Fondamenti di Ingegneria del Software a.a. 2025/2026
Laboratorio 2 – Static Analysis

Malchiodi Riccardo: s5680500
Ponassi Giulia: s6145273

SonarQube for IDE

Classe User

Prima

```java
public class User {
    private String userID;
    private String username;
    private String firstname;
    private String lastname;
    private List<String> titles = new ArrayList<String>();
    private String[] roles = new String[5];
    private boolean accountActive;
    private Cart cart;
```

The diamond operator ("<>") should be used (java:S2293)

Intentionality issue | Not clear | Maintainability

Dopo

```java
public class User {
    private String userID;
    private String username;
    private String firstname;
    private String lastname;
    private List<String> titles = new ArrayList<>();
    private String[] roles = new String[5];
    private boolean accountActive;
    private Cart cart;
```

Prima

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

Return of boolean expressions should not be wrapped into an "if-then-else" statement (java:S1126)

Intentionality issue | Not clear | Maintainability

Dopo

```java
public boolean isActive(){
    return isAccountActive();
}
```

Prima

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

Strings and Boxed types should be compared using "equals()" (java:S4973)

Intentionality issue | Not logical | Reliability

Dopo

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

Prima

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

Standard outputs should not be used directly to log anything (java:S106)

Adaptability issue | Not modular | Maintainability

Dopo

```java
public boolean isEquals(User u){
    return userID != null && userID.equals(u.userID);
}
```

Prima

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

Standard outputs should not be used directly to log anything (java:S106)

Adaptability issue | Not modular | Maintainability

Dopo

```
import java.util.logging.Logger; // aggiunto da noi per sistemare printUserInfo()
```

```
public User(String userID, String username, String firstname, String lastname,
            boolean accountActive, List<String> titles, String[] roles) {
    this.userID = userID;
    private static final Logger logger = Logger.getLogger(User.class.getName()); // serve per sostiruie i system.out
    this.username = username;
    this.firstname = firstname;
    this.lastname = lastname;
    this.accountActive = accountActive;
    this.titles = titles;
    this.roles = roles;
}
```

```
    public void printUserInfo() {
     if (logger.isLoggable(java.util.logging.Level.INFO)) {
         logger.info(String.format(format: "User info: %s %s (username: %s)", firstname, lastname, username));
     }
  }
```

Prima

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

Generic exceptions should never be thrown (java:S112)

Intentionality issue | Not complete | Maintainability

Dopo

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

Prima

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

"hashCode" and "toString" should not be called on array instances (java:S2116)

Intentionality issue | Not complete | Reliability

Dopo

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

Prima

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

Method names should comply with a naming convention (java:S100)

Consistency issue | Not identifiable | Maintainability

A "for" loop update clause should move the counter in the right direction (java:S2251)

Intentionality issue | Not logical | Reliability

Standard outputs should not be used directly to log anything (java:S106)

Adaptability issue | Not modular | Maintainability

Dopo

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

Classe UserManager

Prima

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

Modifiers should be declared in the correct order (java:S1124)

Intentionality issue | Not clear | Maintainability

Constant names should comply with a naming convention (java:S115)

Consistency issue | Not identifiable | Maintainability

Dopo

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

Prima

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

Modifiers should be declared in the correct order (java:S1124)

Intentionality issue | Not clear | Maintainability

The diamond operator ("<>") should be used (java:S2293)

Intentionality issue | Not clear | Maintainability

Mutable fields should not be "public static" (java:S2386)

Consistency issue | Not conventional | Maintainability

Dopo

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

Prima

```
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;
    }
}
```

Credentials should not be hard-coded (java:S6437)

Responsibility issue | Not trustworthy | Security

Why is this an issue? | How can I fix it? | More Info

Dopo

```
public boolean findUserFromDB(String userID) throws SQLException {
    String dbUrl = "jdbc:mysql://localhost:3306/mydatabase";
    String dbUser = System.getenv(name: "DB_USER");
    String dbPassword = System.getenv(name: "DB_PASSWORD");

    String query = "SELECT firstname, lastname FROM USERS WHERE username = ?";

    try (Connection conn = DriverManager.getConnection(dbUrl, dbUser, dbPassword);
         PreparedStatement stmt = conn.prepareStatement(query)) {

        stmt.setString(parameterIndex: 1, BASIC_USER_ID + userID); // sostituisce il primo ? nella query con l'username completo dell'utente
        ResultSet rs = stmt.executeQuery();

        return rs.next(); // true se esiste almeno una riga
    }
}
```

NOTA: per convenzione abbiamo messo le istruzioni SQL in maiuscolo.

