

DA339A Laboration L3

Syfte

Laborationens syfte är att ge en grundläggande genomgång av hur enklare program skrivs i en textredigerare. Laborationen täcker grundläggande koncepten variabler, tilldelning, operationer, String och input från användare, detta skapar en helhetsbild av hur allt detta kan bygga ett program från grunden.

Följande tas upp i denna labb:

- Kort repetition om Git commit / undo commit
- Variabler
 - Konstanta variabler
- Tilldelning
- Operatorer
- Strängar (String)
- Inläsning från användare (Input)

Redovisning

Laboration L3 ska inte redovisas.

Förberedelser

Den här laborationen är direkt kopplad till det innehåll som finns i föreläsning F2 till och med F3. Utöver detta är det viktigt att följande har gjorts:

- Installerat Visual Studio Code (VS Code) (Laboration 1)
- Installerat JDK (Laboration 2)
 - OBS! Inga hjälpverktyg för Java ska installeras i VS Code, i senare Laborationer kommer detta finnas med. Men i början är det väldigt viktigt att utföra dessa laborationer utan hjälpmedel.
- Om inte gjort enligt instruktion i Laboration 1, skapa en mapp att spara relevanta filer till för denna laboration.

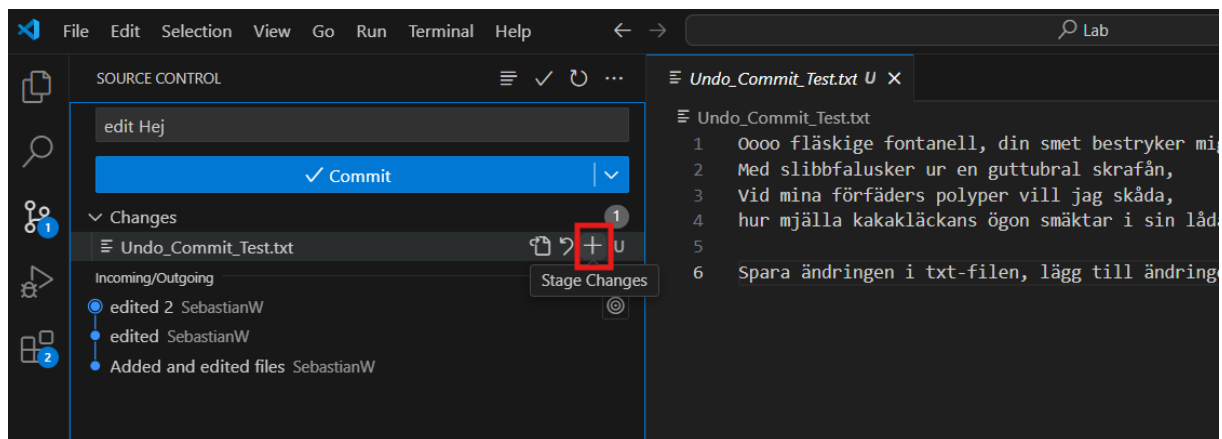
Repetition Git – commit / undo commit

I laboration L1 lärde ni er hur man gjorde en commit i Visual Studio Code med Git. I följande uppgift kommer ni lära er hur man först gör en commit och därefter tar bort/ånga (undo) en commit.

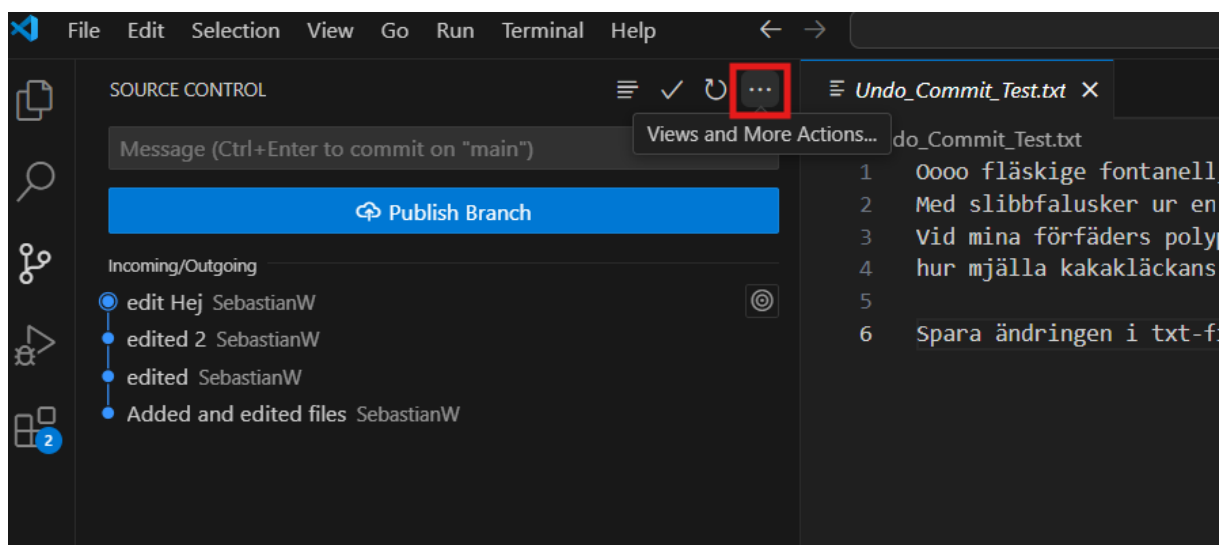
Öppna Visual Studio Code och det repo som ni använde i L1. Lägg till/skapa en ny txt-fil och döp den till `Undo_Commit_Test`. Lägg till texten nedan i txt-filen:

*Oooo fläskige fontanell, din smet bestryker mig.
Med slibbfalusker ur en guttubral skrafån,
Vid mina förfäders polyper vill jag skåda,
hur mjälla kakakläckans ögon smäktar i sin låda*

