Inlämningsuppgift 3 – Design patterns, exempel

Builder pattern

I ett tidigare skolprojekt där vi skulle göra en ”Todo app”, alltså en app med uppgifter för användarna, blev AppUser-konstruktorerna lite småjobbiga. Detta var inget stort problem på något sätt, men jag tänkte ändå göra om denna kod med hjälp av Builder pattern.

Dessa tre konstruktorer användes:

```
public AppUser(String userName, String firstName, String lastName,
LocalDate regDate, String password,
               double balance, Set<AppUserRole> roleSet, Set<TodoItem>
todoItems) {
    this.userName = userName;
    this.firstName = firstName;
    this.lastName = lastName;
    this.regDate = regDate;
    this.password = password;
    this.balance = balance;
    this.roleSet = roleSet;
    this.todoItems = todoItems;
}

public AppUser(String userName, String firstName, String lastName,
LocalDate regDate, String password, double balance, Set<AppUserRole>
roleSet) {
    this.userName = userName;
    this.firstName = firstName;
    this.lastName = lastName;
    this.regDate = regDate;
    this.password = password;
    this.balance = balance;
    this.roleSet = roleSet;
}

public AppUser(String userName, String firstName, String lastName,
LocalDate regDate, String password, double balance) {
    this.userName = userName;
    this.firstName = firstName;
    this.lastName = lastName;
    this.regDate = regDate;
    this.password = password;
    this.balance = balance;
}
```

Med min nyvunna kunskap om Builders gjorde jag om denna kod. Jag la till ett UUID för id. Satte detta som ett nödvändigt fält, samt userName och password. Det är vad som krävs i denna app för att vara en AppUser.

I klassen AppUser lade jag till klassen Builder. Dess konstruktor ser ut såhär:

```
public Builder(UUID userId, String userName, String password) {
    this.userId = userId;
    this.userName = userName;
    this.password = password;
}
```

”Required fields” alltså. I klassen Builder gjorde jag sedan setter-metoder för resten av fälten, ”optional fields”.

```
public Builder withFirstName(String firstName) {
    this.firstName = firstName;
    return this;
}
```

Builder avslutas med en build-metod. AppUser-klassen har sina fält lika tidigare men konstruktorn är tom. La till getters och setters, men behövs inte för Buildermönstret.

För att instansiera ett objekt med hjälp av builder gör man såhär:

```
AppUser appUser1 = new AppUser.Builder(UUID.randomUUID(), "test1", "your
password").build();
```

appUser1 har endast de nödvändiga fälten.

appUser2 har även för- och efternamn. Det spelar ingen som helst roll i vilken ordning fälten sätts, eftersom metoderna är namngivna.

```
AppUser appUser2 = new AppUser.Builder(UUID.randomUUID(), "test2", "your
password")
        .withLastName("Anka")
        .withFirstName("Arne")
        .build();
```

Att skriva en Builder är inget jättejobb. När man väl gjort det kan man på ett smidigt och enkelt sätt skapa objekt med stor flexibilitet. En viktig sak är att man dessutom inte behöver tänka på ordningen på parametrarna då man skapar objekt (även om dagens IDE:er hjälper en väldigt mycket här). Jag läste ett kul exempel på https://dzone.com/articles/design-patterns-the-builder-pattern där ränta och saldo väldigt lätt kunde blandas ihop i en bankkontokod. Jag kommer helt klart använda mig av builders i framtiden då klasserna blir lite större och flexibilitet önskas!

Min kod hittar du på https://github.com/Loffis/TDD-Assignment3