# Kode

**Package**

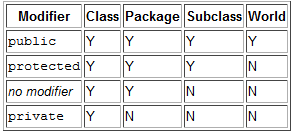
1. Navngivning står med lower camelcase

**Klasser**

1. Navngivning står med uppercamelcase
2. Er acces levels (private, public, protected) brugt så det passer med klassens brug
3. Løser klassen opgaver som ikke hører til dens primære hovedfunktion
4. Ingen:
   1. unødvendige imports
   2. ubrugte metoder
   3. udkommenteret linjer kode som ikke bliver brugt
5. Klassens størrelse er overskuelig (hold den gerne under 150 linjer). Hvis der er for mange metoder, bør klassen deles op.
   1. Den kan evt. deles op i en subklasse
6. Har klassen high cohesion (Høj samhørighed)

**Metoder**

1. Navngivning står med lower camelcase
2. Navngivning på metoderne er tydelig og visende for hvad metodens funktionalitet er
3. Fornuftig brug af service metoder
4. Lad ikke metoder være længere end 10 linjer (gerne hold den på 8)
5. Er acces levels (private, public, protected, default) brugt så det passer med metodens brug



**Variable navne**

1. Navngivning står med lower camelcase
2. Er acces levels (private, public, protected, default) brugt så det passer med variablernes brug

**Generelt**

1. Fornuftig brug af comments.
   1. Ved kode kompliceret kode.
   2. Ingen unødvendige kommentare.
2. Fornuftig brug af linjeskift (fx efter statements og for at skabe overblik i koden)
3. Er der et godt høj-niveau design
   1. arkitektoniske faktorer der især vedrører kobling og samhørighed, herunder (tegn på) anvendelse af designmønstre
4. Er der et godt lav-niveau design
   1. Designmønstre, læsevenlighed osv.

**Test**

1. Er størstedelen af koden dækket af test
   1. prioriter avancerede metoder, hvis alt ikk er dækket
2. I brug af test, bruges mock-klasser fornuftigt