Lisa Westlund
Caroline Forslund
Johanna Szepanski

PEER REVIEW av Marco Villegas, Isabel Estungs och Mattias Pavics projekt

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

Verbose List
- Personnummer fattas
- Antal båtar ska inte vara med

Redigera medlem
Väljer man att redigera en medlem som inte finns kraschar programmet.

När man redigerar en medlem kan man ändra medlemsid vilket förstör den persistenta lagringen över tid och det finns inget som kontrollerar att eventuella dubbletter skapas. Redigerar man ett medlemsid så att två medlemmar har samma id påverkar detta hanteringen av båtar. Båtarna kommer då tillhöra flera medlemmar. Syns först när man lägger till, redigerar eller ta bort en båt eller när man köra om programmet.

Vore bra att direkt få en medlemslista när man vill redigera en medlem. Som det är nu får man gå tillbaka till en lista och räkna medlemmens position i listan (här kunde medlemsid använts om det hade varit unikt).

Ägarbyte
Ägarbyte av en båt kan ske till ett medlemsnummer som inte finns i listan vilket göra att de finns i listan, men inte tillhör en riktig medlem. Syns först när listan sparats om (Verbose List). Vi gillar funktionen även om den inte fanns med i kravlistan

Skapa medlem
Man kan skapa medlem utan namn. Det gör att listan förskjuts och fel värden hamnar på fel plats. Om man lämnar vissa fält tomma kan det vara en bra idé att i textfilen ha blankrader. Syns först när listan sparats om (Verbose List).

Bra saker
Tydligt gränssnitt och användarvänlig felhantering. Bra att man kommer tillbaka till huvudmenyn och inte fastnar.

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?

Vore bra om det fanns en README-fil med instruktioner om vilken fil som ska köras. Lite krångligt för oss Mac-nördar att förstå.

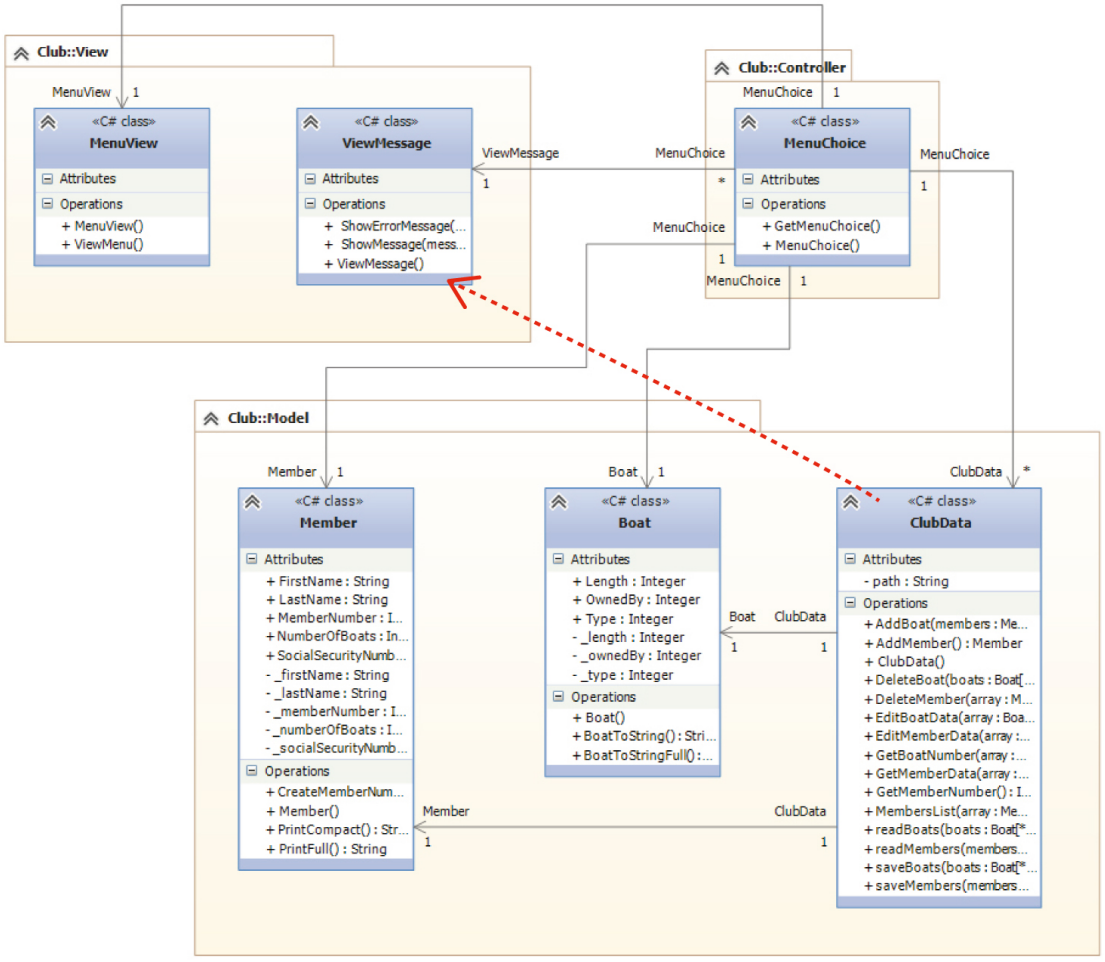
Does the implementation and diagrams conform (do they show the same thing)? Are there any missing relations? Relations in the wrong direction? Wrong relations? Correct UML notation?

Kommentarer på klassdiagrammet (se bild nedan). Vi tror dock inte att ni ska fokusera på att få klassdiagrammet att matcha er nuvarande kod utan strukturera om koden först.

