**Grönt = Klart**

* **Grupparbete:** Att man blir underkänd om man inte kan visa att man deltagit aktivt är naturligtvis uppenbart, men man kan också bli underkänd om man gör en alltför stor andel av uppgiften själv. Flera av nedanstående punkter blir helt eller delvis meningslösa i alltför små grupper.
* **Versionshantering:** Allt arbete ska versionshanteras. Detta gäller kod, test, byggscript, dokumentation och slutrapporten. Från versionshanteringshistoriken ska det tydligt framgå att alla deltagit i arbetet. Vid redovisningen måste versionhanteringssystemet vara tillgängligt, och salen vi kommer att vara i är (troligtvis) inte utrustad med en stationär dator. Om ni inte använder DSVs Subversion-repository måste därför någon i gruppen ha med sig en dator.
* **Enhetstestramverk:** Ett enhetstestramverk i stil med JUnit ska användas. Ni får gärna använda andra typer av testverktyg också om ni tycker de tillför något, men detta är inget krav.
* **Byggscript:** Ett byggscript, till exempel Ant, ska kunna användas för att bygga och testa systemet. **– ERIK is on it**
* **Ytterligare verktyg:** Ni får gärna testa på att använda andra typer av verktyg, till exempel byggservrar eller ärendehanteringssystem, men detta är inget krav.
* **Testdriven utveckling:** Ni ska tillämpa testdriven utveckling under projektet, och detta ska gå att följa i versionshistoriken.
* **Testfallsdesign:** Ni ska välja ut några lämpliga delar av projektet och tillämpa minst två av de tre testdesigntekniker som tas upp i boken: ekvivalensklassuppdelning, beslutstabeller och tillståndsmaskiner. De testfall som tas fram på dessa sätt kan utgöra grunden för utveckling av dessa delar med TDD eller tas fram i efterhand för att komplettera de testfall som ni redan har tagit fram genom att tillämpa testdriven utveckling.
* **Granskning:**Ni ska genomföra minst en formell granskning av koden. Testa gärna mer informella typer av granskningar också så att ni har något att jämföra med. Ett tips är att prata med någon annan grupp och granska varandras kod.
* **Kodkritiksystem:** Efter att ni genomfört granskningen och rättat de saker som kommit fram så ska ni testa minst ett kodkritiksystem, till exempel Findbugs, på koden. Detta ska göras både på koden som den ser ut då, och på koden som den såg ut innan ni granskade den. Vilka typer av fel hittar verktyget? Är det samma fel som ni hittade?
* **Statiska mått:** När ni är klara med projektet så ska ni använde ett verktyg i stil med Metrics 1.3 för att ta fram statiska mått på koden. Dessa ska redovisas i slutrapporten tillsammans med de slutsatser ni drar av dem. Det räcker alltså inte med att bara upprepa den information som siffrorna själva ger, utan ni måste kunna tolka dem också.
* **Dynamiska mått:** Ett minimikrav på era tester är att de uppnår 100% täckningsgrad på programsatserna i koden. Detta ska visas med hjälp av ett verktyg i stil med Emma. Testa gärna även att försöka uppnå högre nivåer av täckningsgrad på någon mindre del av koden.
* **Profiler:** Slutligen ska ni också testa på att använda en profiler på koden. Här är det lämpligt att ni väljer ut någon intressant del av koden och skriver ett speciellt test eller testprogram som bara är till för att ni ska köra profilern på det. Detta är test blir kanske inte helt realistiskt, men eftersom det inte ingår i uppgiften att implementera ett tillämpningsprogram så får vi göra så.

# Kvar att göra

1, Kör ett kodkritiksystem tex Findbugs. Rätta inte koden!

2, Granskning av koden:

Ni ska genomföra minst en formell granskning av koden. Testa gärna mer informella typer av granskningar också så att ni har något att jämföra med. Ett tips är att prata med någon annan grupp och granska varandras kod.

3, Rätta koden utefter granskningen

4, Kör kodkritiksystem igen.  
 Vilka typer av fel hittar verktyget? Är det samma fel som ni hittade under punkt 1?

**Klar med projektet:**

1, Använd ett verktyg i stil med Metrics 1.3 för att ta fram **statiska mått** på koden.

Dessa ska redovisas i slutrapporten tillsammans med de slutsatser ni drar av dem. Det räcker alltså inte med att bara upprepa den information som siffrorna själva ger, utan ni måste kunna tolka dem också

2, **Dynamiska mått**

Ett minimikrav på era tester är att de uppnår 100% täckningsgrad på programsatserna i koden. Detta ska visas med hjälp av ett verktyg i stil med Emma. Testa gärna även att försöka uppnå högre nivåer av täckningsgrad på någon mindre del av koden

3, **Profiler**

Slutligen ska ni också testa på att använda en profiler på koden. Här är det lämpligt att ni väljer ut någon intressant del av koden och skriver ett speciellt test eller testprogram som bara är till för att ni ska köra profilern på det. Detta är test blir kanske inte helt realistiskt, men eftersom det inte ingår i uppgiften att implementera ett tillämpningsprogram så får vi göra så.