F2 – Datatyper och variabler

ID1004 Objektorienterad programmering

Fredrik Kilander fki@kth.se

Datatyper

- Java är ett starkt typat språk
- Varje slags data har en *datatyp*

Datatyp	Javasyntax	Exempel
Teckensträng	String	”Alan tar en kaka”
32-bitars heltal	int	0, -4999, 1000000
64-bitars flyttal	double	3.141592d
Enskilt tecken	char	’H’, ’s’, ’\n’
Sanningsvärde	boolean	true, false
Objekt	*<klassnamn>*	...

Variabler

- Variabler är namngivna databehållare
- Varje variabel deklareras med en datatyp
- En variabel kan bara ta värden ur sin datatyp

```
int count = 0;
String name = ”Alan”;
double x;
double y = 4.1
double z = 0;

x = y + z;
```

Javakompilatorn ser obönhörligt till att datatypreglerna efterlevs.

Konstanter

- Konstanter är variabler som inte ändras
- De får sitt värde vid initialisering
- Konstanter ger mer läsbar kod
- De deklareras med reserverat ord **`final`**
- Namnges med stora bokstäver och underscore

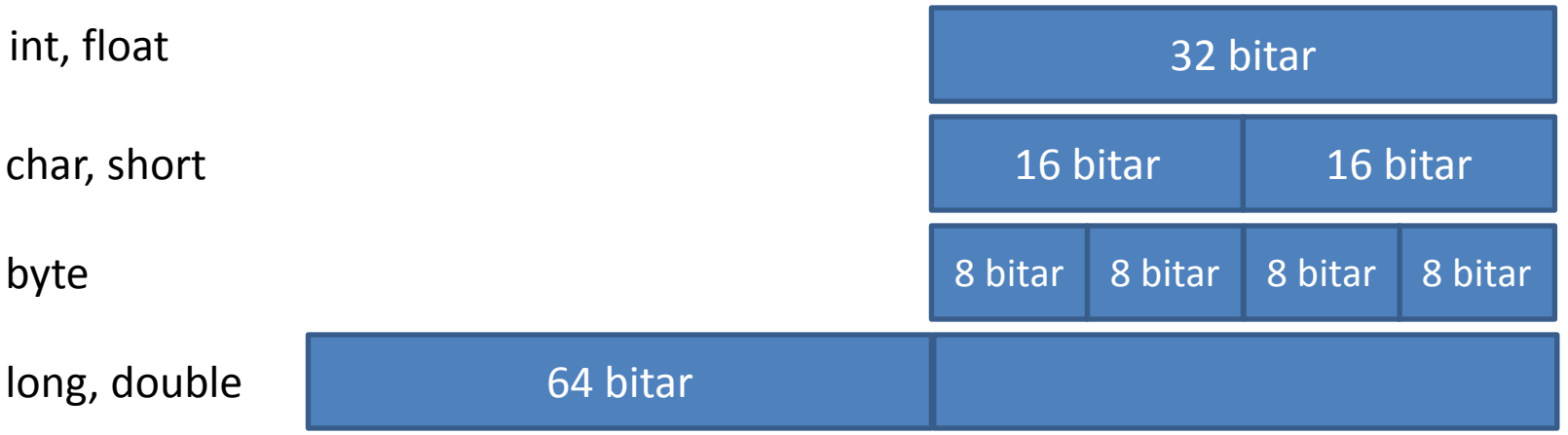
```
final double PI = 3.141592;
final int SECONDS_IN_HOUR = 60 * 60;

double area = radius * radius * PI;
int hours = seconds / SECONDS_IN_HOUR;
```

Kompilatorn kontrollerar att konstanter inte tilldelas nya värden.

Primitiva datatyper

- Primitiva datatyper innehåller tal
- Datatypen bestämmer hur mycket som får plats och hur det tolkas vid läsning
- Vid tilldelning av variabeln ändras innehållet men tolkningen är densamma



Primitiva datatyper

- `int antalTrappor = 5;`
- `short nextSample = -32720;`
- `char delimiter = 'X';`
- `long ms = 2847284978;`
- `double z = 0.0023;`

Referensdatatyper

- En referensdatatyp avser alltid en viss klass
- En referensvariabel pekar på ett objekt (en instans av klassen)
- Om en referensvariabel inte pekar på ett objekt så är värdet **`null`**

```
String s = null;
Lincoln abe = new Lincoln();
```

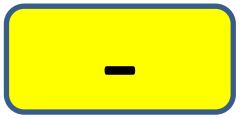
Klasser och deras objekt utgör den stora mängden av datatyper.