Dependency ska visas som streckade linjer, inte heldragna. Vad vi kan se har ni inga associations (visas som heldragna linjer), utan ni har bara dependencies. Pilarna bör alltså vara streckade. Se Larman [1, p250, ch16.1].

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], där det menas att hjälpmetoder kan exkluderas. Ni behöver inte visa allt, speciellt inte när ni har så många metoder i klasserna.

Vi förstår inte riktigt multiplicitetssiffrorna, då t.ex. ClubData kan hantera många båtar.



Sekvensdiagram

Ni hade bara behövt gör sekvensdiagram för en input och en output. Sekvensdiagrammen är fel skapade. Rutorna ska innehålla klasser och pilarna ska illustrera metoder. Se exempel av Larman [1, p224, ch15.1].

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?

Vi upplever inte att koden är separerad enligt MV principen (inte heller MVC). Vi listar några exempel nedan. Känns lite som att ni har rört ihop begreppen och vilka klasser som ska ha vilka ansvar.

Modellen ska inte känna till eller använda en View. T ex har ni instansierat ett nytt objekt av typen ViewMessage i ClubData och kallar på en av dess metoder och skickar med strängar från model. Något som har med gränssnittet att göra och borde skötas helt och hållet i en View. Se Larman [1, p209, ch13.7].

På ett flertal ställen formateras strängar i modellklasser innan de returneras. T ex PrintCompact() och PrintFull i Membermodellen. Vilket också är en views ansvar.

I Member och Boat har ni hårdkodat in initieringsvärden i konstruktorn. Objekten borde få sina värden via argument till konstruktorerna och inte via Properties. Är detta testvärden som ni glömt att ta bort?

Känns som det händer väldigt mycket i controllerns enda metod. En uppdelning i mindre metoder eller fler controllers skulle göra koden lättare att förstå.

Controllern (Menuchoice) borde också få ett större ansvar och vara som en hantlangare mellan view och model. Ta emot medlemsinfo från viewn, skapa ett medlemsobjekt i Membermodellen(via controllern) och skicka Memberobjektet vidare till ClubData (via controllern). Klassen ClubData ser vi som DALklass.

Metoden addMember innehåller ReadLine som läser input från användaren och därför bör ligga i en view.

Is the requirement of a unique member id correctly done?

Metoden createMemberNumber i Membermodellen skapar ett randomiserat medlemsid. Vilket är helt ok om man också kontrollerar att två medlemmar inte kan få samma id. Se tidigare kommentarer. Id skulle kunna användas mer för att underlätta operationer för användaren.

What is the quality of the implementation/source code?

Code Standards
Snyggt formaterad kod, riktiga getters och setters i modellerna.

Naming
Metoderna börjar ibland med stor bokstav, ibland med liten. Annars lätt att förstå vad metoder gör och variabler är. Bra kommentarer kanske lite i överkant med tanke på att er namnsättning är så pass enkel att förstå.

Duplication
Med tanke på att ni använder medlemmens position i en array snarare än dess medlemsid får ni väldigt komplicerade och långa kodstycken. Finns säkert utrymme för mindre hjälpmetoder framförallt i ClubData. T ex hade metoden readBoatsPosByMember() kunnat strykas.

Dead Code
GetBoatNumber har 0 referenser i klassen ClubData

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

Objects are connected using associations and not with keys/ids.
Inga associationer bara dependencies.

Is GRASP used correctly?
Creator pattern: Några av de avgörande faktorerna för vilken klass som ska skapa ett objekt är dels vilken klass som har information som objektet behöver, och om en klass använder objektet mycket. När ni skapar en Member så gör ni det i er ClubData. Just nu är det den klass som har mest information om en medlem, men den klass som bör få information om en medlem bör dock vara en view, med tanke på att det är i en view information ska läsas in. Å andra sidan har er ClubBoat mycket användning av ett Memberobjekt, men den skulle kunna få tillgång till Memberobjektet när det behövs genom argument till metoder.
Se Larman [1, p282, ch17.8].

Classes have high cohesion and are not too large or have too much responsibility.
Klasserna skulle med fördel kunna delas upp i mindre bitar. Framförallt controllern. Mindre hjälpmetoder för att göra koden lättare att förstå.

Classes have low coupling and are not too connected to other entities.
Metoderna är väldigt specifika och är svåra att återanvända. Klasserna borde kunna hållas mer generella och på så sätt vara lättare att återanvända. Det gäller även metoderna.

Avoid the use of static variables or operations as well as global variables.
Inga statiska variabler eller globala variabler.

Avoid hidden dependencies.
Vi har inte kunnat upptäcka några.

Information should be encapsulated.
Alla klasser ligger i namespaces och Member och Boat har sina field inkapslade i Properties.

Inspired from the Domain Model.
Ja det är den på så sätt att en medlem äger en båt.

Primitive data types that should really be classes (painted types)
Skapandet av ett unikt medlemsid, med tillhörande domänregler kunde ha lagts i en egen klass.

As a developer would the diagrams help you and why/why not?
Klassdiagrammet hade hjälpt, men styr för mycket då det är för detaljerat.

What are the strong points of the design/implementation, what do you think is really good and why?
Bra färgkodning av gränssnittet, bra felmeddelande som egentligen inte behövs enligt kraven.

What are the weaknesses of the design/implementation, what do you think should be changed and why?
Det största problemet är att koden inte är strukturerad enligt model-view-principen. Se kommentarer ovan.

Do you think the design/implementation has passed the grade 2 criteria?
Tyvärr inte. Se ovanstående svar. Vi gissar att ni har kämpat hårt för att få textfilen att fungera på rätt sätt då vi hade samma problem. Det slutade med att vi skrotade textfilen och använde xml istället. Vips blev livet lättare.

Referenser

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