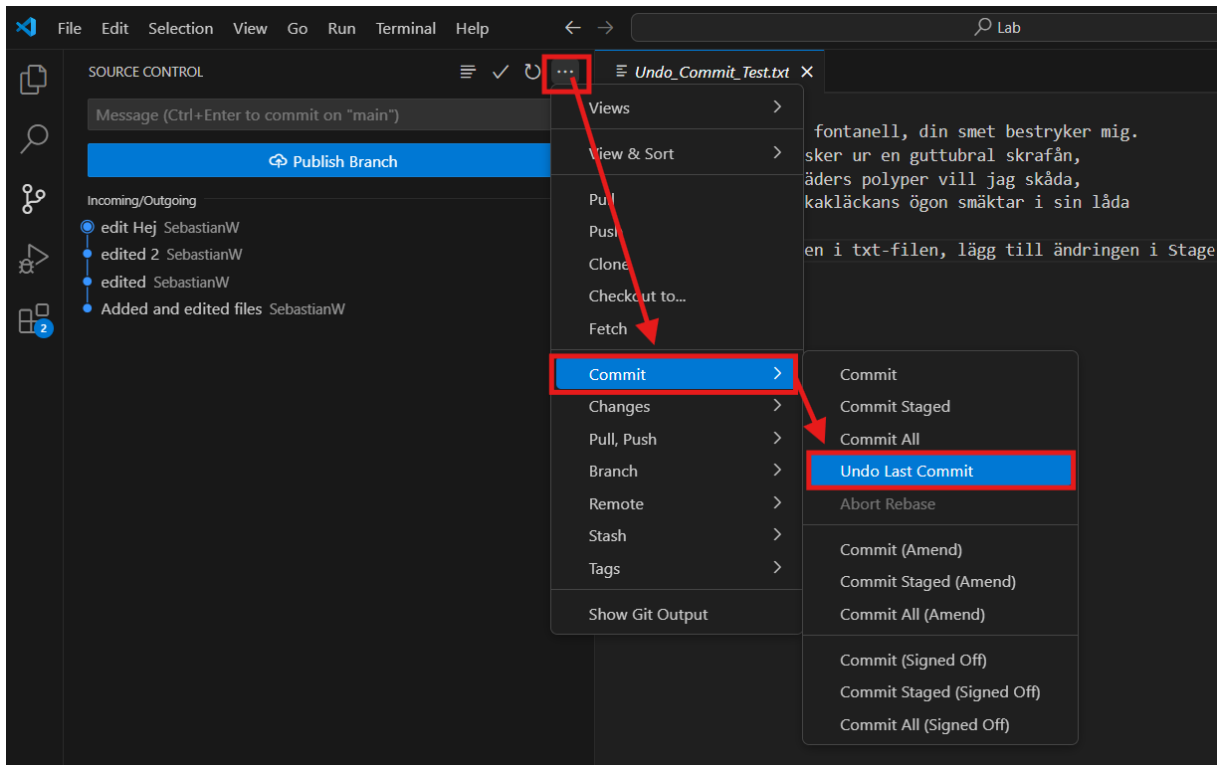
Spara ändringen i txt-filen, lägg till ändringen i Stage Commit genom att trycka på "+"-tecknet bredvid ändringen.



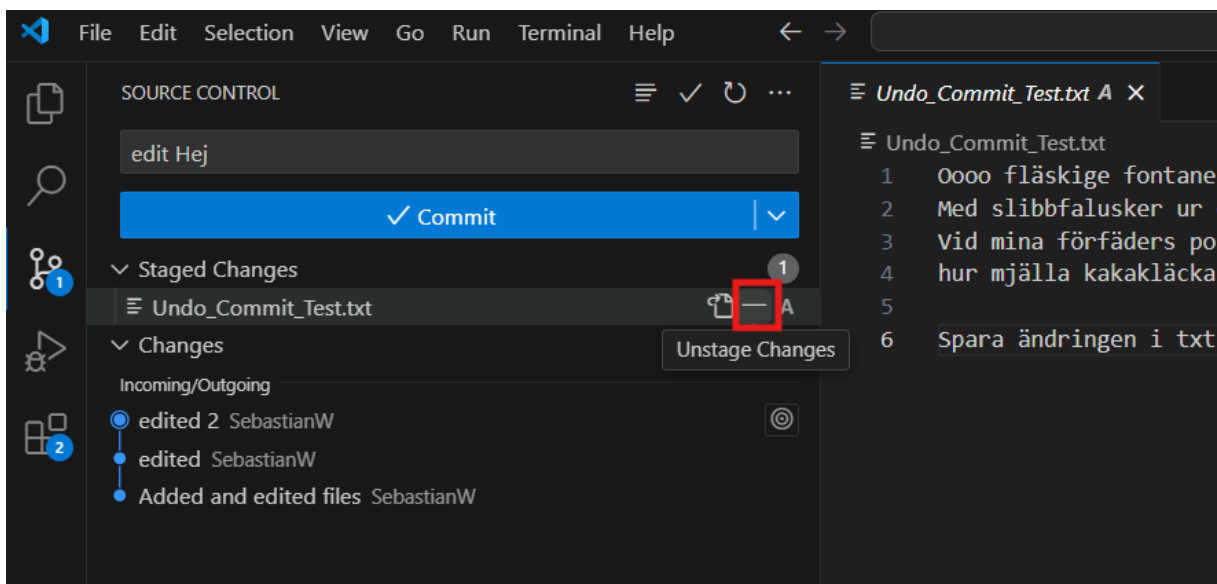
Skriv ett commit meddelande (bestäm meddelandet själv) och tryck på knappen Commit. När commit är färdigt är det dags att ånga (undo) detta commit. Gå till source control (bör redan vara där) och klicka på de tre punkterna enligt bilden nedan.



Orientera igenom menyn tills ni kommer till Undo Last Commit knappen (se bild nedan) och klicka på den.



Efter Undo Last Commit Knappen är tryckt hamnar den senaste commit som gjorts i Stage Commit listan igen. Klicka på ”-”-tecknet för att ta bort er commit från Stage Commit, se bild nedan.



Nu har den senaste commit ångrats och kan ändras eller tas bort. Experimentera lite med de andra commit alternativen (gör detta dock i ett separat test repo ifall något skulle gå fel).

Uppgifter

Följande uppgifter innehåller inga guider för hur kod ska skrivas utan beskriver detta abstrakt. Tanken är att utifrån den information som har getts ut på föreläsningar och i kurslitteratur återskapa korrekt syntax och struktur i uppgifterna. Om det inte går att komma vidare fråga alltid antingen kurskamrater eller labbassistenter innan facit används.

Under laborationen kommer det att arbetas med olika typer av aritmetiska operationer i Java, nedan finns en tabell med en kort beskrivning.

Vanliga operationer

Operation	Betydelse
+	Addera två tal tillsammans
-	Subtrahera ett tal från ett annat
/	Dividera ett tal med ett annat tal
%	Modulus, detta visar resten av en division.