Referensdatatyper

```
{

  String s = null;
  …
}
```

s refererar inte till någon instans
Värdet är **`null`**

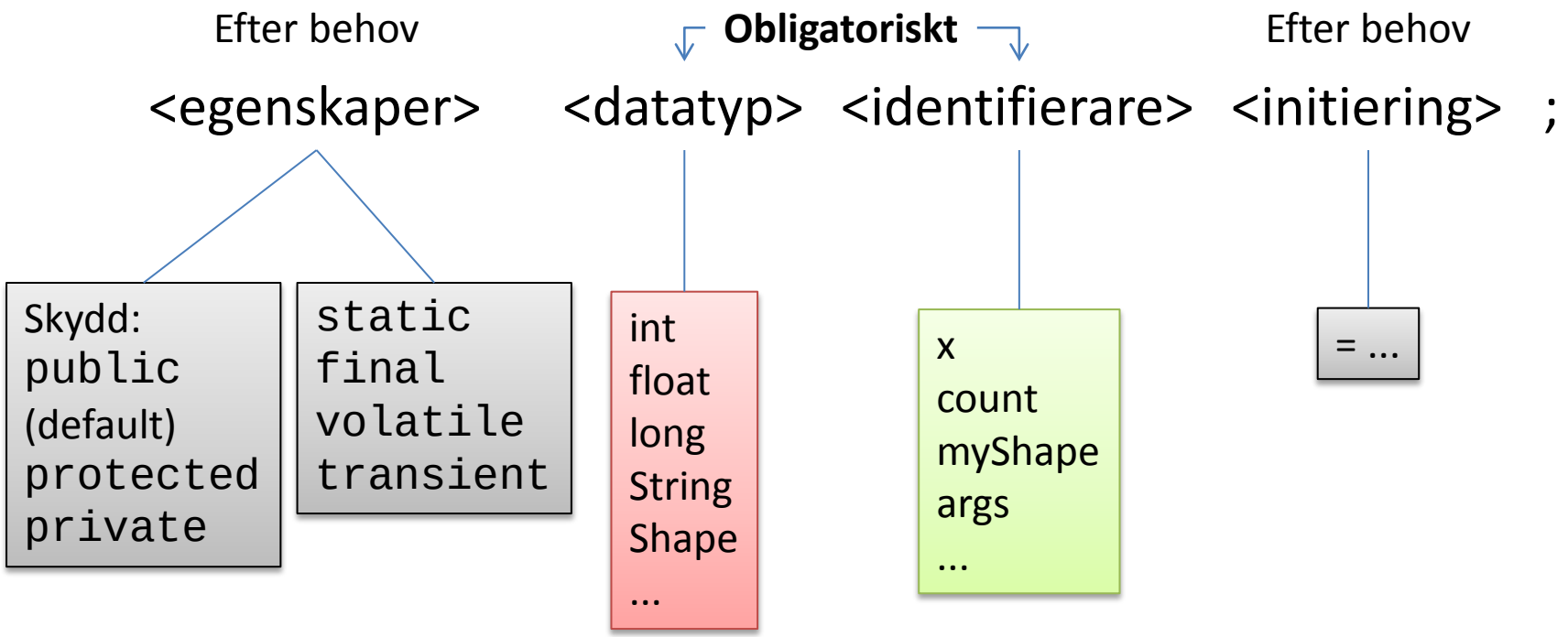
String s 

Referensdatatyper

```
{
  String s = null;
  …
  s = ”Fredrik Kilander”;
  …
}
```

s refererar till en instans av String
Värdet är skilt från **`null`**

Att deklarera variabler



Aritmetiska operatorer

- **+** addition (även strängsammansättning)
- **-** subtraktion
- ***** multiplikation
- **/** divison (olika för heltal och flyttal)
- **%** modulo (rest vid heltalsdivision)

Operatorers precedensordning

- Multiplikation och division utförs först
- Därefter addition och subtraktion
- Från vänster till höger vid samma precedens.

```
a + b + c / e * f – 1;
a + b + ((c / e) * f) - 1
```

- Använd parenteser. Även om du har järnkoll på precedensordningen så kanske inte nästa person som ska arbeta med koden har det.

Operatorers precedensordning

- `n = 4 + 3 * 2; // n == 10`
- `n = 4 + (3 * 2); // n == 10`
- Parenteser styr uträkningen
- `n = (4 + 3) * 2;  // n == 14`

Operatorers precedensordning

- `n = 10 / 2 * 5; // n == ?`

Operatorers precedensordning

- `n = 10 / 2 * 5; // n == 25`

- Från vänster till höger vid samma precedens.

