

Oblig 1 – Titas Palijanskas

Oppgave 1.1:

JRE er en programvare ment for sluttbrukere som lar dem kjøre Java applikasjoner på enhetene deres, mens JDK er en utviklingssett laget for utviklere slik at de kan både skape, kompilere og kjøre Java applikasjoner.

Oppgave 1.2:

Prosessen for å bygge og kjøre en java-applikasjon begynner med å gjøre endringer til filen, så skriver inn terminalkommando «javac «programnavn».java», da vil programmet kompileres i bytekode slik at JVM kan tolke det. Etter det skal man skrive «java «programnavn».java» for å så kjøre den, og da vil man eventuelt få output dersom man har satt opp for at programmet gjør slikt.

Oppgave 1.3:

Compile-time feil oppstår ved kompileringen, altså da javac-kommandoen blir kjørt, og da kan man oftest forvente at det er enten feil i syntaks eller semantikken til programmet man forsøker å kompilere.

Run-time feil oppstår når feil skjer i selve koden etter at den har blitt kompilert, et eksempel for dette kan være det å dele noe på null, som er en typisk matematisk feil innenfor programmering.

Oppgave 1.4:

En klasse er en måte å bygge objekter slik at de arver variabler og verdier som er definert innenfor oppgitte parametere, eller utfører gitte funksjoner som er definert innenfor klassen.

Et objekt er da en form for en variabel som kan uttrykke flere variabler og utføre funksjoner som ble uttrykt i klassen den er opprinnelig fra.

Chatlogger:

Oppgave 1.1

Gjeldende teorioppgave: Forklar hva JRE (Java Runtime Environment) og JDK (Java Development Kit) er. ▼

Mitt svar: JRE er en programvare ment for sluttbrukere som lar dem kjøre Java applikasjoner på enhetene deres, mens...

P Programmering 2 Chat-Assistent • Tinkintas „Gem“ ⋮

Hei! Det er hyggelig å hilse på deg. Jeg er her for å hjelpe deg gjennom kurset i Programmering 2.

Dette er en god start på forklaringen din! Du har fanget opp hovedforskjellen mellom hvem de er ment for (sluttbrukere vs. utviklere) og hva deres primære funksjoner er.

Oppgave 1.2:

Gjeldende teorioppgave: Forklar flyten av hvordan java-applikasjoner blir bygget og kjørt. ▼

Knytt også forklaringen din opp mot terminal-kommandoene, javac og java.

Mitt svar: Prosessen for å bygge og kjøre enn java-...


P Programmering 2 Chat-Assistent • Tinkintas „Gem“ ⋮

Dette er en god beskrivelse av selve arbeidsflyten! Du har forstått rekkefølgen i prosessen og rollene til de ulike verktøyene. For å sikre at svaret ditt er teknisk helt presist i henhold til pensum i Programmering 2, er det et par viktige detaljer vi bør se nærmere på.

Oppgave 1.3:

Gjeldende teorioppgave: Forklar forskjellen på Compile-time og Run-time errors. Gi gjerne noen eksempler.

Mitt svar: Compile-time feil oppstår ved kompileringen, altså da
javac-kommandoen blir kjørt, og da kan man oftest...

 **Programmering 2 Chat-Assistent** • Tinkintas „Gem“


⋮

Dette er en veldig god og presis forklaring! Du skiller korrekt mellom når feilene oppstår og hvilken type feil som er typiske for hver fase. I et kurs som Programmering 2 er det viktig å kunne kategorisere disse feilene, da det hjelper deg med feilsøking (debugging).

Oppgave 1.4:

Gjeldene teorioppgave: Forklar forskjellen på en klasse og et objekt.



Mitt svar: En klasse er en måte å bygge objekter slik at de arver
variabler og verdier som er definert innenfor oppgitte...

 **Programmering 2 Chat-Assistent** • Tinkintas „Gem“

⋮


Dette er en god start, og du har forstått kjernen: at det finnes en sammenheng mellom en definisjon (klassen) og noe konkret (objektet).

Oppgave 2.2:




Gjeldende oppgave: Gravitasjonen på månen er ca. 16.5 prosent av jordens. Skriv et program som beregner din vekt på månen. Google kan være en god ressurs for å finne ut nøyaktig hvordan man beregner dette :)

Tankeprosess: Begynner med output som ber brukeren skrive inn egen vekt i kilogram, så leser av programmet neste linje, så oppretter vi en variabel for å konvertere streng til integer, men dette legger vi inn i en try catch slik at hvis vi får et tall som ikke er delelig, f.eks. 0, så fanger vi erroren slik at programmet behandler den trygt, ellers så ganger vi bare vekten ut med 0.165 slik 16.5% er og legger det som output.


