Lisa Westlund
Caroline Forslund
Johanna Szepanski

PEER REVIEW workshop av Jan Tran

Test the runnable version of the application in a realistic way. Note any problems/bugs.

- Användare kan själv skriva in medlemsid och sätta samma id på flera medlemmar. Det finns ingen kontroll för att id:n är unika.
- Använder medlemmens position i listorna och inte medlemsid för att utföra olika operationer. Vilket gör det onödigt komplicerat. Det fungerar inte med ReadLine() när man ska redigera medlem om man inte trycker Enter flera gånger.
- Om man redigerar en medlem och lämnar namnet tomt går det inte att lägga tillbaka ett namn igen.
- Den kompakta listan ska inte innehålla personnummer.
- Båttyp och båtlängd saknas från det utökade listan. Dock behöver inte antalet båtar vara med.
- Båttypsmenyn är lite ihoptrasslad. Sparar inte till den angivna kategorin. Lägger man till en motorseglare så blir en kanot. Kanot blir annan.
- Bra med felmeddelandet som gör att man slipper krasch i applikationen.
- Bra att man ser medlemsinfo när man redigerar en enskild medlem.

Try to compile/use the source code using the instructions provided. Can you get it up and running? Is anything problematic? Are there steps missing or assumptions made?

Bra med en README-fil med instruktioner. Inga konstigheter att köra programmet.

Does the implementation and diagrams conform (do they show the same thing)? Are there any missing relations? Relations in the wrong direction? Wrong relations? Correct UML notation?

Klassdiagrammet saknar relationer överhuvudtaget och innehåller alla implementerade attribut och metoder. I ett klassdiagram behöver inte allt finnas med. Man väljer ut det viktigaste för att undvika röriga och svårbegripliga diagram. Se t ex Larman [1, p258, ch16.6].

Det Input-sekvensdiagram som handlar om att ta bort en båt stämmer bra, men eftersom vi tycker att en del kod borde omstruktureras bör även diagrammet ritas om.

Output-sekvensdiagrammet måste delas upp i många små sekvensdiagram. Ett sekvensdiagram ska visa ett scenario av ett användarfall. Se Larman [1, p177, ch10.5]. Kom dock ihåg att du inte behöver göra alla användarscenario i denna workshop. Case 4 som ligger längst ner hade kunna vara ett eget diagram.

Is the Architecture ok?

Is there a model view separation, is the model coupled to the user interface, is the model specialized for a certain kind of IU (for example returning formated strings to be printed), are there domain rules in the UI?

I stort tycker vi att applikationen följer MVC principen med några undantag. Se nedan.

UserController
Applikationen läser in användarinput Läser in i controllern vilket strider mot MVC principen. Viewen borde läsa in input som sedan hämtas controllern. Du initierar flera objekt av typerna ConsoleView och MomberModel. Går det inte att återvända de som redan skapats vore det bra om de hette olika saker för att undvika förvirring. Ett alternativ kan vara att ha fler viewer som har olika ansvarsområden.

BoatModel
När undantag kastas är det bättre att viewen sätter meddelandet.

MemberModel
Membermodellen har en association till Båtmodellen genom en lista av typen Boat. Medlemsobjektet och Båtobjektet hade kunnat hållas helt skilda. Som det är nu kan Membermodellen inte återanvändas till något annat än just den här båtklubben. Modellen innehåller metoder som bättre hör hemma i DAL, t ex removeBoatList(). Eftersom DAL sköter kontakten med textfilen bör de metoder som skriver till eller läser från filen finnas där.

Överlag kunde instanserna av de olika objekten återanvändas bättre.

ConsoleView
Rätt saker händer i viewn. Den skulle hantera ännu mer som i nuläget sker i controllern. T ex input vid menyval. De kompakta- och utökade listorna är exakt likadan, vilket nog förklarar varför viss data inte hamnar på rätt ställe. Lite död kod här och där.

Is the requirement of a unique member id correctly done?
Användaren kan själv ange medlemsid och kontroll saknas för att säkerställa att två medlemmar inte får samma id. Alltså är de inte unika.

What is the quality of the implementation/source code?
En hel del bortkommenterad testkod ligger kvar. Blandad engelska och svenska. Bra med getters och setters i modellerna.

Code Standards
Skulle kunna snygga till koden lite, med konsekventa radbrytningar och indenteringar.

Naming
Namngivningen är inte självförklarande. Flera variabler består enbart av en bokstav, vilket gör koden onödigt svåra att förstå. Hade varit lättare om det funnits fler kommentarer, men det saknas lite varstans. Variabelnamn som test1 och test2 har inte ändrats till bättre namn innan release.

Duplication
Inte något anmärkningsvärt.

Dead Code
Boat modellen innehåller en metod Add() som inte används. Add() och AddMember() i membermodellen används inte heller. Projektet behöver en genomgång då det finns oanvänd kod lite här och där.

