Metode Avansate de Programare, sectia Informatică 2022–2023, LABORATOR 7

DEADLINE: Săptămâna 13-14

Contents

Partea I – Cerinte

Continuati laboratorul 6, adaugand urmatoarele doua functionalitati:

- **F3. Retragerea unei cereri de prietenie, daca utilizatorul care a trimis-o se razgandeste. (2p)**
- **F4. Simulati trimiterea de mesaje printr-o interfata grafica (5p)**
- **Cerința non-funcționala - calitatea codului: cohesion, coupling, inheritance, complexity (2p)**
  - Folositi Metrics Reloaded pt calculul metricilor de cohesion, coupling, complexity, inheritance, identificate in suportul teoretic de mai jos. Mai precis se vor calcula următoarele metrici:
    - Pt COHESION:
      - LCOM Lack of Cohesion of Methods (LCOM1, LCOM2, LCOM3, LCOM4)
      - Cohesion among Methods of a Class (CAMC)
      - Normalized Hamming Distance (NHD)
    - COUPLING:
      - Coupling between objects (CBO)
      - Afferent Coupling (Ca)
      - Efferent Coupling (Ce)
      - Instability (I)
    - INHERITANCE:
      - Depth of Inheritance Tree (DIT)
      - Number of Children (NOC).
    - COMPLEXITY
      - Metricile stabilite pt lab 5
  - Construiti o noua arhiva cu numele grupa_nume_prenume si uploadati-o la adresa: arhiva-proiecte-metrici Aceasta arhiva va trebuie sa contina si codul sursa analizat pt lab 7 si codul sursa de la lab 5

Partea II – Suport teoretic

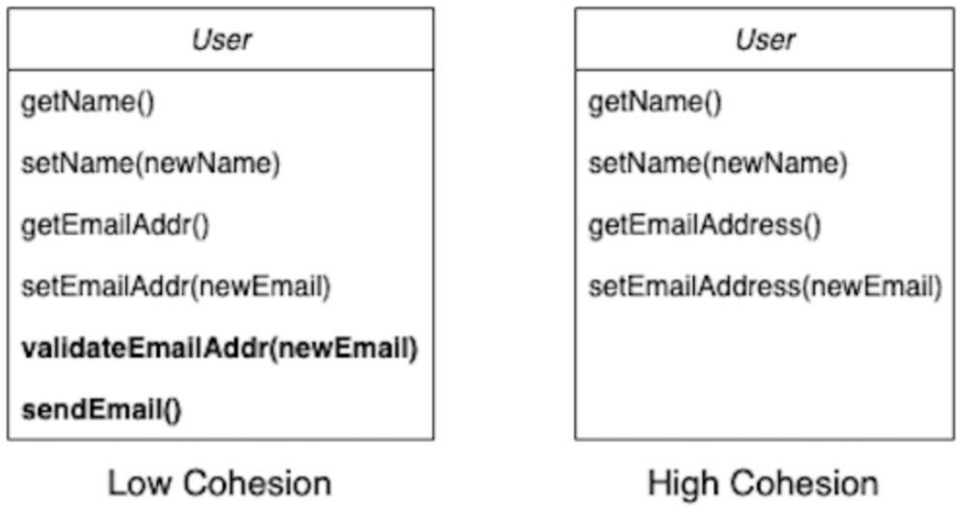
Code cohesion– What/Why/How

One essential principle in object-oriented software design is that modules should have high cohesion and low coupling. In this lesson, we will refer to classes, packages and domains as modules.

1. What is code cohesion?

Cohesion represents the degree to which the components of a module belong together. In other words, cohesion denotes how related the elements from that module are.

For example, we can see below that the first User class has a low cohesion because it contains methods that should not belong in this class, whereas the second class is more precise and only includes methods related to the user.



2. Why is code cohesion important?

A class with low cohesion breaks one of the most well-known object-oriented principles, the Single Responsibility Principle. It indicates that the module has more than one reason to change. Therefore having a high cohesion value of modules is desired. High cohesion relates to software quality attributes such as maintainability, readability and understandability. More precisely, if a module has high cohesion, making changes to a functionality is more manageable and less risky because all the classes related to that functionality are in one place.

