Lösningsförslag: Instuderingsfrågor, del A

Uppgift 1.

Följande regler gäller för namngivning av identifierare i Java:

1. Ett identifierarnamn består endast av bokstäver, siffror, understrykningstecken('_') och dollartecken ('$').
2. Ett identifierarnamn får vara godtyckligt långt.
3. Ett identifierarnamn får ej börja med en siffra.
4. De reserverade orden i Java är otillåtna identifierarnamn.

Följande identifierare är således ogiltiga:

any.time	innehåller tecket '.'
1stStreet	inleds med en siffra
now&then	innehåller tecket '&'
float	**float** är ett reserverat ord
tooHot?	innehåller tecket '?'
final score	innehåller ett blanktecken

Uppgift 2.

a) 4	b) 13	c) 4
d) 1	e) 2	f) 4
g) 5	h) 2	i) 1

Uppgift 3.

a) Antalet är 17 styck.
b) Antalet är + myNumber + styck.
c) Antalet är 1712 styck.
d) Antalet är 29 styck.

Uppgift 4.

a) 1	b) 3	c) 5.1
d) 28	e) 1.275	f) 13.4
g) 25.6	h) 1.0999999999999996	i) 1.3000000000000007

Uppgift 5.

a) **int**	b) **double**	c) String
d) **boolean**	e) String	f) **char**

Uppgift 6.

Uttrycken x + y * z och x + (y * z) är ekvivalenta, eftersom * har högre prioritet än +.

Uppgift 7.

Följande satser är fel:

int c = 3, d = 4.5;	variabeln d är av typen **int**, men 4.5 är av typen **double**
double x, a;	variabeln a är redan deklarerad som en **int**
char teck = "y";	variabeln teck är av typen **char**, men "y" av typen String

Uppgift 8.

a) alt:= alt + base + col4 + sum;
b) pressure:= (temp + entropy) * spec2;
c) gradient:= gradient - (hgt - slope);
d) eff = eff + full * Math.exp(h3 * Math.log(loss));
e) x = -b + Math.sqrt(b * b - 4 * a * c);

Uppgift 9.

Följande tilldelningar är felaktiga:

c) i + j = k;
Vänsterledet i tilldelningssatsen skall vara den variabel som skall tilldelas uttrycket i högerledet av tilldelningssatsen. Här står uttrycket i vänsterledet och variabeln i högerledet.

h) y = a + b;
Variablerna a och b är av datatypen **boolean** och operatorn + finns ej för denna datatyp.

i) i = x + y;
Variabeln i är av datatypen **int**, medan uttrycket x + y är av datatypen **double**. Således uppstår en typkonflikt, eftersom ett värde av typen **double** inte kan lagras i en variabel av typen **int**.

Uppgift 10.

Implicit typomvandling innebär att en variabel av en "större" typ automatiskt kan tilldelas ett värde av en "mindre" typ, eftersom det "mindre" värdet ryms i den "större" typen. Exempelvis kan en variabel av typen **double** tilldelas ett värde av typen **int** (datatypen **double** inkluderar alla heltal som finns i **int**).

Explicit typomvandling innebär att man med hjälp av en speciell sats kan begära att ett värde av en viss typ görs om till ett värde av en annan typ. Detta är nödvändigt om man vill tilldela ett värde en "större" typ till ett värde av en "mindre" typ. Typomvandlingen innebär i detta fall ofta att det nya värdet endast blir ett närmevärde till det gamla värdet. Exempelvis kan en värde av **double** omvandlas till ett värde av **int** med explicit typomvandling på följande sätt

```
int intTal = (int) 10.1234567;          //värdet blir 10
```

Uppgift 11.

Tilldelningssatsen

```
MAX_SIZE = 50;
```

ger ett kompileringsfel, eftersom MAX_SIZE är en konstant och således inte får tilldelas ett nytt värde.

Uppgift 12.

a) 81.0 b) 4.0 c) 26 d) 13.0

Uppgift 13.

a) 1 / (2 * Math.PI * Math.sqrt(L * C))
b) C * h / Math.pow(Math.pow(h, 2) + Math.pow(d, 2), 3)
c) Math.sin(Math.pow(x, 2) + Math.pow(y, 2))
d) Math.sqrt((a+b)*(a-b)) / (Math.pow(a,2) + 1 / (Math.pow(a, 3) - 2))

Uppgift 14.

