Peer Review av *AEY Investment Group* träningsapp i TDA367

Anton Ekström, Elin Nilsson, Arvid Svedberg, Jacob Bengtsson, Amanda Papacosta

Do the design and implementation follow design principles?

Till stora delar följer koden Single- Responsibility principle då koden är väl uppdelad och de olika klasserna håller lite funktionalitet var men med vissa undantag. Exempelvis factory-klassen "WorkoutPlanFactory" som skapar både Exercises, Workouts och Plans.

Det finns många publika metoder samt getters och setters, vilket borde kollas på. I modellen används varken arv eller composition. Det finns heller inga interfaces i koden.

Does the project use a consistent coding style?

I stora drag så är koden sig lik i hela projektet. På vissa ställen har namn skapats på olika sätt (stor bokstav, understreck).

Is it easy to maintain?

Koden är för närvarande ganska enkel i modellen så den kommer vara enkel att jobba med och därför borde det gå lätt att lägga till ytterligare funktionalitet vilket är positivt.

Can we easily add/remove functionality?

Eftersom det inte finns särskilt mycket logik i modellen så lär det vara förhållandevis enkelt att utveckla på den. Det finns inte heller några mönster eller designval som skulle försvåra utökning på kodbasen.

Are design patterns used?

Ett antal designmönster används i projektet, följande har vi identifierat:

Factory mönstret utnyttjas för att abstrahera instansieringen av objekt. Det utnyttjas väl. Klassen *WorkoutPlanFactory* skapar flera objekt med orelaterade typer, vi hade tyckt det varit mer lämpligt om varje Factory enbart skapade en typ av objekt. Namnet kan ge intrycket att factoryn skapar klassen *WorkoutPlan* men denna existerar inte.

Klassen *PlanBuilder* ser ut som den var tänkt att implementera designmönstret *Builder*, dock saknar den metoder som skapar objekten. Klassen kanske inte är färdig men den skulle även behöva en metod för att instansiera en *Plan*.

Det finns också en början på användning av mönstret *Adapter* men det verkar inte vara färdigställt då inga interfaces hittas.

Are proper names used?

Ja, det känns enkelt att förstå vilka ansvar de olika klasserna har då namnen är beskrivande. Ett undantag är kanske *WorkoutPlanFactory* och *PlanBuilder* klasserna som inte riktigt (än i alla fall) gör det som deras namn säger. Men de grundläggande klasserna som *Plan*, *Workout* och *Excercise* är väldigt välbeskrivande.

Is the design modular? Are there any unnecessary dependencies?

Syns inga starka dependencies mellan klasserna utan de verkar rätt självständiga.

Does the code use proper abstractions?

Det finns inte alldeles för många abstraktioner i form av gränssnitt eller abstrakta klasser, men det finns början på *Builder* och *Factory* mönstren som abstraherar objektskapande.

Is the code well tested? Are there any security problems, are there any performance issues?

Det finns en bra början för att testa programmet, men ej färdigställd då det endast finns ett fåtal tester.

Det finns en hel del kvar att jobba på när det gäller funktionalitet. Det funkar bra att byta mellan olika flikar men det är också det enda man kan göra. Därför är det svårt att testa säkerhetsproblem och prestanda.

Is the code easy to understand? Does it have an MVC structure, and is the model isolated from the other parts?

Istället för MVC så följer projektet en MVVM- struktur. Det görs med en tydlig uppdelning av Model, View och View-Model. Modellen är avskild från de andra delarna.

Sammanfattande tankar:

Vissa user-stories kanske hänger ihop lite väl mycket, de har samma confirmation-kriterier. Detta gäller främst de första två. Kodbasen har en bra början men fortfarande saknas mycket av funktionaliteten som visas upp i projektets RAD. Kodbasen visar på bra extensibility och därför bör programmet kunna fortsätta att utformas på ett bra sätt. Koden är också väl kommenterad på flera ställen med tydliga beskrivningar av metoder. Det verkar finnas en bra grund att arbeta vidare med, men som behöver en del jobb för att få ihop ett färdigt och användbart program.