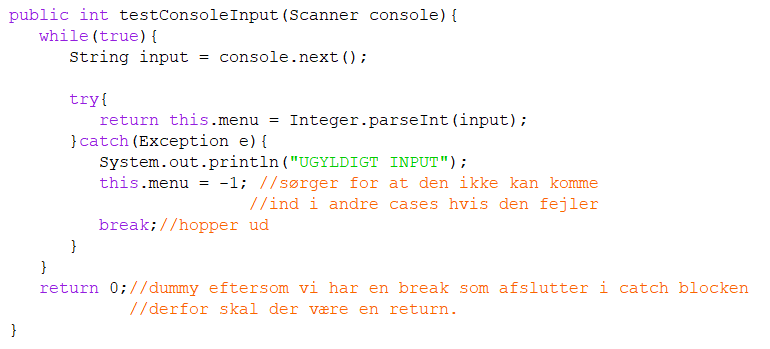
**testConsoleInput():**

Denne metode er lavet til at kontrollere om inputtet er et tal som bruger indtaster.



Kode snippet 1 viser en metode der indeholder try/catch som er en form for error handling. Man kommer altid ind i while loopet eftersom denne er true. Try indeholder den ”normale” kode – den kode som vi forventer går godt.   
Hvis alt går godt, så kører den videre, men hvis ikke kører catch blokken   
Catch hvis der kastes en Exception.  
”Integer” er en klasse nedarvet fra ”Numbers” klassen, der igen er nedarvet fra ”Object” klassen.  
Integer har en masse fields og metoder i sig, som vi gør brug af her.   
Integer.parseInt(input) tager imod en String variabel og omdanne denne til int.

Metoden bliver genbrugt i adskillige sammenhæng hvor det giver mening at bruge det. Et eksempel kan være alder der skal indtastes. Dette er for at undgå redundans.



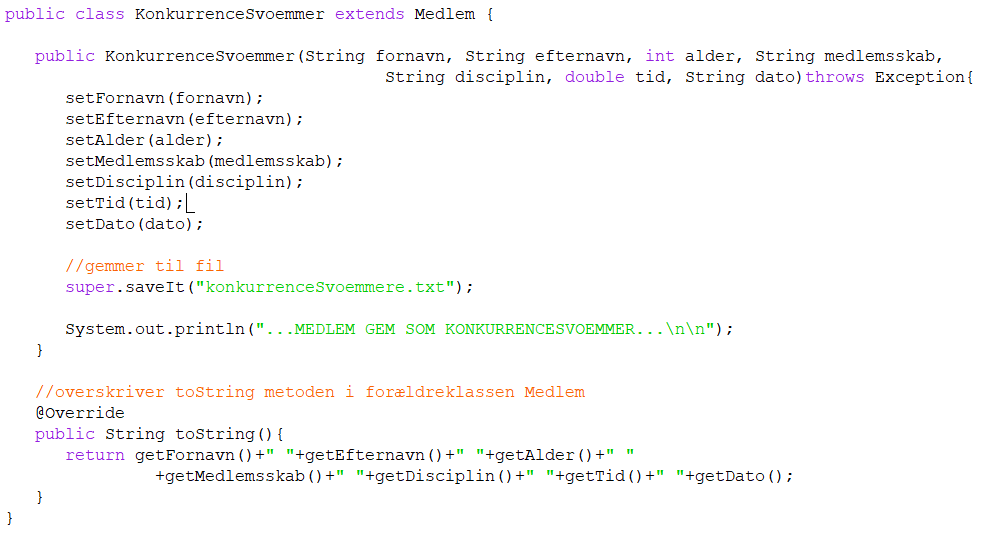
Kode snippet 2 Metoden er implementeret i setter for tid. Det er i stedet for at have redundans og lave endnu en metode som vist i kode snippet 1 for at kontrollere input fra bruger.

**opretMedlem():**

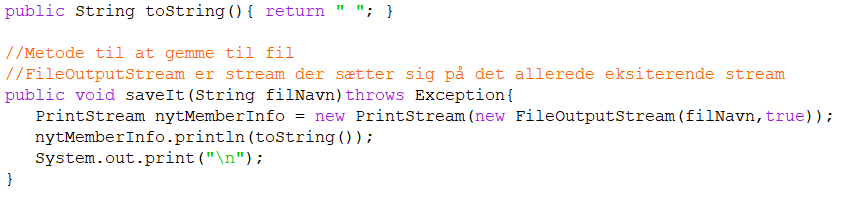
Når man vælger at oprette et medlem, så bliver man spurgt om hvilken slag medlem man vil oprette. Enten motionist eller konkurrencesvømmer. Dette er fordi en konkurrencesvømmer skal have flere informationer tilknyttet sig end motion.   
Så vi har valgt at vælge to metoder der får gemt informationerne om medlem på hver sin måde.   
Vi har en klasse til ”Motionist” og en klasse til ”KonkurrenceSvoemmer”. Dette har vi gjort så vi kan overskrive metoder der er nedarvet fra forældre klassen ”Medlem”, samt have constructorer til hver klasse med forskellige parametre.



Kode snippet 3 Her ses et KonkurrenceSvoemmer objekt, constructoren er kaldt med "new" operator og har alle de værdier som er blevet tastet og sat i setterne, så de kan kaldes i getters i constructorens parameteroverførsel.

Nu er vi i KonkurrenceSvoemmer klassen. Constructoren får overført sin værdier og kalder derefter en metode fra forældre klassen ”Medlem”, ”super.saveIt(…)”.   
Der opnås også polymorphisme ved at man overskriver toString().

Her er vi i forældre klassen ”Medlem”, som har de to metoder der bruges defineret. Det er disse der over skrives.

Så ved at vi får oprettet nogle klasser der har hver deres værdier der skal gemmes, opnås polymorhpisme hvilket er samme kode bare andet output.

Kode snippet 4 saveIt(...) gemmer den nye stream oven på den nuværende stream.

**redigerKontingent():**

I redigerKontingent() kan man redigere et enkelt medlem