# Planering

Till början var planen att ha ett dict med all data i. Som är uppkomsten till klassen JsonHandling, och har två ändamål; att läsa och att skriva till en given fil. JsonHandling är designad att inte vara beroende på en någon annan klass. För att lösa "Hotell" problemet delades uppgifterna upp i två grupper; front-end och back-end. Dvs det visuella till användaren (front-end) och det som sker bakom kulisserna (back-end). Det tas hand om med HotelManager (back-end) och HotelInterface (front-end). Vi gjorde till början en pseudokod som skulle ge oss enklare översikt till projektet. Därav liknar det inte det slutgiltiga projektet. De metoder som förväntades implementeras var följande (med förväntade indata som är intilliggandes):

* Vid registrering
  + SSN (personnummer)
  + Namn
  + Ålder
* Vid avregistrering
  + SSN
* Vid redigerande av registrering
  + SSN
  + Namn
  + Ålder
  + Nytt SSN
* Vid bokning
  + SSN
  + Rum nummer
  + Meddelande till hotellet
* Vid att ta bort bokning
  + SSN
  + Avregistrera användare (Sant eller falskt)
* Vid redigering av bokning
  + SSN
  + Nytt rum (valfritt)
  + Nytt meddelande (valfritt)
* Vid incheckning
  + SSN
* Vid utcheckning
  + SSN
* Vid nytt rum
  + Namn
  + Pris
  + Plats (antal personer ett rums totala kapacitet tar emot)
  + State (om det är ledigt eller inte)
  + Beskrivning (kort beskrivning)
  + Övrigt (övrig info, såsom internet, tv)
* Vid redigering av rum
  + Namn
  + Pris
  + Plats (antal personer ett rums totala kapacitet tar emot)
  + State (om det är ledigt eller inte)
  + Beskrivning (kort beskrivning)
  + Övrigt (övrig info, såsom internet, tv)
* Vid borttagning av rum
  + Rum nummer

Utdata som förväntas skriva till filen, för att spara undan:

* Användare (Alla registrerade användare)
  + SSN
    - Namn
    - Ålder
* Rum (Alla rum på hotellet, indexen representerar rumsnumret)
  + Namn
  + Pris
  + Plats (antal personer ett rums totala kapacitet tar emot)
  + State (om det är ledigt eller inte)
  + Beskrivning (kort beskrivning)
  + Övrigt (övrig info, såsom internet, tv)
  + User (vem som bokat, om det är bokat annars är det tomt)
  + Meddelande (meddelande från den bokade, om det är bokat annars är det tomt)
* Bokningar (Har koll på vilka som är bokade)
  + SSN
    - Rum nummer
    - Incheckad
* Avregistreringar (Uppdateras när man avregistrerar sig)
  + SSN
    - Namn
    - Ålder
    - Totala registreringar

## Pseudokod för Hotell bokning

- Programstart

- Importerar bibliotek

- Definierar klassen DataHandling med attributen Filnamn och Path

- Kollar om Filen existerar, om inte skapar ny Fil

- Definierar två metoder för att läsa och skriva data till filen

- Definierar klassen HotelManager, som hanterar data och bokningar via diverse metoder