Vanliga datatyper

Datatyp	Betydelse
boolean	Lagrar ett påstående, true eller false
char	Lagrar ett enstaka tecken, till exempel 'a'
double	Lagrar flyttal, exempelvis 56,54. Kan även tolkas som <i>double</i> $\approx \mathbb{R}$ till ca 15 siffrors precision
int	Används för att lagra heltal, exempelvis 3. Kan även tolkas som <i>int</i> $\approx \mathbb{Z}$ i intervallet - 2 147 483 648 till 2 147 483 647
String	Lagrar strängar, exempelvis "Detta är laboration 3!"

OBS! Även om det är bekvämt, försök att skriva all kod på engelska då detta är industristandard, detta är även förväntat i laborationer längre fram. Det enda som inte behöver vara på engelska är text som förekommer inom citattecken "text".

Uppgift 1 – Räkna och jämföra tal

Uppgift 1a

Starta VS Code, när VS Code har startats, skapa en ny fil vilket ska döpas till Calculator.java. Spara denna i mappen Lab3. Följande instruktioner ger sekvensen vilket koden ska skrivas i:

1. Skapa klassen Calculator
2. Skapa main-metoden
3. Skriv ut texten "Välkommen till förmodligen världens bästa kalkylator!"

Efter detta har gjorts kompilera och exekvera filen Calculator, om allt fungerar ska det skrivas ut: Välkommen till förmodligen världens bästa kalkylator!

Uppgift 1b

Nästa steg är att få kalkylatorn att faktiskt utföra en typ av aritmetisk operation. Följande instruktioner ska läggas till i nuvarande klassen Calculator.

1. Dividera heltalet 9 med heltalet 5, lagra detta i variabeln nbr1 vilket ska vara ett heltal.
2. Dividera heltalet 9 med heltalet 5, lagra detta i variabeln nbr2 vilket ska vara ett flyttal.
3. Dividera flyttalet 9.0 med heltalet 5, lagra detta i variabeln nbr3 vilket ska vara ett flyttal.
4. Dividera flyttal 9.0 med flyttalet 5.0, lagra detta i variabeln nbr4 vilket ska vara ett flyttal.
5. Skriv ut följande text "Tal 1, 9/5 ger: " samt konkatenera detta med nbr1
6. Skriv ut följande text "Tal 2, 9/5 ger: " samt konkatenera detta med nbr2
7. Skriv ut följande text "Tal 3, 9/5 ger: " samt konkatenera detta med nbr3
8. Skriv ut följande text "Tal 4, 9/5 ger: " samt konkatenera detta med nbr4

Ifall minnet sviker visar Laboration 2 hur en så kallad sträng-konkatenering görs.

När ovan instruktioner har följts, kompilera och sedan exekvera Calculator. Går det att observera några märkliga beteenden i koden när det kommer till de olika talen? Facit innehåller förklaring.

Uppgift 2 – Temperaturkonvertering

Som visat ovan är variabler väldigt användbara när det ska lagras information av något slag för vidare användning i andra delar av programkoden, nedan följer uppgifter av liknande slag men däremot ska även input från användare användas nu.

Uppgift 2a – Celsius till Fahrenheit

I denna uppgift ska Celsius konverteras till Fahrenheit med hjälp av följande formel:

$$F = \frac{9}{5} \times C + 32$$

Med ovan formel skriv ett program vilket räknar ut F med hjälp av C. Följande bör göras:

1. Skapa klassen *CelsiusToFahrenheit*
2. Skapa main metoden
3. Ange ett välkomstmeddelande, exempelvis *"Detta program konverterar Celsius till Fahrenheit!"*
4. Skapa en lämplig variabel för C och fyll den med ett värde
5. Skapa en lämplig variabel för F och fyll den med ett värde
6. Lagra resultat av konvertering i lämplig variabel
7. Skriv ut resultatet av konverteringen, exempelvis *"X grader Celsius är Y grader Fahrenheit!"*

Tips: Om det blir ett oväntat resultat, fundera på om det sker en heltalsdivision eller flyttalsdivision.

Uppgift 2b – Fahrenheit till Celsius

I denna uppgift ska Fahrenheit konverteras till Celsius med hjälp av följande formel:

$$C = \frac{5}{9} \times (F - 32)$$

Med ovan formel skriv ett program vilket räknar ut C med hjälp av input F från användaren. Följande bör göras:

1. Skapa klassen *FahrenheitToCelsius*
2. Skapa main metoden
3. Ange ett välkomstmeddelande, exempelvis *"Detta program konverterar Fahrenheit till Celsius, ange en temperatur som ska konverteras: "*
4. Ta input från användare
5. Lagra input från användare i lämplig variabel
6. Lagra resultat av konvertering i lämplig variabel
7. Skriv ut resultatet av konverteringen, exempelvis *"X grader Fahrenheit är Y grader Celsius!"*

Tips: Om det blir ett oväntat resultat kontrollera följande:

- Kontrollera ifall divisionen som sker är en heltals- eller flyttalsdivision
- Kontrollera input från användare, är det heltal eller flyttal
- Glöm inte att input från användare kräver klassen Scanner

Uppgift 3 – Formatering av strängar

I vissa program, t.ex. från Uppgift 2b, kan temperaturer skrivas ut med många decimaler, till exempel 0 grader Fahrenheit är -17.7777777777778 grader Celsius. För att göra utskriften mer läsbar utan att ändra innehållet på någon av variablerna kan antalet decimaler begränsas i utskriften.

För att åtgärda ovan problem gör följande:

1. Studera guiden för [printf på Baeldung](#)
2. Direkt under raden med `System.out.println` som skriver ut resultatet, *lägg till en rad med `System.out.printf`.*
3. Formatera utskriften så att *Celsius visas med max två decimaler.*

Uppgift 4 - Biljettprogram

I denna uppgift ska tidigare kunskaper kombineras för att skapa ett program vilket skriver ut ett kvitto till köpta biobiljetter.

Information:

- En vuxenbiljett kostar 100 kr.
- Barnbiljetter kostar 75 % av priset för en vuxenbiljett (25 % rabatt).
- Programmet behöver veta: antal vuxna, antal barn och köparens namn.

