Almuthanna Jamal Aljajan

Oblig 4

Teori

Oppgave 1 - Ord og begreper

Lag deg en oversikt over hva følgende ord/begreper/teknologier betyr/er:

- **Javalin**

  Javalin er et enkelt web-rammeverk for Java og Koltin, fordelen med å bruke Javalin at det er lett kode for å få kommunikasjon mellom backend og frontend så det gir oss mulighet til å skrive vår backend i Java og å lage frontend ved å bruke Vue.js som er et javascript-rammeverk. I tillegg kjører Javalin på Jetty som er en egen webserver for Java, så ved å bruke javalin får vi vår egen lokale webserver. For å kunne bruke javalin, trenger vi å definere en instans av Javalin-klassen. dette gjøres ikke ved å bruke en konstruktør fordi Javalin bruker builder-pattern som vil si at det har static metoder som gir oss en instans av objektet. Det gjør også at vi kan kalle forsklige metoder etter hverandre og likevel får tilbake instans av javlin klasse. Foreksempel vi kan kalle Javalin.create() som gir oss et objekt av Javalin , så kaller vi start() som starter webserveren. Når vi har en instans av Javalin-klassen, må vi definere path-er mot webserveren vår og definerer også hva som skal skje når vi får request på disse pathene. Dette gjøres ved å kalle get() metode som tar to parametre. Den først paremeter som vi definerer i form av String og den andre paremeter (tar Handler i form av Interface så vi kan lage ananym klasse for i mplementere dette interfacet)er en handling det som bli gjort mot den pathen og det kan være enten en ananyom klasse som implementere Handler Interfacet

- **Vue.js**

  Vue.js er et JavaScript rammeverk, vi bruker det i hovedsakk for å få en relativt enkelt frontend. Det anbefalt av utviklerne bak Javalin fordi Vue gjør det ganske enkelt for å integrere med javalin (mellom backend og frontend). For å få syntax-highlighting for Intellij Ultimate må vi installere vue. Vue er et ekstern bibloteket for å kunne bruke det i prosjektet vårt så må vi sette opp det som dependises i gridle-build, i teelegg trrnger vi Jackson bibloteket som gjør det veldig lett for oss omgjør java-objekter til jason format.for å sende dataene til frontend i form av jason. For å legge oppsett til posjektet vårt, gjør vi dette ved å legge frontend filer i resources mappen. Når vi bruker vue.js, så må definere den vue-mappe som javalin vil lette etter den. Når vi bruker javalin og vue sammen , kreves det også en layout.html fil som en template til hvor komponenter og innhold legges. Til slutt lager vi vue-filer for sidene som vi vil vise ved hjelp av vue, html og CSS. Vue- filene består av tre deler i hovedsak. I template del, legger vi alle html og vue kode for hva som skal vises. I script delen, kommer selve vue koden, her definerer vi komponet-navn som skal brukes for å binde vue sammen med javalin.

- **Anonym Klasse**

Det er en type nestet klasse. Det brukes til for å gjøre koden mer konsis. Det Kan deklarere (lage "blueprinten") OG instansiere klassen samtidig og det brukes gjerne i forbindelse med å lage en implementasjon av et interface.

- **MVC (konseptet, og hver enkelt del)**

  Det er en måte å dele logikken i programmet vårt i forsilge lag. MVC står for Model View Controller

  Model er det som representerer dataene som behandles i programmet

  View er grensesnittet altså det brukeren ser, den er typisk presentasjon av data

  Controller det vil si kodelogikken i programmet som binder og setter opp kommunikasjon mellom mode og view

  - **Model**

    Model representerer data i applikasjonen vår, (så alle de fleste klassene vi har lagt for å representere data for eksempel Dyr,Fly og Student er Model klasser. En model klasse innholder bare informasjon om modellen, og det vil si at modellen vet ingenting om hvordan presenteres i grensesnittet.

  - **View**

    View er grensesnittet , altså det brukeren ser, i vårt programmet er vue-filer. View'et vet ingenting om modellen, eller logikken i programmet . Den er bare vet hvordan skal presentere og vise datene som får i jason-format . view Kan gi beskjed når brukeren gjør noe foreksempel om at en knapp er blitt trykket på, men vet ikke hva som skal gjøres når det skjer. Det er controlleren på en måtte har ansvar for hva som skal skje eller hvordan skal behandle det . For oss, er dette .fxml filene . Vi kunne teknisk satt opp hele viewet ved hjelp av kode hvis vi ville .fxml gjør det enkelt for oss

  - **Controller**

    Controller Kommuniserer mellom Model og View. Controlleren henter data fra modellen og sender det videre til viewet og hvor i viewet det skal vises. Controlleren får beskjed fra view-et når noe i viewet blir klikket på/eller handlinger utført og gjør noe basert på disse handlingene. Controlleren har også ansvaret for å oppdatere modellen og vise ny data i viewet

    Den eneste klassen i systemet som vet om «alle» klassene

Oppgave 2 – Kodesammenligning

Oppgave 2.1

I oppgave 2.1, har jeg definert en ny klasse som heter CeletialBody. deretter flyttet jeg alle instanse variabelen (name, radius og mass) som er felles mellom Planet og Star til CeletialBody.I celetialBody definert jeg get og set metoder for disse variabelene. I tillegg definert jeg en konstruktør som tar disse variabelen som parameter. etterpå lar jeg Planet og Star arver fra CeletialBody og når jeg har gjørt det fikk jeg en feil i konstruktør for begge klassene. den feilen var på grunn av arv. fordi når vi definerer et objekt fra sub-klasse, så egentlig definerer vi et objekt også fra super klasse. for å løse dette problemet, har vi to alternativer enten definerer vi en tøm konstruktør i super-Klasse (CeletialBody) eller kaller vi super-konstruktør i konstruktørene som finnes i Planet og Star og det er som jeg gjørt.