Prima

```
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);
      }
    }
}
```

'List.remove()' should not be used in ascending 'for' loops (java:S5413)

Intentionality issue | Not clear | Maintainability

Dopo

```
void removeEmptyTitlesFromUser(User user) {
    List<String> titles = user.getTitles();
    for (int i = titles.size() -1; i >= 0; i--) {
      if (titles.get(i).isEmpty()) {
        titles.remove(i);
      }
    }
}
```

Prima

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

"catch" clauses should do more than rethrow (java:S2737)

Intentionality issue | Not clear | Maintainability

Dopo

```
void addCartToUser(User user, Cart cart){
  user.linkCart(cart);
}
```

Alcune problematiche non rilevate da SonarQube ma che pensiamo possano essere migliorate:

- In User, l'uso diretto di liste/array viola l'information hiding
- roles viene gestito come array fisso, poco flessibile.
- In UsersManager, la lista users è statica e condivisa, causando possibili problemi di concorrenza e design.
- UsersManager mescola logica di business e accesso al database, violando il principio di *Single Responsibility*.

DesigniteJava

a) I metodi con Complessità ciclomatica più alta nell'intero progetto sono:

- removeEmptyTitlesFromUser (CC = 3)
- updateProductQuantity (CC = 3)

b)

Classe User:

1) Metodo: **User**
    a) **Tipo smell:** Long parameter List
    b) **Perché è un problema:** Metodi o costruttori con molti parametri sono difficili da invocare correttamente. È facile confondere l'ordine degli argomenti o dimenticarne uno.
    c) **Effetti Negativi:** Riduce la leggibilità e aumenta la probabilità di errori.

2) Metodo: **deactivateAccount**
    a) **Tipo smell:** Complex Conditional
    b) **Perché è un problema:** Le espressioni booleane complesse sono difficili da comprendere rapidamente.
    c) **Effetti Negativi:** Rende il codice difficile da leggere e da testare.

Classe UsersManager:

1) Classe: **UsersManager**
    a) **Tipo smell:** unutilized Abstraction
    b) **Perché è un problema:** Se la classe non è mai usata, è "codice morto".
    c) **Effetti Negativi:** Aggiunge ingombro e complessità inutile alla codebase. Chi legge il codice potrebbe perdere tempo a cercare di capire una classe che non ha alcun impatto sul programma.

2) Metodo**: findUserFromDB**
    a) **Tipo smell:** Long Statement.
    b) **Perché è un problema:** Uno statement è considerato "lungo" quando contiene troppi elementi o operazioni nella stessa riga. In questo caso, l'istruzione di apertura della connessione al database (*DriverManager.getConnection(...)*) risulta lunga perché include l'URL, l'utente e la password nella stessa riga.
    c) **Effetti Negativi:** Uno statement troppo lungo può ridurre la leggibilità del codice e rendere più difficile identificare rapidamente i parametri o le eventuali modifiche future.
    d) **NOTA:** in questo caso lo statement (secondo noi) è necessario per aprire una connessione in un'unica istruzione e non può essere semplificato ulteriormente. Viene identificato come smell ma possiamo "ignorarlo" in quanto lo statement è semanticamente compatto e leggibile.

c) Sì, UsersManager non rispetta pienamente i principi di **astrazione, decomposizione, modularità e information hiding** perché combina troppe funzionalità in un'unica classe e gestisce direttamente i dettagli del DB, riducendo flessibilità e chiarezza. Inoltre non rispetta la coesione in quanto esegue più operazioni; la coesione ci dice che una classe deve svolgere un solo compito (viene anche indicata con la dicitura SRP).

d) Sì, oltre a violare i principi di buon design come spiegato **nel punto c** abbiamo:

1. Assenza di separazione dei tre stati. La classe mescola:
    - **Data logic:** accesso al DB (findUserFromDB)
    - **Business Logic:** gestione utenti e carrelli (addUser, addCartToUser)
    - Non c'è un chiaro confine tra i livelli, quindi il sistema non segue l'architettura a tre stati, che rende più difficile manutenzione, test e evoluzione futura.

2. Ridotta modularità e chiarezza.

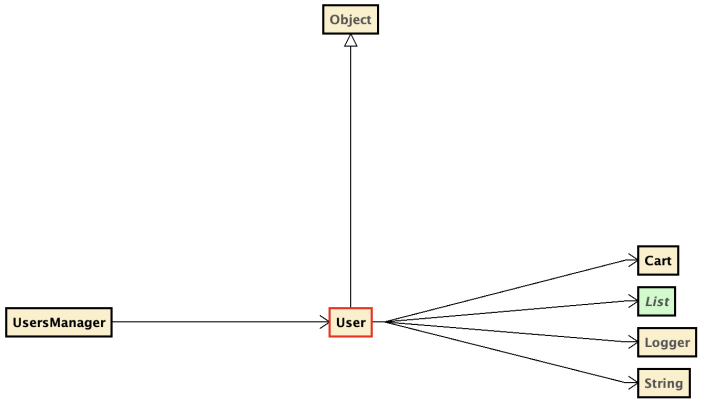
3. il file **ArchitectureSmells.csv** rileva un problema, ma solo nel codice autogenerato da Gradle.
    - **Smell:** Feature Concentration (Concentrazione di Funzionalità).
    - **Dove:** Package org.gradle.accessors.dm.

