booleans

values	true or false
literals	true false
operations	and or n
operators	&& ||

if

boolean expression

```
if ( x > y )
{
   int t = x;
   x = y;
   y = t;
}
```

sequence of statements

op	meaning	true	false
==	equal	2 == 2	2 == 3
!=	not equal	3 != 2	2 != 2
<	less than	2 < 13	2 < 2
<=	less than or equal	2 <= 2	3 <= 2
>	greater than	13 > 2	2 > 13
>=	greater than or equal	3 >= 2	2 >= 3

parse

`int Integer.parseInt(String s)`	convert s to an int value
`double Double.parseDouble(String s)`	convert s to a double value
`long Long.parseLong(String s)`	convert s to a long value

if og if-else

absolute value	`if (x < 0) x = -x;`
put the smaller value in x and the larger value in y	`if (x > y) { int t = x; x = y; y = t; }`
maximum of x and y	`if (x > y) max = x; else max = y;`
error check for division operation	`if (den == 0) System.out.println("Division by zero"); else System.out.println("Quotient = " + num/den);`
error check for quadratic formula	`double discriminant = b*b - 4.0*c; if (discriminant < 0.0) { System.out.println("No real roots"); } else { System.out.println((-b + Math.sqrt(discriminant))/2.0); System.out.println((-b - Math.sqrt(discriminant))/2.0); }`

non-negative discriminant?	`(b*b - 4.0*a*c) >= 0.0`
beginning of a century?	`(year % 100) == 0`
legal month?	`(month >= 1) && (month <= 12)`

loops

compute the largest power of 2 less than or equal to n	`int power = 1; while (power <= n/2) power = 2*power; System.out.println(power);`
compute a finite sum $(1 + 2 + \ldots + n)$	`int sum = 0; for (int i = 1; i <= n; i++) sum += i; System.out.println(sum);`
compute a finite product $(n! = 1 \times 2 \times \ldots \times n)$	`int product = 1; for (int i = 1; i <= n; i++) product *= i; System.out.println(product);`
print a table of function values	`for (int i = 0; i <= n; i++) System.out.println(i + " " + 2*Math.PI*i/n);`
compute the ruler function (see PROGRAM 1.2.1)	`String ruler = "1"; for (int i = 2; i <= n; i++) ruler = ruler + " " + i + " " + ruler; System.out.println(ruler);`

while loop

initialization is a separate statement

loop-continuation condition

```
int power = 1;
while ( power <= n/2 )
{
   power = 2*power;
}
```

braces are optional when body is a single statement

body

for loop

initialize another variable in a separate statement

declare and initialize a loop control variable

loop-continuation condition

increment

```
int power = 1;
for (int i = 0; i <= n; i++)
{
   System.out.println(i + " " + power);
   power = 2*power;
}
```

body

funksjon

signature | return type | method name | argument type | argument variable

```
public static double harmonic ( int n )
{
   double sum = 0.0;
   for (int i = 1; i <= n; i++);
      sum += 1.0/i;
   return sum;
}
```

local variable

method body

return statement

class

```
public class Charge
{
   private final double rx, ry;
   private final double q;

   public Charge(double x0, double y0, double q0)
   {  rx = x0; ry = y0; q = q0;  }

   public double potentialAt(double x, double y)
   {
      double k = 8.99e09;
      double dx = x - rx;
      double dy = y - ry;
      return k * q / Math.sqrt(dx*dx + dy*dy);
   }

   public String toString()
   {  return q +" at " + "("+ rx + ", " + ry +")";  }

   public static void main(String[] args)
   {
      double x = Double.parseDouble(args[0]);
      double y = Double.parseDouble(args[1]);
      Charge c1 = new Charge(0.51, 0.63, 21.3);
      Charge c2 = new Charge(0.13, 0.94, 81.9);
      double v1 = c1.potentialAt(x, y);
      double v2 = c2.potentialAt(x, y);
      StdOut.printf("%.2e\n", (v1 + v2));
   }
}
```

class name

instance variables

constructor

instance variable names

instance methods

test client

create and initialize object

invoke constructor

object name

invoke method

funksjoner - eksempler

absolute value of an int value	`public static int abs(int x) { if (x < 0) return -x; else return x; }`
absolute value of a double value	`public static double abs(double x) { if (x < 0.0) return -x; else return x; }`
primality test	`public static boolean isPrime(int n) { if (n < 2) return false; for (int i = 2; i <= n/i; i++) if (n % i == 0) return false; return true; }`
hypotenuse of a right triangle	`public static double hypotenuse(double a, double b) {  return Math.sqrt(a*a + b*b);  }`
harmonic number	`public static double harmonic(int n) { double sum = 0.0; for (int i = 1; i <= n; i++) sum += 1.0 / i; return sum; }`
uniform random integer in [0, n)	`public static int uniform(int n) {  return (int) (Math.random() * n);  }`
draw a triangle	`public static void drawTriangle(double x0, double y0, double x1, double y1, double x2, double y2 ) { StdDraw.line(x0, y0, x1, y1); StdDraw.line(x1, y1, x2, y2); StdDraw.line(x2, y2, x0, y0); }`