Operatorers precedensordning

- `n = 10 / 2 * 5; // n == 25`
- Från vänster till höger vid samma precedens.
- `n = 10 / (2 * 5); // n == 1`

Tilldelningsoperatorer

- `=` vanlig tilldelning

- `+=` addera högerledet och tilldela
- `-=` subtrahera högerledet och tilldela
- `*=` multiplicera med högerledet och tilldela
- `/=` dividera med högerledet och tilldela
- `%=` tag modulo högerledet och tilldela

Tilldelningsoperatorer

- **=** vanlig tilldelning
- `k = k + 1;`
- Denna form kan alltid användas

Tilldelningsoperatorer

- **+=** addera högerledet och tilldela

- `k += 1; // Samma som k = k + 1`

Tilldelningsoperatorer

- **`-=`** subtrahera högerledet och tilldela

- `k -= 3; // Samma som k = k - 3`

Tilldelningsoperatorer

- **`*=`** multiplicera med högerledet och tilldela

- `k *= 3; // Samma som k = k * 3`

Tilldelningsoperatorer

- **`/=`** dividera med högerledet och tilldela

- `k /= 2; // Samma som k = k / 2`

Tilldelningsoperatorer

- **%=** tag modulo högerledet och tilldela

- `k %= 5; // Samma som k = k % 5;`

Boolska operatorer

- Boolska operatorer ger **`true`** eller **`false`**
- < mindre än
- > större än
- <= mindre eller lika med
- >= större eller lika med
- == lika med (även identitet för ref.variabler)
- != skilt från

Boolska operatorer

- Boolska operatorer ger **`true`** eller **`false`**
- `5 < 7 // true`
- `10 <= 9 // false`
- `0 != -1 // true`
- `5 == 7  // false`

Logiska operatorer

- Logiska operatorer (tar **`true`** eller **`false`** som operander)
- && och
- || eller
- ! negation

Operatorerna && och || är kortslutna, dvs om vänsterledet är tillräckligt så beräknas inte högerledet.

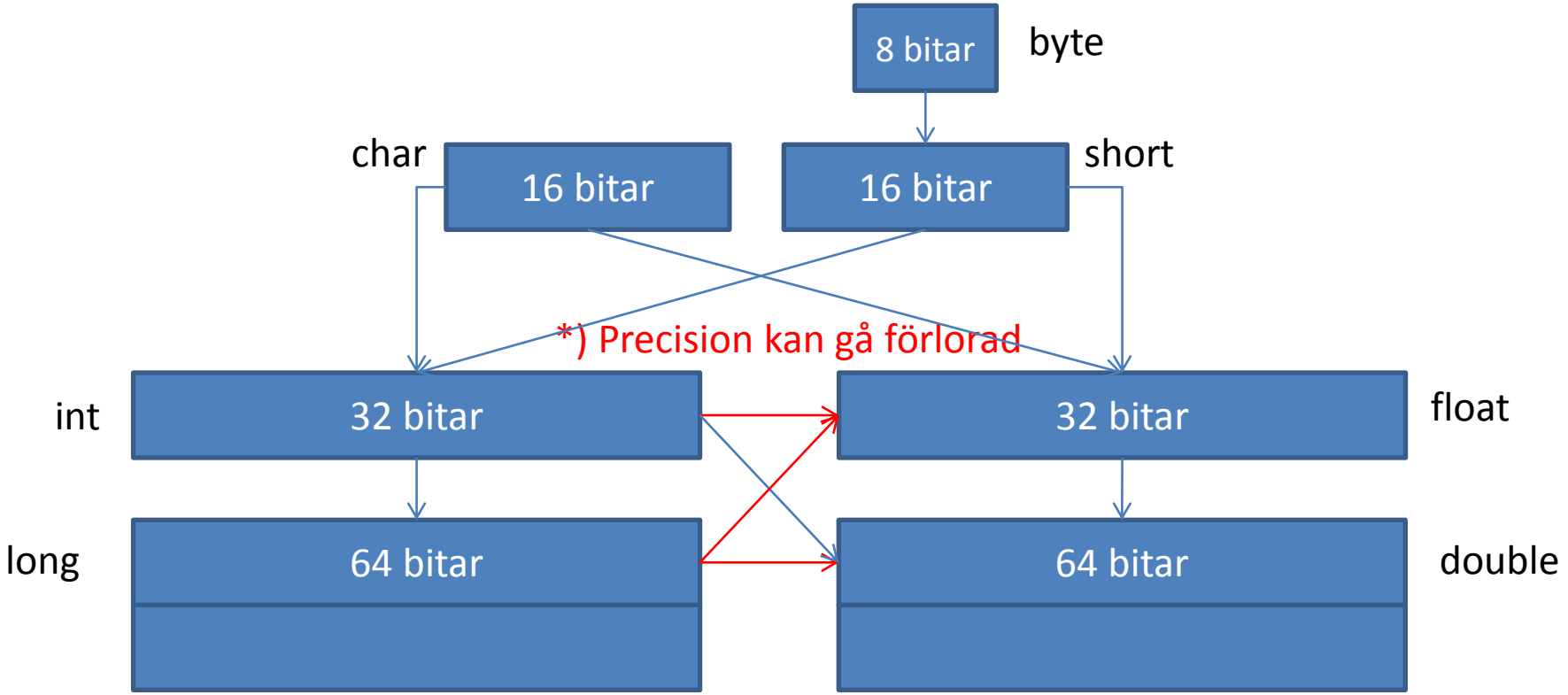
Logiska operatorer

```
int i;
…
if ((0 < i) && (i < upperLimit)) {
  // i är större än noll OCH
  // mindre än upperLimit
}

int k;
…
if ((k < 20) || (60 < k)) {
  // k är mindre än 20 ELLER
  // större än 60
}
```