- **Spiegazione:** Questo smell indica che il package sta gestendo troppe funzionalità diverse e non correlate tra loro.

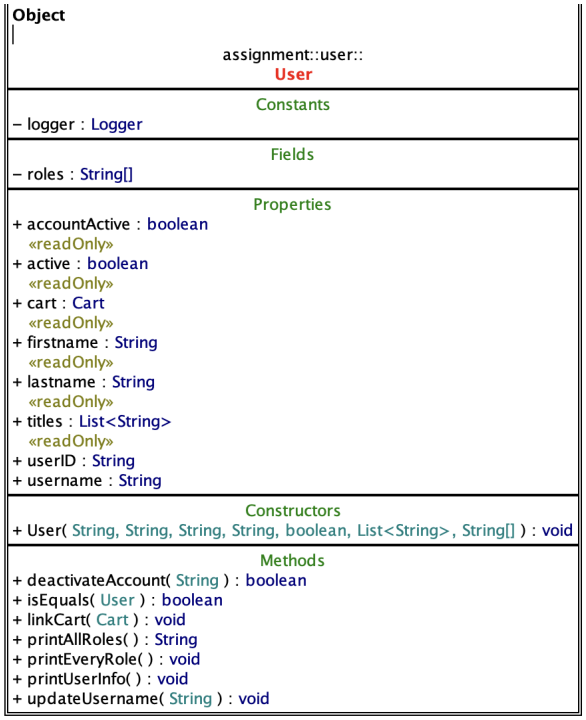
Class Visualizer

Classe User

Relations Diagram:



Preview:

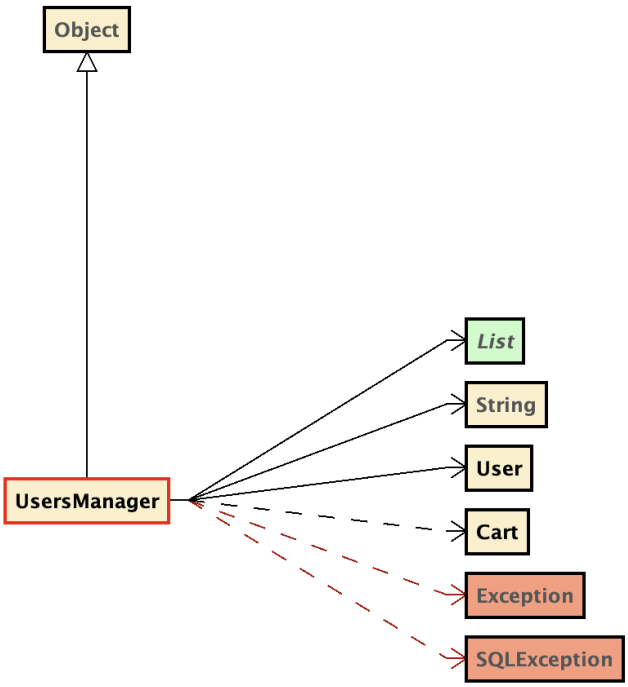


Uses: 4

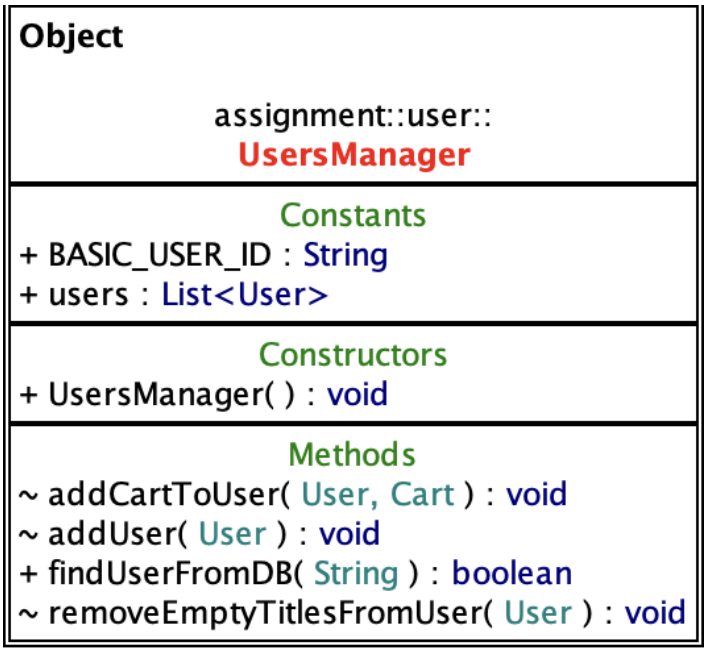
Used by: 1

Classe UsersManager

Relations Diagram:



Preview:

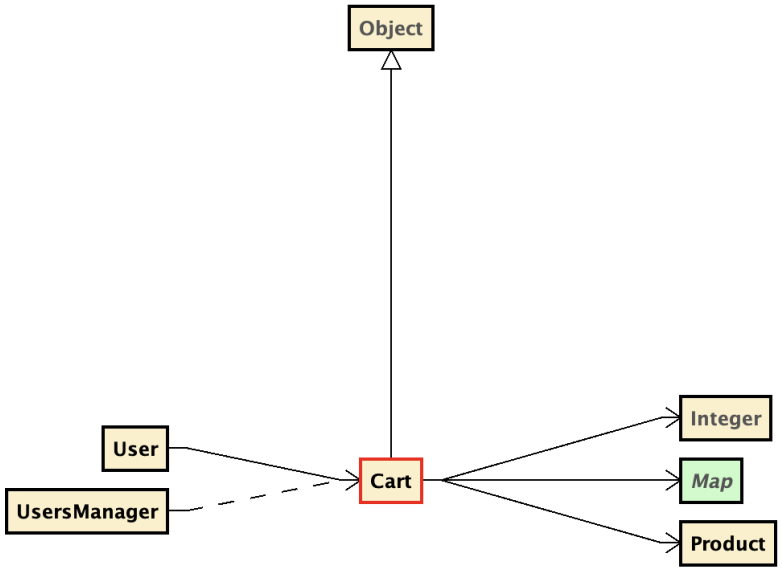


Uses: 6

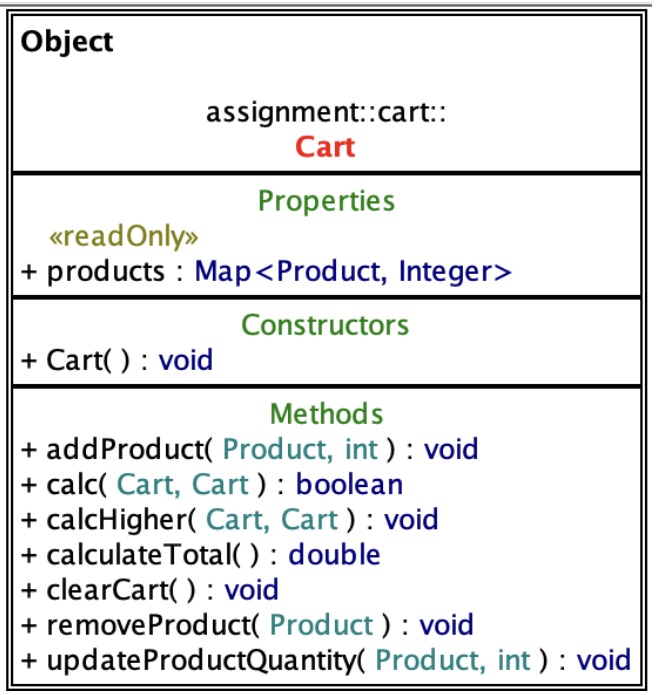
Used by: 0

Classe Cart

Relations Diagram:



Preview:

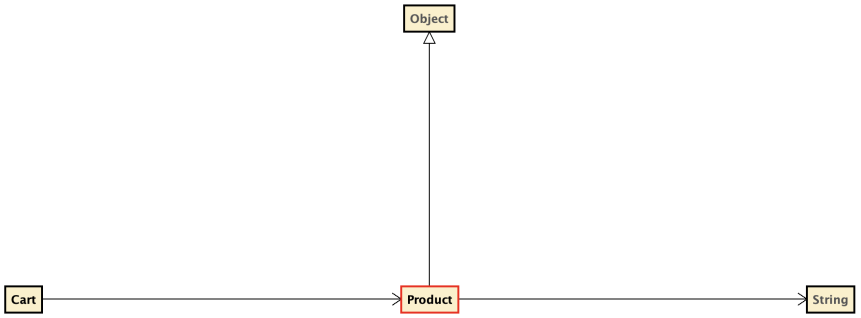


Uses: 3

Used by: 2

Classe Product

Relations Diagram:



Preview:

Object

assignment::cart::
Product

Fields
– isAvailable : boolean

Properties
+ brand : String
+ category : int
+ description : String
+ id : String
+ name : String
+ stockQuantity : int
+ unitPrice : double

Constructors
+ Product(String, String, String, double, int) : void

Methods
+ IsAvailable( ) : boolean
+ setFullDetailedProduct(String, String, String, double, int, String, int, boolean) : void
+ setIsAvailable(boolean) : void

Uses: 1

Used by: 1