Gör så här:

1. Skapa klassen Ticket med en main-metod.
2. Skriv ett välkomstmeddelande.
3. Be användaren ange sitt namn, antal vuxna och antal barn.
4. Beräkna:
 - a. Barnrabatt (25 % av vuxenpris per barn)
 - b. Totalbelopp att betala
5. Skriv ut kvittot med printf, t.ex.: TBD

Tips:

- Använd Scanner för input.
- Multiplicera antal barn med vuxenpris och rabatten för att beräkna barnrabatten.
- ***Totalebelopp = vuxenpris × antal vuxna + barnpris × antalet barn***

Uppgift 5 – Typkonvertering

I uppgift 1 utfördes enkel division på flyt- och heltal, något som kunde observeras i denna uppgift var hur JDK avrundade resultat automatiskt om rätt datatyper inte användes. I denna uppgift ligger fokus på att själv styra vilken datatyp som ska användas, genom så kallad type casting. I Java betraktas överdriven eller felanvänd type casting ofta som en signal på brister i programdesignen, men det är ändå en viktig teknik att känna till och förstå.

Som student är det svårt att ha koll på ett ungefärligt genomsnittsbetyg under studietiden, skapa därav ett program vilket kan göra denna uträkning i stället.

Information:

- Programmet räknar genomsnittsbetyget för en student
- Betyg anges som heltal i intervallet $1 \leq x \leq 5$
- När genomsnittsbetyget beräknas måste resultatet presenteras som ett flyttal
- Minimum av tre kurser, anta att användaren känner till alla dessa

Gör så här:

1. Skapa klassen *TypeConversion* och dess *main-metod*
2. Skriv ett *välkomstmeddelande*
3. *Skapa lämpliga variabler*, glöm ej en variabel vilket lagrar antalet kurser
4. Be användare *ange* sitt namn
5. Be användare *ange* betyg för respektive kurs
6. *Beräkna genomsnittet* av alla betygen
7. Skriv ut studentens *namn* och *genomsnittsbetyg* med två decimaler

Tips:

- Fråga efter varje betyg separat. För de som är mer insatta kan iteration användas.
- Vid beräkning av genomsnitt från heltal behöver typkonvertering anges, exempelvis (double) *det som ska räknas ut*.

Uppgift 6 – Parentes vid aritmetisk operation

I denna uppgift ska [*PEMDAS*](#), som används i matematiken, tillämpas för att skapa en grundläggande förståelse för hur JDK utför aritmetiska operationer.

Information:

- Programmet utför olika aritmetiska operationer
- Flyt- och heltal kan användas, men måste vara av samma typ och tal för att kunna observera skillnader från en beräkning med parentes och en utan.

Gör så här:

1. Skapa klassen *Pemdas* och dess main-metod
2. Skriv ett *välkomstmeddelande*
3. Utför en *aritmetisk operation utan parenteser*, lagra resultat i lämplig variabel
4. Utför en *aritmetisk operation med parenteser*, lagra resultat i lämplig variabel
5. Skriv ut *resultaten*

Tips:

- Reflektera över vad som sker innanför parentesen. Till exempel adderas tal i parentes innan multiplikation.

Exempel:

$$5 + 5 * 3 = 20$$
$$(5 + 5) * 3 = 30$$

Uppgift 7 – Beräkna resten

I vissa programmatiska uppgifter kan det vara bra att kunna avgöra om ett tal har en rest eller inte, exempelvis går detta att använda för att kontrollera ifall ett tal är jämt eller udda. För att beräkna resten används uttrycket *modulo* både inom matematik och programmering.

I denna uppgift ska ett enkelt program skrivas vilket beräknar vad resten av ett tal blir samt avgöra om ett tal är jämnt eller inte.

Information:

- Programmet ska räkna resten av olika divisioner alltså inte kvoten.
- Både flyt och heltal kan användas
- Modulo används inte bara för att ta reda på jämna tal utan även för andra syften, till exempel:
 - Kontrollera delbarhet med ett valfritt tal exempelvis `tal % divisor == 0`
 - Iterera igenom index i en lista (`i % n` för att hålla index inom gränserna)
 - Upprepa mönster, t.ex. färger, veckodagar eller steg i en loop
 - Kryptografi och hash-funktioner

Gör så här:

1. Skapa klassen Modulo och dess main-metod
2. Skriv ett välkomstmeddelande
3. Beräkna resten för ett flertal olika divisioner och spara resultaten i varsina variabler
4. Beräkna om ett tal är jämnt spara resultaten i variabel
5. Beräkna om ett tal är ojämt spara resultaten i variabel
6. Skriv ut resultaten

Tips:

- Se tabellen för modulo-operationen i Java som presenterades tidigare i kursen.
- För att avgöra om ett tal är jämnt används 2 som divisor i modulo, exempelvis `tal % 2 == 0`

Uppgift 8 – Konstanter

Ibland kan uppgifter kräva att data manipuleras med ett återkommande värde, exempelvis vid beräkning av ränta. I dessa situationer kan det vara användbart att använda konstanter. Konstanter fungerar som vanliga variabler men har speciella regler kring värdeändring.

I denna uppgift ska ett enkelt program skrivas som multiplicerar ett valt tal med en konstant, samt experimentering med att ändra konstantens värde för att observera hur programmet reagerar.

Information:

- Välj själv ett nummer som ska multipliceras
- Den konstanta variabeln ska vara 5

Gör så här:

1. Skapa klassen *ConstantTest* och dess *main-metod*
2. Skriv ett *välkomstmeddelande*
3. Beräkna det *egenvalda antalet* med *konstanten* och lagra *resultatet* i en lämplig variabel
4. Skriv ut *resultatet*
5. *Ändra på konstantens värde* och observera vad som händer

Tips:

- Använd nyckelordet *final* för att ange att en variabel ska vara konstant.

Extrauppgifter inget facit

Uppgift 9 – Förbättra kalkylatorn

Skapa en kalkylator som tar input från användaren vid varje beräkning. Tillämpa kunskaper från uppgifter 1–8 för att bygga programmet. Programmet ska kunna hantera flera operationer och visa resultatet direkt.

Uppgift 10 – Förbättra modulatoräknaren

Skapa en modulatoräknare som tar input från användaren, beräknar resten vid division och anger om talet är jämnt eller ojämnt. Använd kunskaper från uppgifter 1–8 för att implementera denna funktion.

Uppgift 11 – Skapa egna övningar med hjälp av AI

I denna uppgift används AI för att generera ytterligare programmeringsövningar. Syftet är att förstärka förståelsen för de koncept som laborationen hittills har täckt, samt att lära sig skapa egna övningar för framtida labbar om fler övningar behövs än de som laborationen erbjuder.