För att kunna använda dialogrutan, dvs anropa metoden JOptionPane.showMessageDialog, måste vi ha tillgång till paketet javax.Swing. Därför måste vi först i programmet ge satsen

```
import javax.swing.*;
```

Uppgift 15.

a) index = index + 1;
b) factor = shoeSize + shoulderBreadth;
c) factor = Math.sqrt((**double**) skoAxel / MAX);
d) start = true;
e) flag = size > index;
f) finished = (ch1 == ch2) || (ch1 == 'n') || (ch1 = 'N');

Uppgift 16.

```
year = date / 10000;
month = date % 10000 / 100;
day = date % 100;
```

Uppgift 17.

Följande tilldelningar är felaktiga:

b) i = x / k;

Variabeln i är av datatypen **int**, medan uttrycket x /k är av datatypen **double**. Således uppstår en typkonflikt, eftersom ett värde av typen **double** inte kan lagras i en variabel av typen **int**.

c) i = Math.pow(k , 2);

Variabeln i är av datatypen **int**, medan uttrycket Math.pow(k ,2) är av datatypen **double**. Således uppstår en typkonflikt, eftersom ett värde av typen **double** inte kan lagras i en variabel av typen **int**.

Uppgift 18.

Genom att använda någon av metoderna System.out.printf eller String.format:

```
System.out.printf("%.3f", size);
```

eller

```
System.out.print(String.format("%.3f", size));
```

Uppgift 19.

Genom att använda någon av metoderna System.out.printf eller String.format:

```
System.out.printf("%02d:%02d:%02d", tim, min, sek);
```

eller

```
System.out.println(String.format("%02d:%02d:%02d", tim, min, sek));
```

Uppgift 20.

Man använder sig av ett objekt av klassen Scanner:

```
import java.util.*; //Scanner finns i detta paket och måste
. . .
    String indata = JOptionPane.showInputDialog("Ange önskade tre värden: ");
    Scanner sc = new Scanner(indata);
    int number = sc.nextInt();
    double freq = sc.nextDouble();
    int count = sc.nextInt();
    . . .
```

Uppgift 21.

Programmet blir mycket lättare att läsa och dess logiska uppbyggnad lättare att förstå, vilket t.ex. underlättar när man söker fel i ett program.

Uppgift 22.

Informellt kan man säga att en algoritm är en beskrivning av hur ett problem skall lösas. En algoritm består av en ordnad, ändlig följd av elementära och entydiga instruktioner.

Uppgift 23.

Sekvens, selektion och iteration.

Uppgift 24.

- Definition av problemet (uppgiftsformulering).
- Analys av problemet.
- Framtagning av lösningsskiss.
- Val och representation av algoritmer.
- Kodning.
- Testning och validering (försäkra sig om att programmet löser uppgiften).
- Dokumentation.
- Programunderhåll.

Uppgift 25.

a) **true** b) **false** c) **true** d) **true**

Uppgift 26.

a) **false** b) **false** c) **false** d) **true**

Uppgift 27.

a) false **b) true** **c) true** **d) false**

Uppgift 28.

a) x > y && y > z
b) x < 0 && y < 0
c) !(x < 0 && y < 0)
d) x == y && x != z

Uppgift 29.

c

Uppgift 30.

```
!(z == x) || (x > 2) && (y > 3)
= !false || true && false
= true || true && false
=  true || false
 =  true
```

Uppgift 31.

```
delbar = tal % 7 == 0;
```

Uppgift 32.

a) x > 3
b) y >= 2 && y <= 5
c) r < 0 && z >=0
d) (a < 0 && b < 0) || (a>= 0 && B >= 0)
e) p == q && q != r

Uppgift 33.

```
number >= 80 && number <=90
```

Uppgift 34.

Problemet med reella tal är att dessa inte kan lagras exakt och att det beräkningar uppstår trunkeringsfel och kancellationsfel. Därför kan man inte jämföra på exakthet när det gäller reella tal utan man får istället jämföra på tillräcklig noggrannhet. Vad tillräcklig noggrannhet är i de aktuella frågeställningarna kan väl diskuteras, men en avvikelse mellan vikterna på några tiondels procent kan väl vara acceptabelt.

a) Math.abs(weight1 - weight2) < 0.1,

b) Math.abs(weight1 -weight2) < 0.00001

c) Math.abs(weight1 -weight2) < 50

Uppgift 35.

a) value >= 1.0

b) value < 0.0 || value >= 1.0

c) smallNumber != 0 && bigNumber != 10000

d) value > 1.0

Uppgift 36.