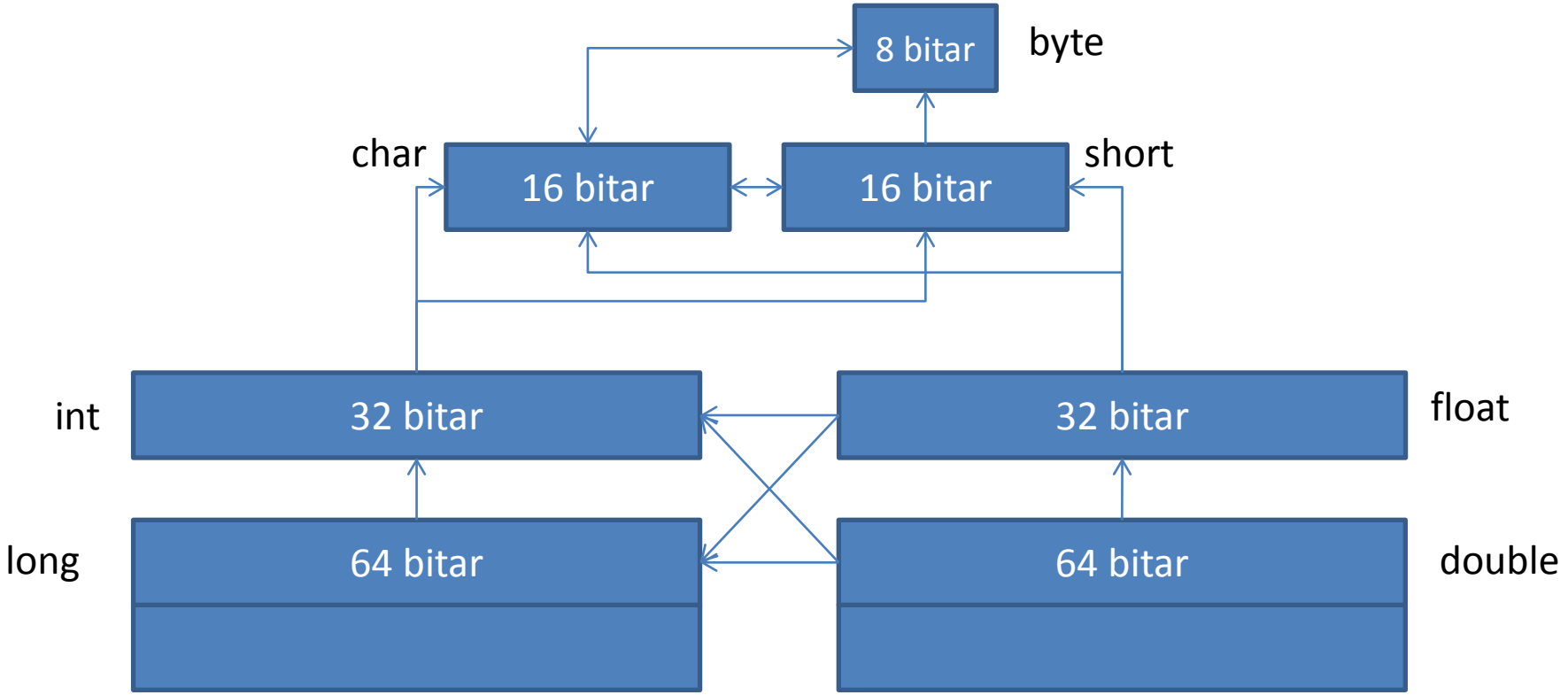
Datakonvertering

- En mindre primitiv datatyp kan utan förlust* konverteras till en större primitiv datatyp



Datakonvertering

- Avsmalnande konvertering riskerar att tappa data. Programmeraren måste vara säker på att det kommer att fungera.