Information:

Laborationen har hittills behandlat: variabler, datatyper, type casting, aritmetiska operationer, parentesers påverkan på prioritet (PEMDAS), modulo, konstanter, användarinput och utskrift med print, println och printf.

Målet är att skapa nya övningar som är relevanta för dessa ämnen.

Gör så här:

1. Formulera en tydlig prompt till AI:n som innehåller:
 - a. Vilket koncept uppgiften ska öva (t.ex. type casting, modulo, PEMDAS eller utskriftsmetoder).
 - b. Krav på input/output, gärna med exempel.
 - c. Om övningen ska använda olika utskriftsmetoder (print, println och/eller printf).
2. Använd AI:n för att generera minst tre nya programmeringsuppgifter baserat på prompten.
3. Testa att skriva koden för minst en av de genererade uppgifterna i Java och kör den.
4. Reflektera kort i labbrapporten:
 - a. Hur väl fungerade prompten?
 - b. Var det något i den genererade uppgiften som behövde justeras?
 - c. Vad lärde du dig av att skapa och lösa uppgifterna själv?

Tips:

- Var tydlig i prompten: inkludera exempel på input/output och specifika utskriftsmetoder om möjligt.
- Tänk på svårighetsnivån – övningarna ska vara genomförbara men utmanande.
- AI kan föreslå lösningar, men förstå alltid logiken innan du accepterar dem.

(exempel på nästa sida)

Exempel:

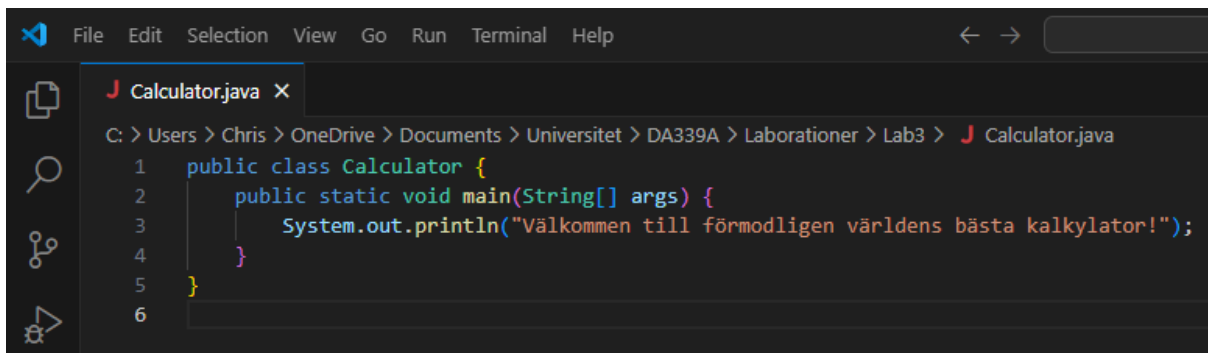
"Skapa tre övningar i Java som övar följande koncept: variabler, datatyper, type casting, aritmetiska operationer, parentesers prioritet (PEMDAS), modulo, konstanter, användarinput och utskriftsmetoder (print, println och printf).

Varje övning ska innehålla:

- En kort beskrivning av uppgiften.
- Information om vilka input som behövs från användaren.
- Exempel på förväntad output.
- Tydlig instruktion om vilken utskriftsmetod som ska användas.
- Generera även korta tips som hjälper att lösa uppgiften, men ge inte hela lösningen."

Facit**Uppgift 1a – Räkna och jämföra tal**

Kod för skapandet av klass, main-metod och utskrift.



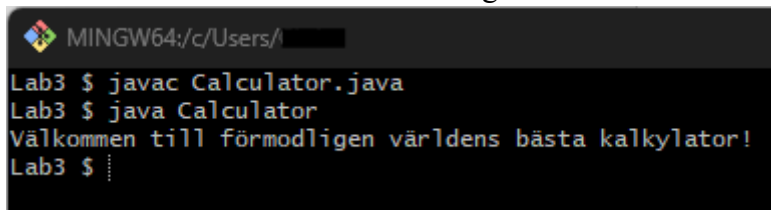
```

1 public class Calculator {
2     public static void main(String[] args) {
3         System.out.println("Välkommen till förmodligen världens bästa kalkylator!");
4     }
5 }
6

```

Figur 1: Calculator-klassen med enkel utskrift

Resultatet av ovan kod vid exekvering finns nedan.



```

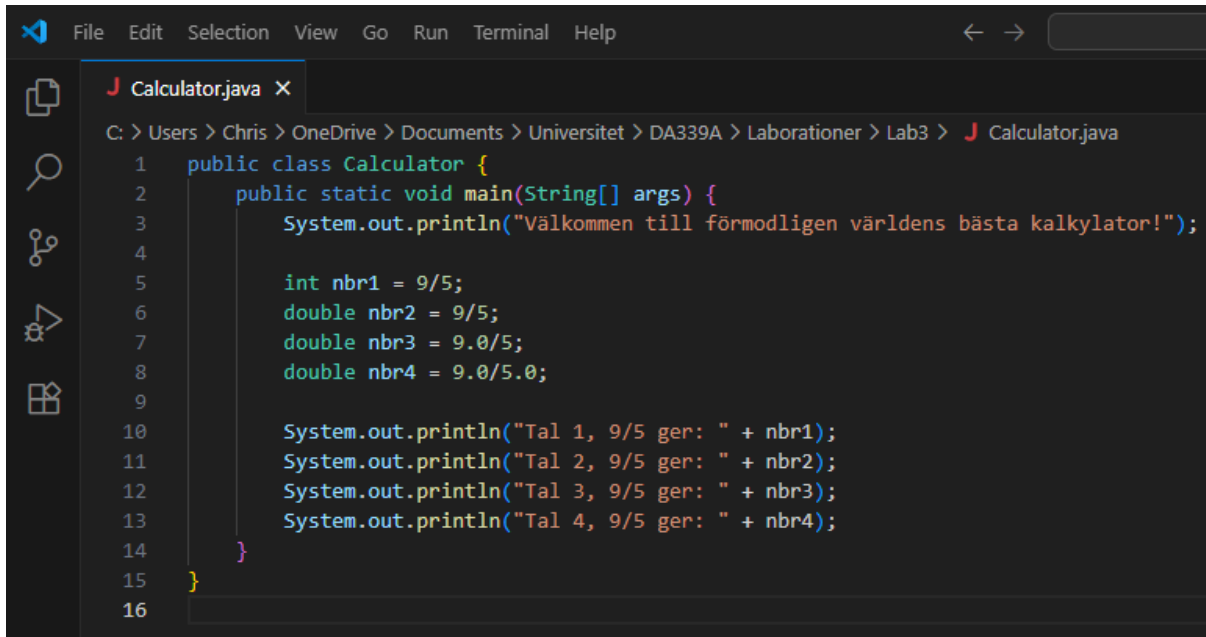
MINGW64:/c/Users/...
Lab3 $ javac Calculator.java
Lab3 $ java Calculator
Välkommen till förmodligen världens bästa kalkylator!
Lab3 $

```

Figur 2: Calculator-klasens resultat i exekvering.