3. How to measure code cohesion?

Software metrics that can be used to measure the code cohesion:

- LCOM Lack of Cohesion of Methods (LCOM1, LCOM2, LCOM3, LCOM4)
- Cohesion among Methods of a Class (CAMC)
- Normalized Hamming Distance (NHD)

Resources:

- https://www.baeldung.com/cs/cohesion-vs-coupling
- https://medium.com/clarityhub/low-coupling-high-cohesion-3610e35ac4a6
- https://www.newthings.co/blog/coupling-and-cohesion-guiding-principles-for-clear-code/

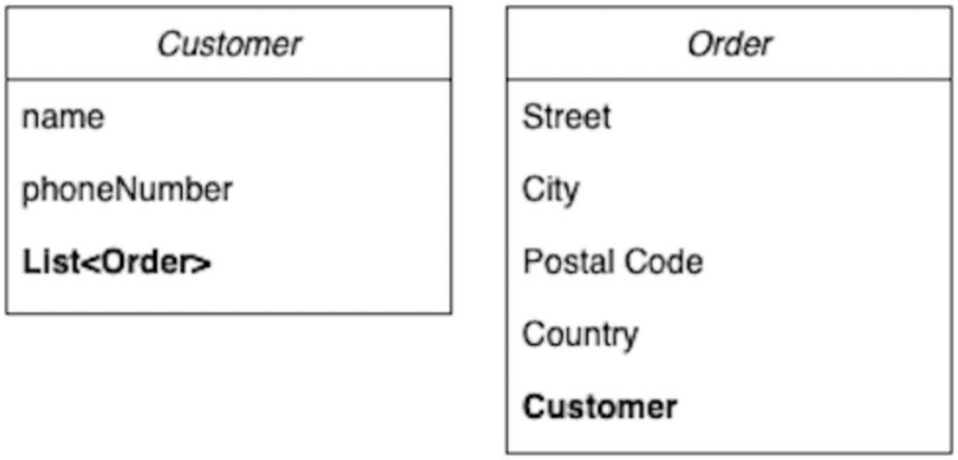
Code coupling – What/Why/How

One essential principle in object-oriented software design is that modules should have high cohesion and low coupling. In this lesson, we will refer to classes, packages and domains as modules.

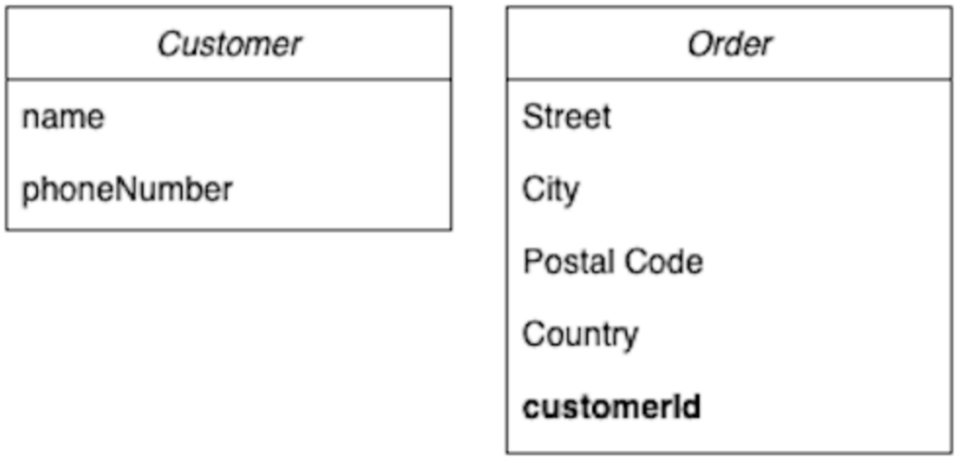
4. What is code coupling?

Coupling represents the degree of interdependence between modules. In other words, coupling refers to how dependent one module is on other modules.

As you can see in the examples below, tight coupling means that the *Customer* and *Order* classes have references to each other. Unfortunately, this means that each time the *Customer* creates a new order, we also need to update the *Customer* class, which is redundant and generates extra complexity. By having only the *customerId* as a reference inside the *Order* class, we reduce the coupling.



Tight Coupling



Loose Coupling

5. Why is code coupling important?
   A good software design should have low coupling between modules. This means that the modules don't have knowledge of the internal structure of the other modules. Therefore, making changes in one module won't affect the others depending on it. Code coupling impacts software quality attributes such as maintainability, testability and readability.

6. How to measure code coupling?

   Software metrics that can be used to measure the code coupling:

   - Coupling between objects (CBO)

- Afferent Coupling (Ca)
- Efferent Coupling (Ce)
- Instability (I)

Resources:

- https://www.baeldung.com/cs/cohesion-vs-coupling
- https://medium.com/clarityhub/low-coupling-high-cohesion-3610e35ac4a6
- https://www.newthings.co/blog/coupling-and-cohesion-guiding-principles-for-clear-code/