What is the quality of the design? Is it Object Oriented?

Objects are connected using associations and not with keys/ids.
I Membermodellen finns association till Boat. Hade kunnat lösas utan att koppla ihop båtmodellen till medlemsmodellen. DAL hade kunnat sköta det. Se Larman [1, p299, ch17.12] där det finns information om low coupling.

Is GRASP used correctly?
DAL är den klass som talar med och har information om textfilerna. Flera metoder som ligger i objektklasserna hade bättre hört hemma där. Viewen borde skapa objekten eftersom informationen kommer in via användaren och skicka dem vidare till kontrollern som i sin tur skickar informationen vidare till DAL. Det finns att läsa mer om creator i Larman [1, p292, ch17.10]

Classes have high cohesion and are not too large or have too much responsibility.
Mängden kod är ok. Inga stora klasser. Member objektet har för många uppgifter och för stort ansvar. Skickar info till DAL som borde gått via kontrollern.

Classes have low coupling and are not too connected to other entities.
Se ovan om kommentarer mellan medlem och båt.

Avoid the use of static variables or operations as well as global variables.
Inte hittat några.

Avoid hidden dependencies.
Inte vad vi kan se.

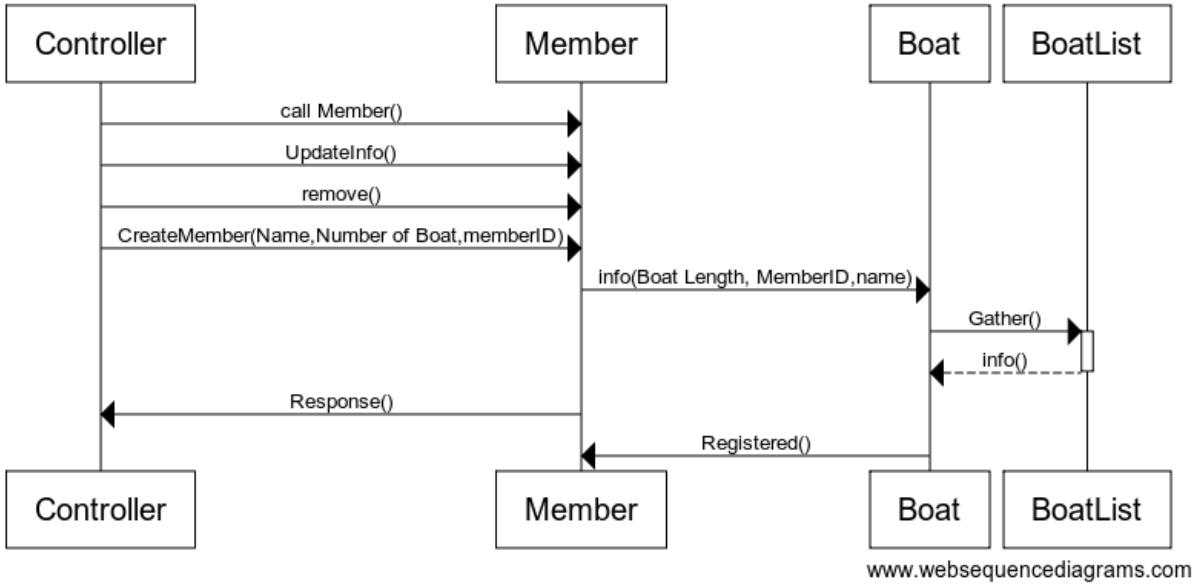
Information should be encapsulated.
Ok med namespaces. Finns getters och setter.

Inspired from the Domain Model.
Ja, såtillvida att en ägare kan registrera en eller flera båtar.

Primitive data types that should really be classes (painted types)
Medlemsid skulle kunna genereras automatiskt och hanteras via en egen klass.

As a developer would the diagrams help you and why/why not?
Klassdiagrammet innehåller inga relationer som berättar hur klasserna ska hänga ihop. De innehåller för mycket information och gör diagrammet rörigt och lämnar lite över till egna lösningar.

Sekvensdiagrammet visar tydligt problemet med den onödigt starka kopplingen mellan Member och Boat. Diagrammet hade inte hjälpt mig som utvecklare att göra en bättre applikation.



What are the strong points of the design/implementation, what do you think is really good and why?
En relativt bra MV-separation som med några mindre justeringar blir mycket bättre. Vi har själva jobbat med att skriva till en textfil och upplevde det svårt att få rätt sak på rätt plats. Det har du dock hanterat bra.

What are the weaknesses of the design/implementation, what do you think should be changed and why?
Se ovan.

Do you think the design/implementation has passed the grade 2 criteria?
Om du fixar MV separationen och ser till att uppfylla kraven i uppgiften(unikt medlemsid och visning av listor samt uppbyggnad av diagram) anser vi att du når godkänt.

Referenser
1. Larman C., Applying UML and Patterns 3rd Ed, 2005, ISBN: 0131489062