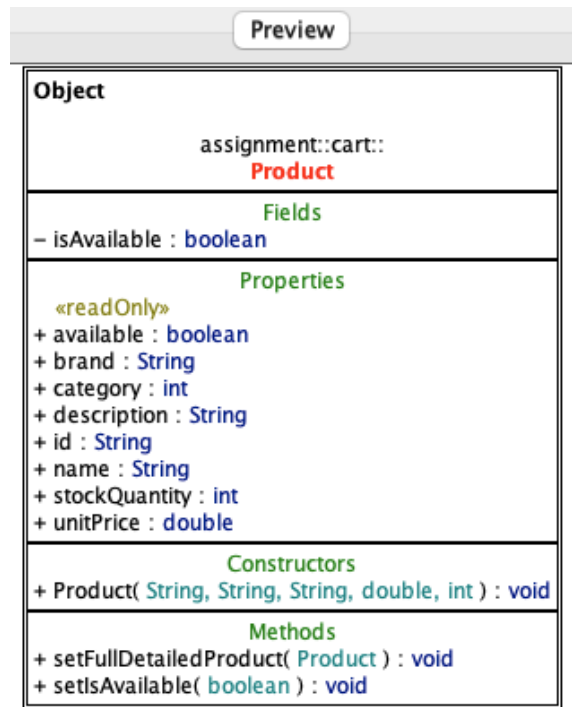




## 2.2 Product.java

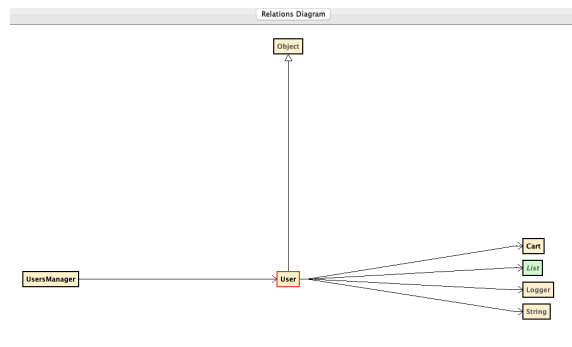
Uses: 1, Used by: 1

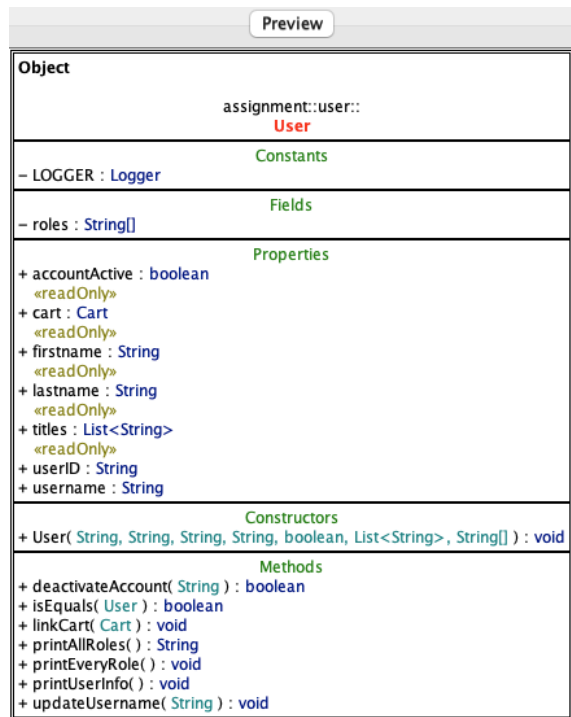




## 2.3 User.java

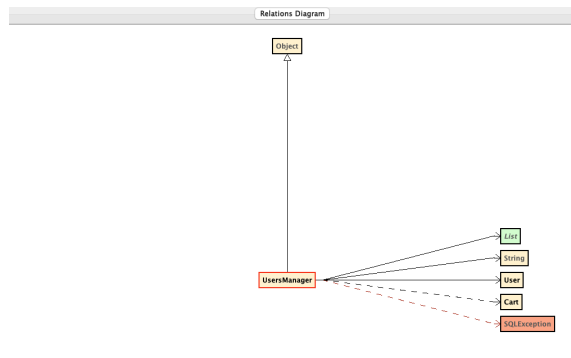
Uses: 4, Used by: 1





## 2.4 UserManager.java

Uses: 5, Used by: 0



Preview

Object

assignment::user::  
**UsersManager**

Constants

+ BASICUSERID : String  
# users : List<User>

Constructors

+ UsersManager() : void

Methods

~ addCartToUser( User, Cart ) : void  
~ addUser( User ) : void  
+ findUserFromDB( String ) : boolean  
~ removeEmptyTitlesFromUser( User ) : void