Assignment 2

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

1 Sonarlint

1.1 Cart.java

Porzione di codice:

```
if (quantity > 0)
    products.put(product, products.getOrDefault(product, 0) + quantity);
    else
products.put(product, products.getOrDefault(product, 0));
```

Tipologia: Maintainability
Correzione:

```
public void addProduct(Product product, int quantity) {
    if (quantity > 0)
        products.put(product, products.getOrDefault(product, 0) + quantity);
    else
        products.put(product, products.getOrDefault(product, 0));
}
```

Porzione di codice `updateProductQuantity`:

```
if (products.containsKey(product)==true)
    if (quantity <= 0)
        removeProduct(product);
    else
        products.put(product, quantity);
```

Tipologia: Maintainability
Correzione:

```
public void updateProductQuantity(Product product, int quantity) {
    if (products.containsKey(product)) {
        if (quantity <= 0)
            removeProduct(product);
    }
    else {
        products.put(product, quantity);
    }
}
```

Porzione di codice `getProducts`:

```
return new HashMap<Product, Integer>(products);
```

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

```
if(this.calculateTotal() >= cart1.calculateTotal() && this.calculateTotal() >= cart2
    .calculateTotal())
    return true;
else
    return false;
```

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

```
calc(cart2, cart1);
```

Tipologia: Maintainability
Correzione:

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

1.2 Product.java

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

```
public boolean IsAvailable() {
    return isAvailable;
}
```

Tipologia: Maintainability
Correzione:

```
public boolean isAvailable() {
    return isAvailable;
}
```

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

```
public void setFullDetailedProduct(
    String id,
    String name,
    String description,
    double unitPrice,
    int category,
    String brand,
    int stockQuantity,
    boolean isAvailable
)
```

Tipologia: Maintainability
Correzione:

```
public void setFullDetailedProduct(Product product) {
    this.id = product.getId();
    this.name = product.getName();
    this.description = product.getDescription();
    this.unitPrice = product.getUnitPrice();
    this.category = product.getCategory();
    this.brand = product.getBrand();
    this.stockQuantity = product.getStockQuantity();
    this.isAvailable = product.isAvailable();
}
```

1.3 User.java

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

```
public boolean isActive(){
    if(accountActive)
        return true;
    return false;
}
```

Tipologia: Maintainability
Correzione: metodo cancellato

Porzione di codice:

```
public boolean deactivateAccount(String id) {
    if (accountActive && this.userID == id) {
        accountActive = false;
        return true;
    }
    return false;
}
```

Tipologia: Reliability
Correzione:

```
public boolean deactivateAccount(String id) {
    if (accountActive && Objects.equals(this.userID, id)) {
        accountActive = false;
        return true;
    }
    return false;
}
```

Porzione di codice:

```
public boolean isEquals(User u){
    return u.userID == this.userID;
}
```

Tipologia: Reliability
Correzione:

```
public boolean isEquals(User u){
    return Objects.equals(u.userID, this.userID);
}
```

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

```
import java.util.logging.Level;
import java.util.logging.Logger;

```

```
private static final Logger LOGGER = Logger.getLogger(User.class.getName());
public void printUserInfo() {
    LOGGER.log(Level.INFO, "User Info: {0} {1} (Username: {2})", new Object[]{
    firstname, lastname, username});
}
```

Porzione di codice:

```
public void linkCart(Cart cart) throws Exception{
    if(cart == null)
        throw new Exception();
    this.cart = cart;
}
```

Tipologia: Maintainability
Correzione:

```
public void linkCart(Cart cart) {
    if(cart == null)
        throw new IllegalArgumentException("Cart cannot be null");
    this.cart = cart;
}
```

Porzione di codice:

```
public String printAllRoles(){
    return roles.toString();
}
```

Tipologia: Reliability
Correzione:

```
public String printAllRoles(){
    return Arrays.toString(roles);
}
```

Porzione di codice:

```
public void PrintEveryRole(){
    for (int i = roles.length; i > 0; i++){
        System.out.println(roles[i]);
    }
}
```

Tipologia: Maintainability
Correzione:

```
    public void printEveryRole(){
        for (int i = roles.length; i > 0; i++){
            System.out.println(roles[i]);
        }
    }
```

Porzione di codice:

```
public void printEveryRole(){
    for (int i = roles.length; i > 0; i++){
        System.out.println(roles[i]);
```

```
    }
}
```

Tipologia: Reliability
Correzione:

```
public void printEveryRole(){
    for (int i = 0; i < roles.length; i++){
        System.out.println(roles[i]);
    }
}
```

Porzione di codice:

```
public void PrintEveryRole(){
    for (int i = roles.length; i > 0; i++){
        System.out.println(roles[i]);
    }
}
```

Tipologia: Maintainability
Correzione:

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

1.4 UserManager.java

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

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

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

```
public boolean findUserFromDB(String userID) throws SQLException {
    try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/
    mydatabase", BASICUSERID+userID, "password");) {
        String query = "select firstname, lastname " + "from USERS where username="+
     (BASICUSERID+userID);
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery();
        while (rs.next())
            if(rs != null)
                return true;
        return false;

    } catch (SQLException e) {
        return false;
    }
}
```

Tipologia: Security
Correzione: Creazione di un file .env

```
public boolean findUserFromDB(String userID) throws SQLException {
    try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/
    mydatabase", BASICUSERID+userID, System.getenv("DB_PASSWORD"));) {
        String query = "select firstname, lastname " + "from USERS where username="+
     (BASICUSERID+userID);
        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet rs = stmt.executeQuery();
        while (rs.next())
            if(rs != null)
                return true;
        return false;

    } catch (SQLException e) {
        return false;
    }
}
```

Porzione di codice:

```
void removeEmptyTitlesFromUser(User user) {
    List<String> titles = user.getTitles();
    for (int i = 0; i < titles.size(); i++) {
        if (titles.get(i).isEmpty()) {
        titles.remove(i);
        }
    }
}
```

Tipologia: Maintainability
Correzione:

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

Porzione di codice:

```
void addCartToUser(User user, Cart cart) throws Exception{
    try {
        user.linkCart(cart);
    } catch (Exception e) {
        throw e;
    }
}
```

Tipologia: Maintainability
Correzione:

```
void addCartToUser(User user, Cart cart) {
    try {
        user.linkCart(cart);
    } catch (Exception e) {
        throw new IllegalArgumentException("Error linking cart to user", e);
    }
}
```

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=?"`.