Testdriven utveckling i Java

Inlämningsuppgift 3

Command

Ofta när jag gör program där det finns många villkor, brukar jag skapa enorma else-if eller switch cases. I dessa är skillnaden oftast bara värdet på variabeln som skiljer. Trots att detta fungerar, blir det väldigt jobbigt att följa den stora mängd kod som i stort sett gör samma sak. Jag skulle kunna minska denna mängden med hjälp av designmönstret *Command*.

I en av mina första uppgifter på EC Utb skulle man göra ett school management system, där man kunde registrera nya studenter och kurser, samt registrera studenter till dessa kurser. Detta implementerades med hjälp av DAO-klasser. Enligt instruktionerna skulle dessa innehålla vissa metoder, men man fick själv välja hur man ville implementera dem. Nedan i bilden (*fig 1*) kan man se en del från den första av många switch cases, där man kan lägga till studenter. Utanför bild, längre ned i koden, hittar man en snarlik switch case som hanterar kurser.

```
16          public static void main(String[] args) {
17                  Scanner in = new Scanner(System.in);
18                  while(session) {
19                          System.out.print("What do you want to do?" + "\nEnter 1 to add or find students, "
20                                          + "\nEnter 2 to create new course or find an existing one, \nEnter 3 to register students to course or "
21                                          + "to edit student or course.\nEnter 4 to quit \nSelection: ");
22                          String firstAnswer=in.nextLine();
23                          switch(firstAnswer) {
24                          case "1":
25                                  System.out.print("\nEnter 1 to add a student, \nEnter 2 to find from Id, "
26                                                  + "\nEnter 3 to find from name, \nEnter 4 to find from email, "
27                                                  + "\nEnter 5 to show all registered students, \nEnter 6 to remove selected student.\nSelection: ");
28                                  answer=in.nextLine();
29                                  switch(answer) {
30                                  case "1":
31                                          System.out.println("\nEnter Id, name, email and address in this order: ");
32                                          int id = Integer.parseInt(in.nextLine());
33                                          String name = in.nextLine();
34                                          String email = in.nextLine();
35                                          String address = in.nextLine();
36                                          Student student = new Student(id, name, email, address);
37                                          System.out.println("Student added: \n   " + StudentDaoList.StudentToString(studentList.saveStudent(student)) +
38                                          break;
39                                  case "2":
40                                          read = studentList.findAll().iterator();
41                                          System.out.println();
42                                          while(read.hasNext()) {
43                                                  System.out.println("Id: " + read.next().getId());
44                                          }
45                                          System.out.print("\nId: ");
46                                          id = Integer.parseInt(in.nextLine());
47                                          System.out.println("Student: \n " + StudentDaoList.StudentToString((studentList.findById(id)))+"\n");
48                                          break;
49                                  case "3":
50                                          read = studentList.findAll().iterator();
51                                          System.out.println();
52                                          while(read.hasNext()) {
53                                                  System.out.println("Name: " + read.next().getName());
54                                          }
55                                          System.out.print("\nName: ");
56                                          name = in.nextLine();
57                                          read = studentList.findByName(name).iterator();
58                                          while(read.hasNext()) {
59                                                  System.out.println("Student: \n " + StudentDaoList.StudentToString(read.next())+"\n");
60                                          }
61                                          break;
62                                  case "4":
```

fig 1.

Detta blir snabbt väldigt svårläst, även om man bortser från hur jag då hanterade input från scannern. Här skulle man kunnat använda sig av ett antal hjälpklasser som hanterar anropen till och från mina DAO-klasser, istället för att göra två klasser och main som var beroende av varandras funktion för att på så sätt få en lägre coupling. Trots att man måste göra fler klasser, blir min main mycket lättare att läsa genom att man istället skickar sin order till en receiver, som i sin tur skickar detta vidare till respektive invoker, sett nedan i bilden (*fig 2*). Hela koden finns på https://github.com/crazvik/assignment3, men är inte fullt implementerad.

```java
    public static void main( String[] args ) {
        Scanner scanner = new Scanner(System.in);
        StudentDaoList students = new StudentDaoList(new ArrayList<>());
        CourseDaoList courses = new CourseDaoList(new ArrayList<>());
        String type, action;
        LocalDate localDate;
        StudentManager manager = new StudentManager(students, courses);

        System.out.print("Enter student or course: ");
        type = scanner.nextLine();
        switch (type.toLowerCase()) {
            case "student":
            case "course":
                System.out.println("Enter an action to take: ");
                action = scanner.nextLine();
                switch (action.toLowerCase()) {
                    case "save":
                        manager.save(type);
                        break;
                    case "delete":
                        manager.delete(type);
                        break;
                    case "read":
                        System.out.println("Enter an action to take: ");
                        action = scanner.nextLine();
                        switch (action.toLowerCase()) {
                            case "email":
                                System.out.println("Enter email to search for");
                                manager.read(type, action, scanner.nextLine(), "", 0, LocalDate.MIN);
                                break;
                            case "name":
                                manager.read(type, action, "", scanner.nextLine(), 0, LocalDate.MIN);
                                break;
                            case "id":
                                manager.read(type, action, "", "", Integer.parseInt(scanner.nextLine()), LocalDate.MIN);
                                break;
                            case "date":
                                localDate = LocalDate.parse(scanner.nextLine());
                                manager.read(type, action, "", "", 0, localDate);
                                break;
                        }
                        break;
                }
                break;
        }
    }
}
```

fig 2.