 **P**

Programmering 2 Chat-Assistent • Tinkintas „Gem“




Hei! Det er flott at du allerede har gjort deg noen gode tanker om hvordan programmet skal fungere. Du har en god oversikt over flyten: input, konvertering med feilhåndtering, beregning og til slutt output.

Dette er koden som følger tankeprosessen:




```
import java.util.Scanner; // Et verktøy for å lese input

public class Oppg2_2 {
    public static void main(String[] args) {...
```

 **P**

Programmering 2 Chat-Assistent • Tinkintas „Gem“



Oppgave 2.3: (Her kombinerer jeg bonusoppgavene)

Tankegang:

Jeg lager en separatkasse der vi deklarerer variablene, så bruker vi en constructor for å få verdiene, samt få tilbake radius og vekten for senere bruk. I main-funksjonen oppretter jeg noen objekter fra klassen planet, deretter lager jeg en planet array slik at objektene kan lagres i ett sted, og det samme gjøres for strengene slik at vi kan hente navnene for output. Deretter setter jeg opp en while-loop som kjører slik at man kan velge handlinger flere ganger ila. programmets kjøretid, så setter jeg opp det slik at man kan legge inn input og basert på input vil man lage en switch med 4 cases som består av det å opprette planet, liste planetene, slette, eller endre. I det å opprette planet skal man lese inn 3 input, navn, radius og masse, radius og masse vil da konverteres til double for at man kan behandle desimaltall. Etter at alt er ferdig så får man input om at planeten har blitt lagt til, samtidig blir planeten lagt til array og telleren for posisjonen til en ledig element blir hevet. Dersom radius eller masse utgir ugyldige tall vil vi gi en numberformatexception error. I det å liste listen vil vi først se om listen er tom er å sjekke om noen av elementene er noe annet en null, ellers blir det lagt opp en for-loop som lister planetene i listen

I det å slette listen vil vi igjen sjekke om listen er tom, og deretter liste planetene. Brukeren vil da taste inn planeten slik at den kan velges, hvis input stemmer med noen av elementene i listen vil den valgte planeten slettes.

I det å endre en verdi i noen av planetene vil vi igjen sjekke om listen er tom, og deretter liste planetene. Da får brukeren velge hva de har lyst til å endre, altså enten navn, radius eller masse. Utifra valget så setter vi opp switch case der vi bare henter planeten som stemmer i listen og endrer verdien dersom den er gyldig og har blitt opprettet før.



Programmering 2 Chat-Assistent • Tinkintask „Gem“

Hei! Det er flott at du har kommet i gang med tankeprosessen rundt oppgaven. Det virker som du har en god oversikt over hvilke komponenter som kreves for å løse denne utfordringen.

Du har beskrevet en logisk flyt som dekker mange av kravene i oppgaveteksten, spesielt når det gjelder bruk av `Scanner`, `while`-løkke og håndtering av de ulike handlingene (opprette, liste, slette, endre).

```
import java.util.Scanner;

public class Oppg2_3 {
    public static void main(String[] args) {
        Scanner read = new Scanner(System.in);...
```

P

Programmering 2 Chat-Assistent · Egendefinert Gem

Dette er et veldig godt utgangspunkt! Du har kontroll på det grunnleggende i Java, og logikken din for `switch`-setninger og feilhåndtering med `try-catch` fungerer bra. Du følger også prinsippet om innkapsling i `Planet`-klassen med `private` felt og tilhørende `get/set`-metoder.

Her er noen observasjoner og forslag til forbedringer basert på koden din:

1. Lagring av planeter (Array vs. ArrayList)

Du bruker en array med fast størrelse: `planets = new Planet[42];`.

- **Begrensning:** Hvis brukeren prøver å legge til planet nummer 43, vil programmet krasje med en `ArrayIndexOutOfBoundsException`.
- **Vilkårlig antall:** Oppgaveteksten ber om at du skal kunne registrere et *vilkårlig antall* planeter. I Java brukes ofte `ArrayList<Planet>` i slike tilfeller, da denne vokser dynamisk etter behov.

2. Gjenbruk av kode (DRY - Don't Repeat Yourself)

Du har identisk kode for å liste ut planeter i både `case "2"`, `case "3"` og `case "4"`.

- **Tips:** Kan du flytte denne logikken ut i en egen metode? For eksempel: `private static void listPlanets(Planet[] planets) { ... }` Dette gjør `main`-metoden din mye kortere og enklere å lese.

Jeg har valgt å la disse forbedringene være i senere tidspunkt, fordi jeg mener at jeg har gjort mer enn nok av en tilfredsstillende jobb for å få et greit system som fyller alle krav. Det finnes alltid forbedringer i en oppgave som dette.

Ressurser:

<https://www.geeksforgeeks.org/java/java-array-empty-check/>

<https://www.geeksforgeeks.org/java/encapsulation-in-java/>

https://www.tutorialspoint.com/java/number_parseint.htm