**Oppgave 1**  
**Lag deg en oversikt over hva følgende ord/begreper betyr:**

* **Refaktorere**

Å refaktorere kode vil si at man rekonstruerer/omstrukturer/endrer på kode man allerede har skrevet. Refaktorering er til for å forbedre attributter som er uten noen funksjon i en software. Det er en måte å forbedre kvaliteten på en programkode og hensikten er å gjøre koden enklere og dermed lettere å lese, vedlikeholde og videreutvikle, samt gi bedre struktur/orden i kode-filen. Refaktorering gjør altså koden kortere men den skal fremdeles ha samme funksjonalitet, derfor gjøres refaktorering som regel i flere trinn.

* **Static (variabel, metode)**

Nøkkelordet static lar oss ha variabler og metoder i en klasse uten å måtte instansiere et objekt før vi kan bruke de. Statiske variabler tilhører klassen og ikke en enkel instans. Vi har både statiske feltvariabler og statiske metoder. I statiske feltvariabler har alle instanser tilgagn til den samme verdien. Verken statiske variabler eller statiske metoder knyttet til en spesifikk instans, men metodene har kun tilgang til statiske feltvariabler og har ingen «this» referanse.

* **Abstract (klasse, metode)**

En abstrakt klasse er en klasse vi ikke kan instansiere et objekt av.

Det kan være klasser vi faktisk ønsker at ikke skal kunne gjøre det, for eksempel klasser vi lager for å abstrahere egenskaper og informasjon

Når vi har to klasser med noe til felles, så kan vi legge det som er felles i en egen klasse som de to klassene kan arve fra.

Problemet da kan bli at man nå kan opprette rene instanser av denne nye klassen. Altså instanser som ikke tilhører noen av de to første klassene, men kun inneholder verdier for felles-klassen.

Dette kan vi unngå ved å legge til nøkkelordet abstract i 'hodet' på felles-klassen. Altså "public abstract class FellesKlasse {".

En abstrakt klasse kan også inneholde abstrakte metoder som er metoder som kun er definert i «hodet» og ikke har noen «kropp». Abstrakte metoder kan ikke lages i vanlige klasser da klassen også må være abstrakt. Når andre klasser arver fra en abstrakt klasse med abstrakte metoder, må arve-klassen implementere de abstrakte metodene og lage logikk/«kroppen».

* **Interface**

Et interface definerer en kontrakt og har ingen implementasjon, men kan derimot implementeres av klasser. Det kan gjøres ved å bruke nøkkelordet ''implements''. Den sier da at den lager en implementasjon i henhold til kontrakten.

Et interface kan minne om en abstract klasse, fordi den kan også inneholde abstrakte metoder. Et interface kan faktisk inneholde kun abstrakte metoder, ikke kontrete metoder.

Et interface derimot kan ikke arves men må implementeres. En klasse kan kun arve fra en annen klasse, men kan derimot implementere flere interfacer.

**Oppgave 2**  
**Gå sammen to og to (en du ikke har samarbeidet med). Ta for dere forrige oblig og forklar deres implementasjon.   
Fokuser på følgende metoder (kan gjerne se på mer om dere ønsker også):**

* **leggTilEpisode(Episode episoden)**
* **oppdaterGjennomsnittligSpilletid()**
* **hentEpisoderISesong(int sesongNummer)**

**Hva har dere gjort? Hvorfor har dere gjort det slik? Hva er forskjellig? Skriv et lite avsnitt om refleksjoner og funn.  
Skriv hvem dere har gått sammen med, men skriv hver deres tekst.**

Jeg gikk sammen med Marcus Johansen på denne oppgaven.

I leggTilEpisode(Episode episoden)-metodene våre har vi benyttet oss av en svært lik metode, men jeg har derimot en ekstra if-setning som sjekker om episoden sitt sesongNummer er akkurat 1 høyere enn antallSesonger, i så fall oppdateres antallet sesonger.   
I oppdaterGjennomsnittligSpilletid()-metodene våre har vi litt forskjeller, men det er kun i hvordan vii har formulert «hodet» i for-setningen vår. Selve koden/«kroppen» gjør det samme bare med ulike variabel navn. En forskjell derimot er at jeg har benyttet ‘’episoder.size()’’ for å få ut antallet episoder som jeg har i arrayet ‘’episoder’’, mens Marcus derimot teller opp antallet episoder i metoden.   
I hentEpisoderISesong(int sesongNummer)-metodene våre har jeg faktisk valgt å ikke lage en metode da jeg syntes dette var enklere. Jeg har derimot i TvSerie laget en instansvariabel som er med i konstruktøren og har en get-metode. Deretter i Main har jeg en for-løkke for å skrive ut alle episodene i en gitt sesong. Marcus derimot har laget en metode i TvSerie som lager et nytt array kalt ‘’episoderISesong’’. Der han benytter en for-løkke til å gå gjennom arrayet med alle episodene og henter ut de av episodene som matcher/er i den gitte sesongen/sesongNummer.

**Oppgave 3**  
**Gå gjennom oppgavene 4-10 under programmering. Lag et klassediagram over alle klassene med variabler og metoder, samt relasjonene mellom disse klassene.  
Hvis du gjør noe mer enn det som er definert i programmeringsoppgaven underveis, løser noe med ekstra metoder eller lignende, lag et modifisert klassediagram i tillegg som reflekterer dette.   
Legg med klassediagrammet(ene) i teoridokumentet.**

