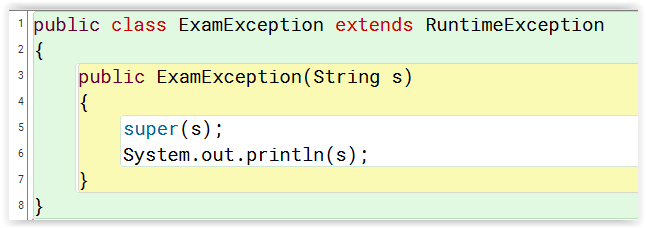
Sensorveiledning, PGR101 V2018

Oppgavene er ment å dekke både kunnskaps-og ferdighetsmål fra emneplanen. Emneplanen skal følge med i «sensurpakken». Alle delemner i emnet er ikke dekket, deriblant JavaFX som det er vanskelig å få til på en god måte i en tretimers skriftlig eksamen med penn og papir… Oppgave 1 og 2 dekker hovedsakelig kunnskapsmål (men med praktisk innslag da de må konkretisere med eksempler). Oppgave 3 og 4 er hovedsakelig ferdighetsmål (med innslag av kunnskap). I denne sensorveiledningen refererer jeg til kapitler i pensumboka. Der er viktig at sensor har denne for å vite hva som kan forventes, særlig rundt kunnskapsspørsmålene. Det følger et Excel-ark med sensorveiledningen. Der kan sensor sette 1-10 poeng for hver deloppgave. Det vil gi en samlet prosentvis score. Scoren er utgangspunkt for en karakter, men alle besvarelser må også sees på i sin helhet, særlig i grensetilfeller.

# Oppgave 3

Løsningsforslag:



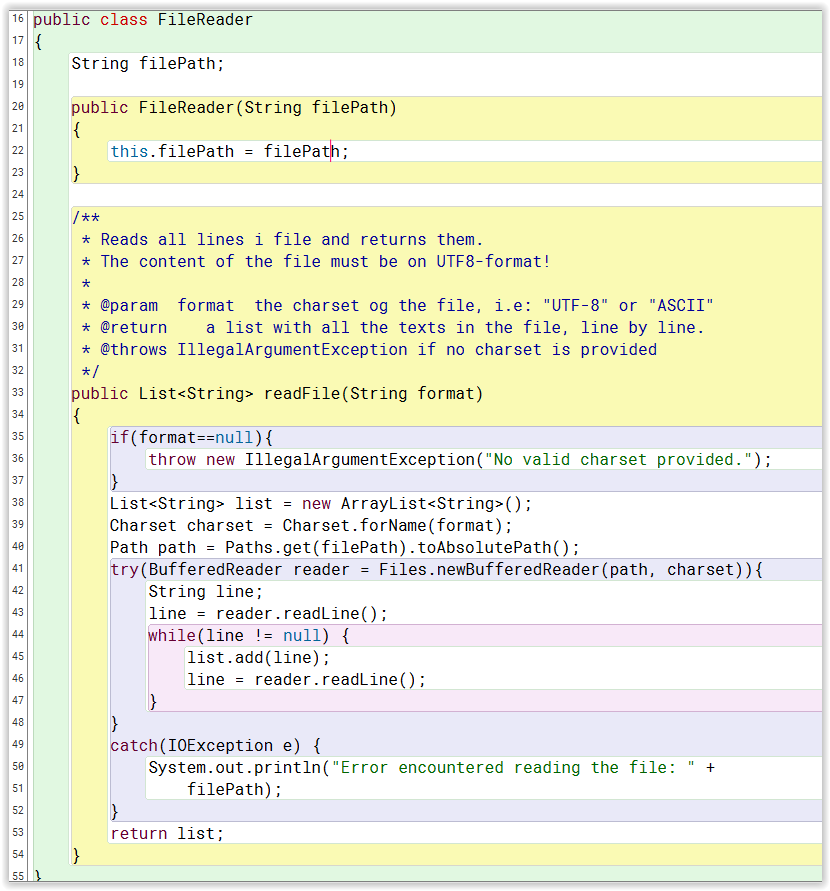
Det er ikke mye kode som skal til, men de må skjønne at de skal:

* Lage en klasse som representerer unntaket.
* Arve fra RuntimeException.
* Benytte super sin konstruktør (som første kodelinje i konstruktøren).

Derfor dekker oppgaven flere delemner i emneplanen (unntakshåndtering, arv, kall på superklassens konstruktør).

# Oppgave 4

Løsningsforslag: (importer er utelatt)



1. Løsning på oppgave A vises ikke av løsningsforslaget. Løsningen er enkel: legge til throws IOException etter metodens signatur. Fint hvis JavaDoc oppdateres med @throws.
2. I sin løsning skal studentene vise til at de skjønner hvordan de skal fange et unntak. MEN: Hva de skal gjøre i catch-clausen er jo et interessant spørsmål. I løsningsforslaget skrives det til sysout, noe som ikke er en veldig elegant løsning (men helt ok på dette nivået). Kanskje noen vil forsøke å loggføre unntaket på et vis? Legg også merke til at det kan kastes unntak fra kodelinje 39 og 40 i løsningsforslaget (noe de kan finne fra vedlegg 3). Det er bare positivt at studentene tar høyde for dette også, men det kreves ikke av oppgaven da den konkret omhandler å unngå kompileringsproblemene.
3. I løsningsforslaget er det valgt å kaste et IllegalArgumentException. Her vil det også kunne komme forslag fra studentene om å returnere null, eller tom liste. Denne problemstillingen skal de kjenne til fra kap 14. Her finnes det ikke et klart fasitsvar, men å returnere null eller tom liste vil skjule at det faktisk skjer et unntak fra normal kjøring. Mange vil påstå at den beste løsningen er å kaste unntaket (som er av type RuntimeException). Da er den foretrukne konvensjonen å ikke presisere at unntaket kan kastets (via throws), men å kommentere det i JavaDoc (slik det er gjort i løsningsforslaget). En fin løsning vil kunne være å ikke gjøre noe som helst, men påpeke at dette automatisk vil skje da IllegalArgumentException nettopp kastes fra Charset.forname (vedlegg 3). Man kan tydeliggjøre dette i koden ved å kunne oppdatere javadoc med @throws med de Runtime-unntakene som kastes fra Paths og Charset.
4. Den foretrukne løsningen er å benytte «Try-with-resoruces» slik det er gjort i løsningsforslaget. En annen (eldre) variant vil være å kalle på close-metoden i BufferedReader i en finally-block.