Identify Code Smells in The SharingApp Project

Jeff Pal

Aug 2023

1. **Duplicate Code**

**1.1. Where does the smell come from?**

Classes:

public class ItemList extends *Observable*

public class ContactList extends *Observable*

Methods:

public *boolean* saveItems(*Context* *context*)

**1.2. What is that smell?**

The implementation of the method saveItems(*Context context*), from classes ItemList and ContactList, are similar and have slight differences, so it's a **duplicate code**.

**1.3. What’s the problem?**

If something needs to change, the code will **need to be updated in multiple places**, what can cause inconsistencies.

**1.4. What's the solution?**

A storage-like class should be created to perform this kind of action.

**2. Data Clumps**

**2.1. Where does the smell come from?**

Classes:

public class AddItemActivity extends *AppCompatActivity*

public class EditItemActivity extends *AppCompatActivity* implements *Observer*

**1.2. What is that smell?**

The classes AddItemActivity and EditItemActivity have groups of data appearing together in different parts of the code as instance variables, what indicates a **data clumps**.

**1.3. What’s the problem?**

That can cause issues related to **duplicate code**, as well as **reducing readability** and **lack of abstraction**, which in turn can violate the principle of encapsulation and abstraction.

**1.4. What's the solution?**

Related data items should be grouped together into a **separate class or structure**, rather than being scattered throughout the codebase. By encapsulating related data into a single class, it centralizes the management of that data.