Uppgift 1b – Räkna och jämföra tal

Kod för aritmetiska operationer i Calculator



```

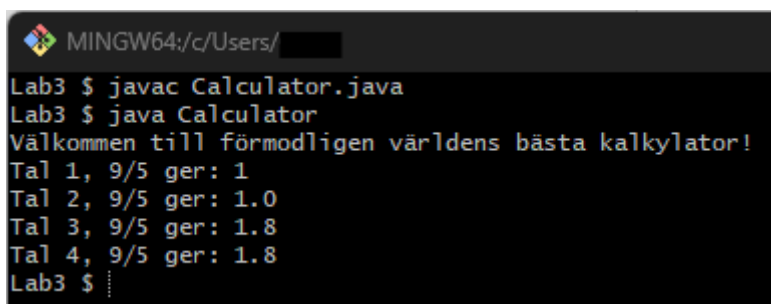
1 public class Calculator {
2     public static void main(String[] args) {
3         System.out.println("Välkommen till förmodligen världens bästa kalkylator!");
4
5         int nbr1 = 9/5;
6         double nbr2 = 9/5;
7         double nbr3 = 9.0/5;
8         double nbr4 = 9.0/5.0;
9
10        System.out.println("Tal 1, 9/5 ger: " + nbr1);
11        System.out.println("Tal 2, 9/5 ger: " + nbr2);
12        System.out.println("Tal 3, 9/5 ger: " + nbr3);
13        System.out.println("Tal 4, 9/5 ger: " + nbr4);
14    }
15 }
16

```

Figur 3: Calculator med aritmetiska operationer.

Som nämnt i 1b uppstår det något märkligt i utskriften av de olika talen i utskriften nedan. Följande kommer en lista över vad som händer:

1. $9/5$ ger 1 eftersom båda operander är heltal (int). Java använder då heltalsdivision, vilket kapar decimalerna (1.8 blir 1). Resultatet sparas i en int.
2. $9/5$ ger fortfarande 1, eftersom båda operander är heltal. Även om resultatet lagras i en double sker beräkningen först som heltalsdivision och decimalerna försvinner innan det konverteras.
3. $9.0/5$ ger 1.8, eftersom en operand är double. Java gör då flyttalsdivision och resultatet blir ett exakt flyttal.
4. $9.0/5.0$ ger 1.8 av samma skäl som i punkt 3 – båda operander är double och flyttalsdivision används.



```

MINGW64:/c/Users/
Lab3 $ javac Calculator.java
Lab3 $ java Calculator
Välkommen till förmodligen världens bästa kalkylator!
Tal 1, 9/5 ger: 1
Tal 2, 9/5 ger: 1.0
Tal 3, 9/5 ger: 1.8
Tal 4, 9/5 ger: 1.8
Lab3 $

```

Figur 4: Utskrift i Git Bash för de olika variablerna.

Uppgift 2a – Celsius till Fahrenheit

```

1 public class CelsiusToFahrenheit {
2     public static void main(String[] args) {
3         System.out.println("Detta program konverterar Celsius till Fahrenheit!");
4         double celsius = 32;
5         double fahrenheit = (9.0/5.0 * celsius) + 32;
6         System.out.println(celsius + " grader Celsius är " + fahrenheit + " grader Fahrenheit!");
7     }
8 }

```

Figur 5: Lösningsexempel på Uppgift 2a.

Nedan följer förklaring av vad som händer, varje nummer i listan motsvarar respektive kodrad.

1. Skapande av klass med nyckelord `class` och klassnamn, med krullparenteser som omsluter all kod som tillhör klassen
2. Skapande av `main`-metod, startpunkt för programmet
3. Utskrift av välkomstmeddelande
4. Deklarering och tilldelning av variabeln `celsius`. Datatypen är `double` eftersom temperaturer ofta hanteras som flyttal
5. Deklarering och tilldelning av variabeln `fahrenheit`. Värdet beräknas med formeln $(9.0/5.0 * celsius) + 32$. $9.0/5.0$ används för att säkerställa flyttalsdivision
6. Utskrift av resultatet med strängkonkatenering, t.ex. "32 grader Celsius är 89.6 grader Fahrenheit!"
7. Slut på `main`-metoden
8. Slut på klassen `CelsiusToFahrenheit`

Uppgift 2b – Fahrenheit till Celsius

```

1  import java.util.Scanner;
2  public class FahrenheitToCelsius {
3      public static void main(String[] args) {
4          Scanner sc = new Scanner(System.in);
5          System.out.print(s:"Detta program konverterar Fahrenheit till Celsius, ange en temperatur som ska konverteras: ");
6          double fahrenheit = sc.nextDouble();
7          double celsius = 5.0/9.0 * (fahrenheit - 32);
8          System.out.println(fahrenheit + " grader Fahrenheit är " + celsius + " grader Celsius!");
9      }
10 }

```

Figur 6: Lösningsexempel på uppgift 2b.