- Definierar Parent-klassen HotellInterface

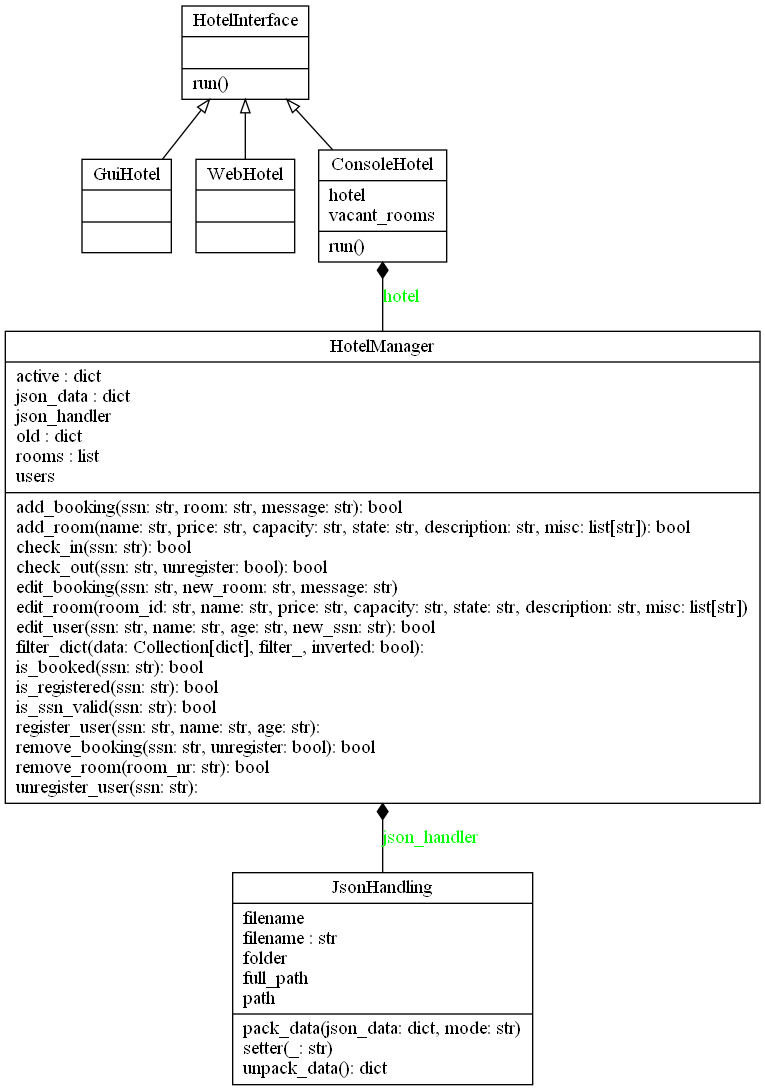
- Definierar klasserna KonsolHotell, GuiHotell och WebHotell som ärver från HotellInterface

- Initierar Hotel objekt

## Aktivitets diagram / Flödesschema:

## 

# Klassdiagram:



# Resultat

Det största som vi inte fick med från planeringen var ett grafiskt gränssnitt, pga dålig tidsplanering. Vi fick däremot med de andra delar vi planerade att få med nämligen klasser för att filhantering, hotellsystemet och även "Frontend" som talar med systemet. Vi gjorde även en möjlig implementation av ett Webb gränssnitt som vi inte hade planerat att lägga till från början, vi gjorde inte helt klart det men det hade varit relativt enkelt att lägga till det, men pga dålig tidsplanering hann vi inte.

# Utvecklingsmöjligheter

Det som skulle kunnat förbättras för både användare och oss själva är ett grafiskt gränssnitt, som hade gjort en enkel och visuell navigering för användaren. Vi hade kunnat arbeta mera på flödesschemat och detaljera processerna mera, vi lade mer fokus på ett mer enkelt förståeligt och grundligt flödesschema. När vi kodade programmet var vi lite för tids optimistiska och lade ner mycket fokus på ett välfungerande och enkelt navigerbar konsol gränssnitt, det blev därför mindre tid till att implementera andra typer av gränssnitt. Vi hade alltså kunnat lagt ner mer tid på andra arbetsområden och planerat arbetet bättre.

# Reflektion

Det gick mestadels som vi hade tänkt oss, arbetsprocessen flöt på som vi tänkte och vi lärde oss nya saker på vägen. Vi stötte på problem med filhantering och hur det ska skötas, därmed utvecklade vi våra kunskaper inom Json och filhantering. Vi lärde oss mycket om filhantering, datastrukturer och fann även en praktisk användning av abstrakta klasser. Den abstrakta klassens funktionalitet var att beskriva hur man talade med hotelsystemet ("Hotelmanager") vilket förenklade framtida implementeringar av diverse gränssnitt. Vi blev nöjda med hur Konsol gränssnitten blev i slutändan, den är enkel att navigera och har submenyer för att förenkla för användare.