Tabellen nedan visar att de tre uttrycken är ekvivalenta

x < 10	(x < 10) == **true**	x >= 10	(x >= 10) == **false**
true	**true**	**false**	**true**
false	**false**	**true**	**false**

Det enklaste av dessa villkor att förstå torde vara x < 10

Uppgift 37.

a) Always b) OK

Uppgift 38.

Utskriften blir:

```
z = 10.0
```

Orsaken till detta är att **else**-satsen inte hör till den yttre **if**-satsen, som indenteringen av programsatserna kan antyda, utan till den inre **if**-satsen. Rätt indenterat ser programsekvensen ut enligt följande:

```
int x = -1, y = 4, z = 10;
if (x > 0)
    if (y > 0)
        z = Math.sqrt(x) + Math.sqrt(y);
    else
        z = 0;
System.out.println("z = " + z);
```

Uppgift 39.

Orsaken är att det står ett semikolon efter villkoret i **if**-satsen!! Således får vi en **if**-sats med en tom sats och utan **else**-del. Alltså hör inte **else** till någon **if**-sats! Om vi indenterar programmet som det logiskt tolkas syns detta tydligare:

```
if (x == 0)
    ;
x = 100;
else
    x = x + 50;
```

Uppgift 40.

Tabellen nedan visar att b och !test har samma värde

test	!test	b
true	**false**	**false**
false	**true**	**true**

Uppgift 41.

a) 12.5 b) 30.0

Uppgift 42.

När man har en **if**-sats inne i en annan **if**-sats säger man att man har nästlade **if**-satser.

Uppgift 43.

a)
```
if (a > c && b > c)
    x = y;
else
    x = z;
```
b)
```
if (a == b || a > c)
    x = y;
else
    x = z;
```

Uppgift 44.

I programsekvens a) har vi en **if**-sats med ett tvåvägsalternativ, medan vi i programsekvens b) har två **if**-satser som båda har envägsvalsalternativ. Så i programsekvens a) kan högst en av tilldelningssatserna genomlöpas, medan i programsekvens b) kan båda tilldelningssatserna genomlöpas.

Om x har värdet -1 innan respektive programsekvens kommer x att ha värdet -1 efter att sekvens a) genomlöpts och värdet -1 efter att sekvens b) genomlöpts.

Om x har värdet 1 innan respektive programsekvens kommer x att ha värdet 2 efter att sekvens a) genomlöpts och värdet 4 efter att sekvens b) genomlöpts.

Uppgift 45.

```
finished = answer =='Q';
```

Uppgift 46.

Variabeln y deklareras i programblocket som hör till **if**-satsen och är därför en lokal variabel i detta programblock, medan satsen System.out.println(y) ligger utanför detta block. Variabeln y är således okänd i denna sats.

Uppgift 47.

```
if (x < 0 && y < 3 && z = 6)
    a = x + y + z;
```

Uppgift 48.

"Algoritm" b) är felaktig. Vi betalar med check även om vi redan har betalt med kontanter.

Uppgift 49.

Följande uttryck

```
(finished && !dead) || (dead && !finished)
```

och

```
finished != dead
```

är logiskt ekvivalenta.

Uppgift 50.

Utskriften blir:

```
33
21 + 12
The answer is: 2112
The answer is: 33
```

Uppgift 51.

```
public class Address {
    public static void main(String[] args) {
        System.out.println(Kalle Anka);
        System.out.println("Vassgatan 32");
        System.out.println("123 45 Ankeborg ");
}//Address
```

Uppgift 52.

```
public class CelciusToFahrenheit {
    public static void main (String[] arg) {
        String indata = JOptionPane.showInputDialog("Ange temperaturen i Celcius: ");
        double celcius = Double.parseDouble(indata);
        double fahrenheit = 9.0 / 5.0 *celsius + 32.0;
        JOptionPane.showMessageDialog(null, "Temperaturen i Fahrenheit är " + fahrenheit);
    } //main
}//CelciusToFahrenheit
```

Uppgift 53.

```
import javax.swing.*;
public class Gross {
    public static void main(String[] arg) {
        String indata = JOptionPane.showInputDialog("Ange antalet: ");
        int antal = Integer.parseInt(indata);
        int gross = antal / 144;;
        int dussin = antal %144 / 12;
        int rest = antal % 12;
        JOptionPane.showMessageDialog(null, antal + " är " + gross + " gross, " + dussin
                                                + " dussin och " + rest + " stycken.");
    } // main
}//Gross
```