1. Import av Scanner – krävs för att kunna läsa input från användaren. Importen måste stå först, innan klassen, eftersom Java behöver veta vilka externa klasser som används innan själva programmet kompileras.
2. Skapande av klass med nyckelord class och klassnamn, med krullparenteser som omsluter all kod som tillhör klassen
3. Skapande av main-metod, startpunkt för programmet
4. Skapande av Scanner-objekt (sc) för att läsa användarens input.
5. Utskrift av välkomstmeddelande
6. Deklarering av fahrenheit samt tilldelning via input från användare.
7. Deklarering och tilldelning av variabeln celsius. Värdet beräknas med formeln $5.0/9.0 * (fahrenheit - 32)$. $9.0/5.0$ används för att säkerställa flyttalsdivision
8. Utskrift av resultatet med strängkonkatenering, t.ex. "32 grader fahrenheit är 0 grader Celsius!"
9. Slut på main-metoden
10. Slut på klassen FahrenheitToCelsius

Uppgift 3 – Formatering av strängar

```

1  import java.util.Scanner;
2  public class FahrenheitToCelsius {
3      public static void main(String[] args) {
4          Scanner sc = new Scanner(System.in);
5          System.out.print(s:"Detta program konverterar Fahrenheit till Celsius, ange en temperatur som ska konverteras: ");
6          double fahrenheit = sc.nextDouble();
7          double celsius = 5.0/9.0 * (fahrenheit - 32);
8          System.out.println(fahrenheit + " grader Fahrenheit är " + celsius + " grader Celsius!");
9          System.out.printf(format:"%f grader Fahrenheit är %.2f grader Celsius!\n", fahrenheit, celsius);
10 }
11 }

```

Figur 7: Lösningförslag uppgift 3.

1. `%f` anges för att konkatenera in variabeln fahrenheit, `%.2f` används för att konkatenera in variabeln celsius avrundat till två decimaler slutligen används `%n` för att skapa en ny rad

Uppgift 4 - Biljettprogram

```
import java.util.Scanner;
public class Ticket {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double ticketPrice = 100;
        double childDiscount = 0.75;
        String name;
        int adults = 0;
        int children = 0;

        System.out.println(x:"Välkommen till Bion!\nDagens film kostar 100kr per person.\nBarn får alltid 25% rabatt!");
        System.out.print(s:"Ange ditt namn: ");
        name = sc.nextLine();
        System.out.print(s:"Ange antalet vuxna: ");
        adults = sc.nextInt();
        System.out.print(s:"Ange antalet barn: ");
        children = sc.nextInt();

        double appliedDiscount = (children * ticketPrice)-((children*ticketPrice)*childDiscount);
        System.out.println(appliedDiscount);
        double total = (adults*ticketPrice) + (children*ticketPrice)*childDiscount;
        System.out.println(total);
        System.out.printf(format:"Kvittot för %s\nTotalt att betala: %.2f\nAvdragen barnrabatt: %.2f\n", name, total, appliedDiscount);
    }
}
```

Figur 8: Lösningalternativ 1 för Uppgift 4.

```
import java.util.Scanner;
public class Ticket {
    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        double adultPrice = 100;
        double childDiscountRate = 0.25; // 25% rabatt
        String name;
        int adults;
        int children;

        System.out.println("Välkommen till Bion!\nDagens film kostar 100 kr per vuxenbiljett.\nBarn får alltid 25% rabatt!");
        System.out.print("Ange ditt namn: ");
        name = sc.nextLine();
        System.out.print("Ange antalet vuxna: ");
        adults = sc.nextInt();
        System.out.print("Ange antalet barn: ");
        children = sc.nextInt();

        double childDiscount = children * adultPrice * childDiscountRate;
        double total = (adults * adultPrice) + (children * adultPrice * (1 - childDiscountRate));

        System.out.printf("Kvittot för %s\nTotalt att betala: %.2f kr\nAvdragen barnrabatt: %.2f kr\n", name, total, childDiscount);
    }
}
```

Figur 9: Lösningalternativ 2 för Uppgift 4

Använda datatyper:

- double används för priser och rabatt, eftersom decimaler kan förekomma
- int används för antal vuxna och barn, eftersom det alltid är heltal
- String används för namn då det är lämpligast att sparas i en sträng

Rabattberäkning:

- Rabatt kan beräknas som
 - $(adults * ticketPrice) + (children * ticketPrice) * childDiscount$ lösningalternativ 1.
 - $children * adultPrice * 0.25$ eller via `childDiscountRate` variabel, lösningalternativ 2.
- Båda metoderna ger samma resultat, men att lagra rabatten i en egen variabel gör koden mer läsbar.

Totalbelopp:

- $Total = antal\ vuxna \times pris + ((antal\ barn \times pris) \times barnrabatt)$ eller i kod $(adults * ticketPrice) + (children * ticketPrice) * childDiscount$ som visas i Lösningförslag 1
- $Total = vuxenpris \times antal\ vuxna + barnpris$ där $barnpris = vuxenpris \times (1 - rabatt)$ eller i kod $total = (adults * adultPrice) + (children * adultPrice * (1 - childDiscountRate))$ som visas i Lösningförslag 2

printf-användning:

- `%.2f` begränsar antalet decimaler till två.
- `%n` används för ny rad på ett plattformsoberoende sätt.

Ordningen på input:

- Läs namn först med `nextLine()`, sedan heltal med `nextInt()`.
- Om man blandar `nextLine()` och `nextInt()` kan extra `nextLine()` behövas för att undvika att läsa in tom rad.

Läsbarhet:

- Att bryta ut priser och rabatt som egna variabler gör koden tydligare och enklare att ändra.
- Båda lösningalternativen är korrekta; skillnaden är främst hur man räknar ut barnrabatten.

Uppgift 5 – Typkonvertering

```

1  import java.util.Scanner;
2
3  public class TypeConversion {
4      Run | Debug
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7
8          String name;
9          int objectOriented = 0;
10         int introDataScience = 0;
11         int databaseTechnology = 0;
12         int nbrOfCourses = 3;
13         double result = 0;
14
15         System.out.println(x:"Välkommen till Betygskonverteraren, ange betyg från 1 till 5!");
16         System.out.print(s:"Ange ditt namn: ");
17         name = sc.nextLine();
18         System.out.print(s:"Ange betyg för Objektorienterad programmering: ");
19         objectOriented = sc.nextInt();
20         System.out.print(s:"Ange betyg för Introduktion till Datavetenskap: ");
21         introDataScience = sc.nextInt();
22         System.out.print(s:"Ange betyg för Databasteknik: ");
23         databaseTechnology = sc.nextInt();
24
25         result = (double)(objectOriented + introDataScience + databaseTechnology) / nbrOfCourses;
26         System.out.printf(format:"Ditt genomsnittsbetyg %s är: %.2f", name, result);
27     }

```

Figur 10: Lösningförslag för typkonvertering.

