Programmering 1 - Uppgift 2

Peter Holm

Juni 2020

1 Del 1

1.1 Vad är en variabel?

Variabler är behållare där man kan förvara data i program. Dess innehåll(värde) kan förändras med ny data genom programmets gång och för att enkelt se vad en variabel innehåller nämner man dem helst med passande namn.

Anteckning till mig själv;
Man kan inte namnge en variabel som man skulle ge ett namn till en hund -bara för att det är gulligt eller det har bra klang. Till skillnad från hunden och dess namn, som är olika enheter, är en variabel och variabelns namn i princip samma sak. Följaktligen bestäms en variabels godhet eller dålighet till stor del av dess namn. Namnge variabler varsamt.

Exempel på dåliga variabler.

```
x = x - xx;
xxx = sixten + omsattningsskatt( sixten );
x = x + forseningsavgift( x1, x) + xxx;
x = x + ranta( x1, x);
```

Exempel på bra variabler.

```
balance = balance - lastPayment;
monthlyTotal = newPurchases + salesTax( newPurchases );
balance = balance + lateFee( customerID, balance ) + monthlyTotal;
balance = balance + interest( customerID, balance);
```

1.2 Vad menas med att ”deklarera variabler”? Ge några exempel.

Att deklarera variabler till ett program kan tolkas som att man tar fram tomma behållare man tänkt förvara kommande data i vid ett senare skede i programmet. Som i vardagslivet, dukar man fram tallrikar och glas innan maten är färdiglagad.

Exempel;

```
/** I formen 'dataTyp variabelNamn;'
  * datatypen anger vilken sorts data variabeln
  * kan tilldelas.
  * datatyper;
  * int tilldelas heltalsvariabler.
  * double tilldelas flyttal (1.3, 25.5, 0.032 etc)
  * String tilldelas texter*/

// Deklarerade variabler
int age, personalCodeNumber;    // Tilldelas heltal.
double height, width, thickness; // Tilldelas flyttal.
String fullName;                // Tilldelas texter.
```

1.3 Vad måste programsatser avslutas med i Java?

I Java och andra C-baserade språk är det vanligt att satser avslutas med semicolon (**;**). Skulle man glömma dessa får man ett fel vid kompileringen och man kan lätt finna felet och rätta till det därefter. Det finns även verktyg som hjälper en finna dessa fel redan medans man skriver raden.

Exempel på ett kompileringsfel som kan dyka upp när man glömt avsluta en sats med semicolon.

```
Test.java:7: error: ';' expected
        Toolkit.getDefaultToolkit().beep()
                                            ^
1 error
```

1.4 Ge några exempel på tilldelningssatser.

```
double pi = 3.14159;

String text = "35.1";
double radius = Double.parseDouble(text); // 35.1

int a, b = 2, c = 3;
a = b + c; // a = 5
```

1.5 Vad menas med att ”initiera variabler”?

Att initiera en variabel är att tilldela ett värde samtidigt man deklarerar variabeln i programmet. Det är viktigt att tilldela en variabel ett värde innan programmet försöker avläsa vad variabeln innehåller. Kompilatorn upptäcker ifall detta skulle ske och ger ut felmeddelanden som;

```
Info.java:10 error: variable name might not have been initialized
             contactInfo = name + "\n" + adress;
                           ^
1 error
```

Att initiera en variabel ser ut som följande;

```
String text = "Java-programmering";
int heltal = 12;
double flyttal = 2.15;
```

1.6 Vad är en literal? Ge några exempel på literaler.

Numeriska tal i program kallas för literaler.

```
// Exempel
int heltal;
double flyttal;
String text;

heltal = 5;    // Heltalsliteral
flyttal = 5.25; // Flyttalsliteral
text = "6.75";    // OBS! En text, inte en literal!
```

1.7 Hur skriver man kommentarer på en rad? och hur skriver man längre kommentarer dvs. kommentarer som sträcker sig över flera rader?

```
// Denna kommentar tar en rad.

/* Denna kommentar
tar flera
rader
i
programmet*/

/** Denna kan vara till dokumentationskommentarer
  * innan programmet, kunna ge
  * informativa beskrivningar
  * till andra
  * programmerare!
  */
```

1.8 Vad händer vid explicit typomvandling?

När man tilldelar värden från en data-typ till en annan, kan dessa data-typer inte vara kompatibla med varandra. Är typerna däremot kompatibla med varandra utför Java en automatiskt typomvandling. Om inte detta går behöver programmeraren manuellt deklarera explicit typomvandling mellan data-typerna.

Automatisk typomvandling sker när data-typerna;

- Data-typerna är kompatibla.
- När man tilldelar en mindre data-typ till en större.

```
// byte -> short -> int -> long -> float -> double
// Automatisk typomvandling

// Exempel
int i = 10;
long l = i; // Automatisk.

float f = l // automatisk

/** Utmatning:
 * int:  10
 * long: 10
 * float: 10.0
 */
```

Vill man tilldela en data-typ av större sort måste man använda explicit typomvandling.

```
// double -> float -> long -> int -> short -> byte

// Exempel
double d = 50.2;

long l = (long)d; // (datatyp)variabel explicit typomvandling
int i = (int)l;

/** Utmatning:
 * double: 50.2
 * long:  50
 * int:   50
 */
```

1.9 Ge några exempel på grundläggande numeriska operatorer?

Exempel

```
int a = 10, b = 20;
```

Numeriska operatorer i Java		
Operator	**Beskrivning**	**Exempel**
+ (addition)	Adderar värden	a + b = 30
- (subtraktion)	Subtraherar värden	a - b = -10
* (multiplikation)	Multiplicerar värden	a * b = 200
/ (division)	Dividerar värden	b / a = 2
% (modulus)	Delar vänsteroperanden med högeroperanden och returnerar resten.	b % a = 0
++ (ökning)	Ökar värdet på operanden med 1	a++ ger 11
- - (minskning)	Minskar värdet på operanden med 1	a- - ger 9

1.10 Vad menas med prioriteringsordning? Ge några exempel.

Som den ordinära matematikens prioriteringsregler, gäller även vid programmeringens aritmetik. Operatorerna * (multiplikation) och / (division) har högre prioritet än + (addition) och - (subtraktion).

Om ett numeriskt uttryck innehåller fler operatorer utförs den operatorn med högst prioritet först. Har operatorerna samma prioritetgrad utförs uträkningen helt enkelt från vänster till höger.
Man kan skifta ordningen med parenteser precis som i vanlig matematik.
Exempel;

```
int x = 5, y = 3;
x + 2 * y;  // Blir 11 i och med multiplikationens prioritet.

(x + 2) * y; // Blir 21 i och med parentesernas inverkan.

x * 2 * y;  // Blir 30.

x * (2 * y); // Blir 30.
```

Följande har inget med java att göra men finner det intressant hur det kan skilja mellan olika programspråk.
Programmeringsspråk som Lisp-språken använder sig av ”omvänd polsk notation” för att uttrycka prioriteringsreglerna. I ”omvänd polsk notation” skrivs operatorerna före operanderna.
Exempel;

```
;;; Aritmetik i Common-lisp

(let ((x 5) ; Variabel initiering
      (y 3)) ; ********************
      (+ x (* 2 y))   ; 11
      (* (+ x 2) y)   ; 21
      (* x 2 y)       ; 30
      (* (* 2 y) x)) ; 30
```