Inläsning av textdata – klass java.util.Scanner

- Konstruktorer
  - **`Scanner (InputStream source)`**
  - **`Scanner (File source)`**
  - **`Scanner (String source)`**
- Nästa token som en teckensträng:
  - **`String next()`**
- All kvarvarande data på aktuell rad:
  - **`String nextLine()`**

Inläsning av textdata – klass java.util.Scanner

- Klass **Scanner** ingår i standardbiblioteket
- Scanner passar bra för strukturerad data, t ex en komma- eller semikolonseparerad lista där man vet vad som kommer

Echo.java

jo

F2 - INGA FLER BILDER

Echo.java

```
//********************************************************************
//  Echo.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the nextLine method of the Scanner class
//  to read a string from the user.
//********************************************************************

import java.util.Scanner;

public class Echo
{
   //-----------------------------------------------------------------
   //  Reads a character string from the user and prints it.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      String message;
      Scanner scan = new Scanner (System.in);

      System.out.println ("Enter a line of text:");

      message = scan.nextLine();

      System.out.println ("You entered: \"" + message + "\"");
   }
}
```

Echo.java

```
//********************************************************************
//  Echo.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the nextLine method of the Scanner class
//  to read a string from the user.
//********************************************************************

import java.util.Scanner;

public class Echo
{
   //-----------------------------------------------------------------
   //  Reads a character string from the user and prints it.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      String message;
      Scanner scan = new Scanner (System.in);

      System.out.println ("Enter a line of text:");

      message = scan.nextLine();

      System.out.println ("You entered: \"" + message + "\"");
   }
}
```

Echo.java

```
//********************************************************************
//  Echo.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the nextLine method of the Scanner class
//  to read a string from the user.
//********************************************************************

import java.util.Scanner;

public class Echo
{
   //-----------------------------------------------------------------
   //  Reads a character string from the user and prints it.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      String message;
      Scanner scan = new Scanner (System.in);

      System.out.println ("Enter a line of text:");

      message = scan.nextLine();

      System.out.println ("You entered: \"" + message + "\"");
   }
}
```

Echo.java

```
//********************************************************************
//  Echo.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the nextLine method of the Scanner class
//  to read a string from the user.
//********************************************************************

import java.util.Scanner;

public class Echo
{
   //-----------------------------------------------------------------
   //  Reads a character string from the user and prints it.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      String message;
      Scanner scan = new Scanner (System.in);

      System.out.println ("Enter a line of text:");

      message = scan.nextLine();

      System.out.println ("You entered: \"" + message + "\"");
   }
}
```

GasMileage.java

```
import java.util.Scanner;

public class GasMileage {
   //-----------------------------------------------------------------
   //  Calculates fuel efficiency based on values entered by the user.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int miles;
      double gallons, mpg;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Enter the number of miles: ");
      miles = scan.nextInt();

      System.out.print ("Enter the gallons of fuel used: ");
      gallons = scan.nextDouble();

      mpg = miles / gallons;

      System.out.println ("Miles Per Gallon: " + mpg);
   }
}
```

Operatorer och uttryck

- Inkrement- och dekrementoperatorerna ++, --
- **`++i`** öka värdet med ett innan läsning
- **`i++`** läs värdet öka därefter värdet med ett
- **`--i`** minska värdet med ett innan läsning
- **`i--`** läs värdet minska därefter värdet med ett

```
if ((0 <= i) && (i < upperLimit)) {
  a = num[i++];
}
```

Dessa operatorer behövs inte så ofta, men är praktiska ibland.

Inläsning av textdata – klass java.util.Scanner

- Nästa token som typad data:
  - **`boolean nextBoolean()`**
  - **`byte nextByte()`**
  - **`double nextDouble()`**
  - **`float nextFloat()`**
  - **`int nextInt()`**
  - **`long nextLong()`**
  - **`short nextShort()`**
- Kastar InputMismatchException

Inläsning av textdata – klass java.util.Scanner

- Finns ett token till?
  - **`boolean hasNext()`**
- Sätter mönster mellan tokens:
  - **`Scanner useDelimiter(String pattern)`**
  - **`Scanner useDelimiter(Pattern pattern)`**
- Returnerar aktuellt mönster:
  - **`Pattern delimiter()`**
- Hitta nästa match om det går:
  - **`String findInLine(String pattern)`**
  - **`String findInLine(Pattern pattern)`**