Mycket av denna kod är lik tidigare uppgifter i laborationen. Den avgörande raden är rad 23, där type casting används:

$$result = (double)(objectOriented + IntroDataScience + databaseTechnology) / nbrOfCourses$$

Summan av betygen konverteras till double innan divisionen sker. Med detta utförs en flyttalsdivision i stället för heltalsdivision, därav kan resultatet även innehålla decimaler. Hade double utelämnats hade en heltalsdivision utförts, detta innebär att JDK automatiskt hade avrundat nedåt till närmaste heltal innan resultatet sparades i variabeln resultat.

Uppgift 6 – Parentes vid aritmetisk operation

```

1  public class Pemdas {
2      public static void main(String[] args) {
3          int x = 5+5*3;
4          int y = (5+5)*3;
5
6          System.out.println(x:"Välkommen till Pemdasprogrammet");
7          System.out.printf(format:"5+5*2 = %d%n", x);
8          System.out.printf(format:"(5+5)*2 = %d%n", y);
9      }
10 }
11

```

Figur 11: Lösning som använder sig av PEMDAS.

Detta program liknar många av de lösningar som redan har gjorts under laborationen, men det viktiga att fokusera på i denna uppgift är variablerna x och y , där PEMDAS-principen kan observeras.

$$int\ x = 5 + 5 * 3$$

Ovanstående operation följer **M** i PEMDAS, d.v.s. multiplikation. Eftersom multiplikation har högre prioritering än addition, multipliceras först 5 med 3 och därefter adderas 5, vilket leder till summan 20.

$$int\ y = (5 + 5) * 3$$

Ovanstående operation följer **P** i PEMDAS, d.v.s. parentes. Eftersom parenteser har större prioritet än multiplikation adderas först innehållet i parentesen, i detta fall adderas 5 med 5 och sedan multipliceras resultatet med 3 vilket får summan 30.

Uppgift 7 Beräkna resten

```

1  public class Modulo {
    Run | Debug
2      public static void main(String[] args){
3          int x = 5 % 2;
4          double y = 5.0 % 2.5;
5          int z = 6 % 3;
6          double a = 6.0 % 4.5;
7          int even = 14%2;
8          int uneven = 7%2;
9
10         System.out.println(x:"Välkommen till restberäknaren!");
11         System.out.printf(format:"5 mod 2 har resten: %d\n", x);
12         System.out.printf(format:"5.0 mod 2.5 har resten: %.2f\n", y);
13         System.out.printf(format:"6 mod 2 har resten: %d\n", z);
14         System.out.printf(format:"6 mod 4.5 har resten: %.2f\n", a);
15         System.out.printf(format:"Talet 14 är ett jämnt tal då det har resten: %d\n", even);
16         System.out.printf(format:"Talet 7 är ett ojämnt tal då det har resten: %d\n", uneven);
17     }
18 }

```

Figur 12: Beräkning av rest med hjälp av moduloteknet i Java

I svarsförslaget ovan använd modulo-operatorn % för att beräkna resten vid en division.

int x = 5 % 2; → Resten när 5 delas med 2 blir 1

double y = 5.0 % 2.5; → Resten när 5.0 delas med 2.5 blir 0,0

int z = 6 % 3; → Resten när 6 delas med 3 blir 0

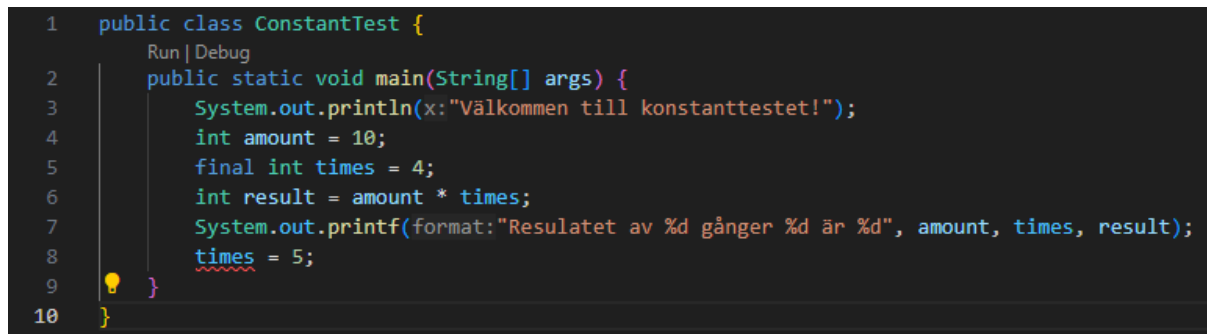
double a = 6.0 % 4.5; → Resten när 6.0 delas med 4.5 blir 1.5

even = 14 % 2; → Resten när 14 delas med 2 blir 0, därav är 14 jämnt

uneven = 7 % 2; → Resten när 7 delas med 2 blir 1, därav är 7 ojämnt

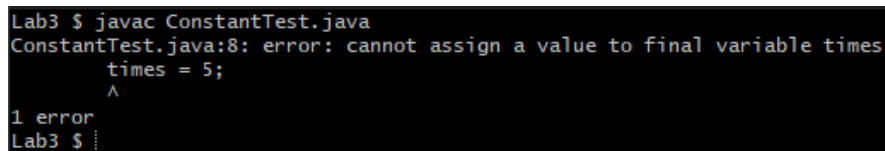
Det som visas i uppgiften är att modulo kan tillämpas på både flyt- och heltal och räknar resten så länge rätt datatyper används. Detta är särskilt användbart i exempelvis logiska kontroller, som när man vill avgöra om ett tal är jämnt eller inte.

Uppgift 8 – Konstanter



```
1 public class ConstantTest {
2     Run | Debug
3     public static void main(String[] args) {
4         System.out.println(x:"Välkommen till konstanttestet!");
5         int amount = 10;
6         final int times = 4;
7         int result = amount * times;
8         System.out.printf(format:"Resulatet av %d gånger %d är %d", amount, times, result);
9         times = 5;
10    }
```

Figur 13: Kod med exempellösning för konstanter.



```
Lab3 $ javac ConstantTest.java
ConstantTest.java:8: error: cannot assign a value to final variable times
    times = 5;
    ^
1 error
Lab3 $
```

Figur 14: Fel vilket uppstår vid tilldelning av en redan satt konstant.

Som kan observeras ovan går det inte att manipulera värdet på en konstant. Om ett försök att tilldela ett nytt värde till en konstantvariabel uppstår det ett kompileringsfel. I detta exempel löses problemet genom att radera den rad som försöker att ändra på konstantens värde.