**Opgaver om datatyper: integer division, type cast og konstanter**

Du skal nu lave nogle tilføjelser til Tid-klassen.

**Integer division**

I Java gør regnearterne addition, subtraktion og multiplikation som forventet, men du kan blive overrasket over hvordan division fungerer. For eksempel i følgende fragment der skal beregne hvor mange brøkdele af en time, der er gået:

int minutter = 59;

System.out.print("Brøkdel af timen, der er gået: ");

System.out.println(minutter / 60);

Outputtet er:

Brøkdel af timen, der er gået: 0

Dette resultat forvirrer ofte folk. Værdien af ​​minut er 59, og 59 divideret med 60 er 0,98333, ikke 0. Problemet er, at Java udfører "integer division", når operanderne er heltal. Integer division runder altid ned til nul, selv i tilfælde som dette, hvor det næste heltal er tæt på. Man kan også beskrive det sådan at alt i et komma-tal efter kommaet bliver smidt væk fra resultatet når man laver integer division.

Som et alternativ kan vi beregne en procentdel i stedet for en brøk:

System.out.print("Procentdel af timen, der er gået: ");

System.out.println(minutter \* 100 / 60);

Det nye output er:

Procentdel af timen, der er gået: 98

Igen er resultatet rundet ned, men nu er det i hvert fald nogenlunde korrekt.

|  |
| --- |
| OPGAVE 1: Udskriv hvor mange procentdele af en time din minutvariabel er. |

**Type cast mellem double og int**

En anden løsning på problemet er at bruge kommatal, som kan repræsentere brøker såvel som heltal.

Java udfører "floating-point division", når en eller flere operander har typen double.

double minutter = 59.0; //Variablen minutter er nu af typen double

System.out.print("Brøkdel af timen, der er gået: ");

System.out.println(minutter / 60);

Outputtet er:

Brøkdel af timen, der er gået: 0,983333333333333333

Selvom doubles er nyttige, kan de være en kilde til forvirring. For eksempel skelner Java imellem int-værdien 1 og double-værdien 1.0, selvom de ser ud til at være det samme tal. De tilhører forskellige datatyper, og som udgangspunkt må du ikke lave tildelinger mellem dem.

Følgende er ulovligt, fordi variablen til venstre er en int og værdien til højre er en double:

int x = 1.1; //compiler fejl

Det er nemt at glemme denne regel, fordi Java i mange tilfælde automatisk konverterer fra en type til en anden. Dette kaldes (implicit) type cast.

double y = 1; //lovlig, men dårlig stil

Det foregående eksempel burde være ulovligt, men Java tillader det ved automatisk at konvertere int-værdien 1 til double-værdien 1.0. Dette er praktisk, men den giver ofte problemer for begyndere. For eksempel:

double y = 1/3; //almindelig fejl

Man kunne måske forvente, at variablen y fik værdien 0.333333. Men i stedet får den værdien 0.0. Udtrykket til højre deler to heltal, så Java laver integer division, hvilket giver int-værdien 0. Omregnet til double bliver værdien der tildeles y til 0.0. Java tilføjer simpelthen bare et 0 efter kommaet!

En måde at løse dette problem på (når du har fundet ud af fejlen) er at gøre højre side til et floating-point udtryk (dvs af typen double). Følgende sætter y til 0,333333, som forventet:

dobbelt y = 1.0 / 3.0; //korrekt

Som udgangspunkt bør du altid tildele floating-point-værdier til floating-point-variabler. Compileren tvinger dig ikke til det, men det er god stil og giver færre fejl.

|  |
| --- |
| OPGAVE 2: Lav dine minut-variabel om til typen double som beskrevet i eksemplet, så du får et kommatal der viser hvor mange brøkdele af en time din minut-variabel er. |

Der er også en anden måde at lave type cast på i Java. Lad os sige at du ikke har særlig meget lyst til at dine minutter skal være af typen double (du har kun tænkt dig at det skal være hele minutter alligevel!). Her er det muligt at lave type cast på en (eller begge) operander, så man midlertidigt konverterer dem til double, selv om de er af typen int. Dette gøre ved at skrive double i parenteser foran den variabel man gerne vil type caste.

int minutter = 59;

System.out.print("Brøkdel af timen, der er gået: ");

System.out.println((double)minutter/60); //minutter er nu double

|  |
| --- |
| OPGAVE 3: Lav dine minut-variablen om til typen double som beskrevet i eksemplet, ved at type caste variablen inde i selve beregningen. |

**Konstanter og literals**

En værdi i et program, f.eks. tallet 60 (eller String’en ”Klaus”), kaldes en literal. Generelt er der ikke noget galt med literals. Men når et tal som 60 optræder i et udtryk uden forklaring, gør det koden svær at læse. Og hvis den samme værdi dukker op mange gange og måske skal ændres i fremtiden, gør det koden svær at vedligeholde.

Sådanne værdier kaldes nogle gange ”magic numbers” (underforstået at det ikke er en god ting at være "magic"). En god praksis er at tildele magic numbers til variable med meningsfulde navne, som dette:

int minutterPerTime = 60;

minutter = timer \* minutterPerTime;

Denne version er lettere at læse, men der er stadig et problem. Variabler kan variere, men antallet af minutter pr. time gør ikke. Når vi først tildeler en værdi til minutterPerTime, bør den aldrig ændre sig. I Java bruger man keywordet final til at sørge for dette.

final int minutterPerTime = 60;

At erklære en variabel final betyder at den ikke kan assignes en ny værdi, når den først er blevet initialiseret. Hvis du prøver, giver det en compile-fejl. Variabler, der er erklæret som final, kaldes konstanter. Konventionen for navngivning af konstanter er at de skrives med store bogstaver og med understregningstegnet (\_) mellem ordene. Sådan her:

final int MINUTTER\_PER\_TIME = 60;

|  |
| --- |
| OPGAVE 4: Lav en konstant i stedet for at have tallet 60 stående i koden, på samme måde som i eksemplet. |