Peer Review Workshop 2

Grupp: Marco Villegas, Isabel Estung & Mattias Pavic.
Grupp att granska: Jan Tran (jt222ic)

Applikationen startar (Yachtclub.exe-filen) men inga funktioner fungerar. När vi går igenom samtliga delar av menyn får vi ett meddelande som säger "Något gick fel".

Att få igång applikationen genom instruktionerna är inga problem. Tydliga och enkla instruktioner.

Vi hade först svårt att öppna klassdiagrammen, men insåg efter en stund att vi först måste öppna projektet i Visual Studio och sedan titta på diagrammen. För en programmerare/utvecklare är det enkelt att förstå att man behöver göra så här för att öppna diagrammen. För en annan person kan detta vara mycket svårare.
Vi hade kanske också då behövt instruktioner till vad man ska fylla i när man skapar en medlem.

Klassdiagrammet är korrekt. Är exakt så som applikationen ser ut. Du skulle kunna skala ned det lite och inte visa exakt alla metoder. Dock så syns inga relationer klasserna emellan. Därför är det svårt att veta om relationerna är rätt eller fel. Larman skriver att UML inkluderar klassdiagram för att illustrera klasser, utseende och deras associationer [Larman kap.16]. I klassdiagrammet vi har granskat att det inte finns några associationer. Enligt Larnman [kap.16.4] ska en navigationspil peka från källan till målet.

MVC-designen ser överlag väldigt bra ut. Finns några rader kod vi inte gillar i klasserna MemberDAL och User. Kod som borde ligga i "View" då de använder sig av "Console.Writeline()" (Rad 67 i MemberDAL och rad 62 i User). [Se Larman, kap.13.7]

Kravet om att "MemberID" ska vara unikt uppfylls tyvärr inte. Det gick att lägga till medlemmar med samma ID.

Kodstandarden är bra och följer en och samma linje. Namngivningen på klasser, metoder och egenskaper är väldigt bra. Enkelt att förstå vad en metod gör utan att kolla på koden inuti den metoden. Vi ser inga direkta duplikationer i koden. Man kan ev. skapa en metod som renderar ut "Console.Writeline()" så slipper man skriva ut den raden hela tiden.

Finns lite död kod i applikationen. Metoden Add() i Boat.cs. Metoderna Add() och addNumber() i Member.cs. Metoden ShowMember() i Console.cs.

Designen använder sig av GRASP på ett bra sätt. Klasserna innehåller för det mesta lagom mycket kod. Det som finns i varje enskild klass hör till den klassen och bör inte vara i någon annan klass. Klasserna använder alltid egenskaperna så: "high-cohesion".

Klasserna har "low coupling". De är väldigt självständiga och är inte beroende av några andra klasser. Designen innehåller heller inte några statiska variabler. Vi har inte heller kunnat hitta några dolda beroenden i designen.

Alla klasser ligger inkapslade i "namespaces" och deras fält är inkapslade inuti egenskaper vilket är tydligt och bra.

Designen är helt klart inspirerad av domänmodellen. Har använt sig av "Member" och "Boat" från modellen på ett liknande sätt i designen.

Vi tycker att sekvensdiagrammet beskriver applikationen på ett bra sätt. Man ser kopplingen mellan klasserna och vad som händer när man vill administrera en medlem eller båt.

Vi tycker att designen och implementationen i nuläget inte klarar kraven för betyg två, men om du skapar en fungerade .exe-fil så bör den göra det. Den följer GRASP bra ("high cohesion" och "low coupling" bland annat), bra MVC-fördelning/arkitekturen är bra. Ren kod att läsa som är lätt att förstå. Det negativa är såklart den icke fungerande .exe-filen. Vi tror dock detta bara är ett litet fel och är enkelt att lösa, eftersom själva programmet fungerar när man kör koden i Visual Studio (vi har testat). Det är dock viktigt att denna ändring sker så att man kan köra programmet enkelt.

Referenser:

Craig Larman, Applying UML and Patterns 3rd Edition, 2004, ISBN: 0-13-148906-2.