Oppgave 2.2

For å finne en Planet basert på navn, definert jeg en metode som retunerer en Planet og tar String (NavnPaPlanet) som parameter.Så brukte jeg vanlige for-løkke for irretere gjennom planet listen og i hver rounde henter jeg navn på planet fra listen og sjekker den med navn som vi får som parameter, så hvis de er like så retunerer vi Planet, hvis ikke retunerer null.

Oppgave 2.3

Jeg har definert følgene variabelen i Planet klasse som konstanter

```
private static final double MASS_IN_MJUP = 1.898E27;
private static final double RADIUS_IN_RJUP = 71492;
private static final double GRAVITATIONAL_CONSTANT = 0.00000000006674;
```

og disse I Star klasse

```
private static final double MASS_IN_MSUN = 1.98892E30;
private static final double RADIUS_IN_RSUN = 695700;
```

grunnen til at jeg har definert slik er at disse variablene hører til klassen, men ikke til spesifikk instans av klassen og i tillegg de har fast verdier. Deretter jeg definerte bare get metoder for disse variablene, fordi disse variablene er definert som final så vi kan ikke endre verdien til dem etter at det fått det.

Oppgave 2.4

I denne oppgave definert jeg en ny Klassen som heter NaturalSatellite, og inni den definerte jeg disse variablene slik:

```
private double semiMajorAxis;
private double eccentricity;
private int orbitalPeriod;
private CelestialBody centralCelestialBody;
```

så definerte jeg get og set metoder for disse variablene og en konstruktør som tar det som parametere. I tillegg lar jeg NaturalSatellite arver fra CeletialBody og Planet arver fra NaturalSatellite og på grunn av arv måtte jeg endre konstruktørene som finnes i NaturalSatellite og Planet klassene. Så måtte jeg definere instans variablene som finnes i

CeletialBody i konstruktøren til NaturalSatellite og deretter kalt jeg super-konstruktør. For konstruktøren til Planeten jeg definerte variabelen som finnes i CeletialBody og NaturalSatellite som parametere og deretter kalte også super-Konstruktør. Og i Main Klasse måtte jeg passe de nye verdiene til objektene som vi har definert der.

Oppgave 2.5

Jeg har definerte metode distanceToCentralBody i NaturalSatellite som skal retunere en double verdi og skal tar degress av typen double som parameter.

Inni metode, definerte jeg variabelen angelInRadians som skal hole verdien til grader i Radian, så jeg konverte grader som vi får til radian ved help av metode Math.*toRadians*(degrees). Deretter definerte jeg vraibelen distance av typen av double som skal holde verdien til formelen etter jeg har konverte det til java-kode ved hjelp av metodene Math.pow() og Math.cos().

Til slutt jeg har ganget resultatene vi får fra variabelen distance med *AVSTANDEN_IN_AU* (149597871) for å få distance in Km så retunertete dette verdi.

```
public double distanceToCentralBody(double degrees) {
    double angelInRadians = Math.toRadians(degrees);
    double distance = ( this.getSemiMajorAxis() * ( 1 - (
Math.pow(this.getEccentricity(), 2) ) ) / ( 1 + ( ( this.getEccentricity() ) * (
Math.cos(angelInRadians) ) ) ) );
    double distanceInKm = distance * AVSTANDEN_IN_AU;
    return distanceInKm;
}
```

I Main-klasse kalte jeg dette metode gjennom earth objektet og jeg passet de forskilge grader som står i oppgaven. I tillegg brukte jeg String.format() metode for å spesifere hvor mange skal etter desimal-point for å få samme resultat som står i oppgaven

```
System.out.println( String.format("%s has a distance of %.0f km to the %s at 0
degrees",earth.getName(),earth.distanceToCentralBody(0),earth.getCentralCelestialB
ody().getName()));
```

Oppgave 2.6

Jeg har definerte metode orbitingVelocity i NaturalSatellite som skal retunere en double verdi og skal tar degress av typen double som parameter.

Inni metode, jeg kalte metode distanceToCentralBody og jeg har sendet grader paramet til den, deretter jeg defiert variabelen speed av typen av double som skal holde verdien til formelen etter jeg har konverte det til java-kode ved hjelp av metodene Math.sqrt().

Til slutt retunertete verdien til speed.

```
public double orbitingVelocity(double degrees) {
    double distance = this.distanceToCentralBody(degrees);
    double speed = ( Math.sqrt(( Planet.getGravitationalConstant()*
```

```
this.getCentralCelestialBody().getMass() ) / distance * CONVERT_KM_TIL_M))*
0.000001;
    return speed;
}
```

I Main-klasse kalte jeg dette metode gjennom earth objektet og jeg passet de forskilge grader som står i oppgaven. I tillegg brukte jeg String.format() metode for å spesifere hvor mange skal etter desimal-point for å få samme resultat som står i oppgaven

```
System.out.println( String.format("At a distance of %.0f, %s has a velocity of
%.2fkm/s",earth.distanceToCentralBody(0),earth.getName(),earth.orbitingVelocity(
0)));
```

Oppgave 2.7

Jeg har *Override* metoden *toString()* i klasser CeletialBody, NaturalSatellite

Jeg har gått sammen med Sadaq Dhiblawe. egentlig det var ikke så mye forskjellen mellom koden min og hans:

den eneste forskjellen at jeg har definerte CeletialBody som en abstract klass, men han ikke gjort det . jeg definerte sånn fordi den klasse finnes i programmet for arv og vi skal ikke definere noe objekter fra den klasse.Jeg synes om par-programmering er en god ide, og det er nyttig fordi vi kan dele kunnskapen og erfaringen mellom oss og programmere med mer effekt måte.