Efter att ha testat, kontrollerat och granskat ert program skapat för att uppnå kraven för godkänt i workshop 2, kommer jag ge mina synpunkter och förslag, enligt de instruktioner som delgivits för denna granskning.

Utan några problem kunde jag kompilera och starta programmet genom “GUI.java”, detta gjorde jag genom programmet Eclipse. Instruktionen för att använda programmet var bra, vilket gjorde det lätt att förstå hur jag som användare skulle göra. Något som var ambitiöst var sökfunktionen för att lista medlemmar/båtar, vilket är ett krav för betyg 3. Det enda som jag tyckte var lite missvisande var “inloggningen”, vilket var mer ett val mellan gå in på olika medlemmar eller skapa en ny medlem.

En bugg att notera är att när användaren skapar ny medlem (konto enligt er) och matar in ett inkorrekt personnummer krashar programmet.

När det kommer till klass diagrammet, tycker jag att det inte riktigt stämmer överens med implementationen i vissa delar. Den saknas relationer som t.ex. mellan “Registry“ till “Boat”, “GUI” till “Boat” och “Member”, “Authentication” till “Registry”. Även skrivs inte alla attribut och metoder ut enligt implementationen, när det ändå har gjorts ett försök till ett noggrant diagram.
Personligen tycker jag det är en rörig design och arkitektur då det används statiska typer i klasserna. Detta medför svårigheter till urskilja om en “model view” sepration uppfylls. En sak som är bra är att ingen av Boat och Member klasserna beroende av de andra “controller” / ”view” klasserna på något sätt, enligt Larman C -“The Model-View Separatio principle states that model (domain) objects should not have direct knowledge of view (UI) objects, at least as view objects.”[Ch 13.7].
Dock finner jag att “GUI”, “authentication” och “Registry” hamnar inom “view” paketet, där “authentication” och “Registry” på något sätt är hjälp-klasser. Det är för rörigt och hade hjälpt att dela upp dessa i paket och även vissa detta i klass diagrammet.

Kvalitén på koden är överlag bra, dock finns det fall för duplicering av kod och lite missvisade namngivning. Duplicering av kod sker bland annat vid skrivning och uppdatering av txt-filer och även kontroll av personnummer. Gällande namngivningen, som jag nämnt lite tidigare, är klassen “Authertication”. Jag tycker att den är missvisande då den både kontrollerar id vid “inloggning” och skapar ny medlem.
Något som sticker i mina ögon är användningen av statiska variabler och metoder. Många fall där de nödvändigtvis inte behöver vara statiska.

Enligt min åsikt både design och arkitekttur är rörig hjälper klass diagrammet inte till så mycket för en utvecklare, då den har många dolda beroende vilket är ett av kraven. Sekvens diagrammen ser bättre ut och ger en utvecklare inblick hur en sekvens för de olika use casen skulle se ut.

Det är många punkter i kraven som inte uppfylls för att betyg 2 skall nås. Här är de punkter som bör ses över:

10. Strict Model-View separation (The model should not depend on the view or user interface in any way (direct or indirect) the user interface (view) should not implement domain functionality)
11. Good quality of code (for example naming, standards, duplication)
12. An object oriented design and implementation. This includes but is not limited to:
    - Classes have high cohesion and are not too large or have too much responsibility.
    - Avoid the use of static variables or operations as well as global variables.
    - Avoid hidden dependencies.
    - Informations should be encapsulated.
    - Use a natural design, the domain model should inspire the design.

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