Typer

Primitiva typer

prim. type	motsv. klass	ex. literaler	default
`boolean`	`Boolean`	`true`, `false`	`false`
`char`	`Character`	`'A'`, `'3'`, `'\n'`	`'\u0000'`
`int`	`Integer`	37, -3, 12345	0
`double`	`Double`	3.1416, 1E-10	0.0

Referensyper

- Array/fält:
  Exempel: `int[], Ball[], double[][]`
  Skapa objekt: `int[] a = new int[10];`
  Initiering: `double[][] data = {{1,3},{4,8}};`
  Indexering: `a[i], 0 <= i < a.length`.

- Klasser:
  Skapas med konstruerare:
  `Ball b = new Ball(10,20,Color.RED);`
  `LifeModel model = new LifeModel(50,50);`

- Interfaces/gränssnitt:
  Deklarerar metoder med resultattyp, namn, parametrar, undantag. Objekt kan inte skapas, men klasser kan implementera ett interface.

Uppräkningstyper

```
enum E {VAL1, VAL2, ...}
```

Variabler

Variabler måste deklareras: `int x; double[] ys;`

Initiering
Instansvariabler and array-element initieras till default-värdet för primitiva typer och `null` för referenstyper. Lokala variabler måste initieras explicit.

Uttryck

Uttryck byggs av variabler, literaler, operatorer och metodanrop.

Binära operatorer i precedensordning

operator	argtyp	restyp
`*`, `/`, `%`	number	number
`+`, `-`	number*	number*
`<`, `<=`, `>`, `>=`	number	boolean
`==`, `!=`	any	`boolean`
`&&`	`boolean`	`boolean`
`||`	`boolean`	`boolean`
=, +=, -=	var, t/number	t/number

*) + kan också ha `String` som argtyp och restyp.
number betyder numerisk primitiv typ.

Andra operatorer

operator	argtyp	restyp
expr`++`, *expr*`--`	number	number
`++`*expr*, `--`*expr*, `-`*expr*	number	number
`!`*expr*	boolean	boolean
expr`?`*expr*`:`*expr*	boolean, *t*, *t*	*t*

Typomvandling
Omvandling av ett värde av typen `int` eller `double` till `double` eller `String` sker implicit vid behov i uttryck. I omvänd riktning krävs explicit typomvandling, t. ex. `(int)`. Typomvandling från en klass till en superklass sker automatiskt. I omvänd riktning krävs explicit typomvandling. Opertorn *instanceof* kan användas för att avgöra om omvandling är möjlig.
Omvandling mellan primitiva typer och dess motsvarande klass sker automatiskt i båda riktningarna.

Generics

```
class C<T1, T2, ...> {...}
interface I<T1, T2, ...> {...}
<T1, T2, ...> rettyp m(argtyp1 argnamn1, ...)
```

Modifierare och andra nyckelord

```
abstract, static, final, private, protected,
public
extends, implements, super, this, throw,
throws, void
```

Satser

expr`;`	*typ var*`;`
	typ var `=` *init*`;`
`break;`	`continue;`
`return;`	`return` *expr*`;`

`if (`*test*`) {`
statements
`}`
	`if (`*test*`) {`
	statements
	`} else {`
	statements
	`}`

`while (`*test*`) {`
statements
`}`
	`do {`
	statements
	`} while (`*test*`);`

`for (`*init*`;` *test*`;` *upd*`) {`
statements
`}`
	`for (`*type var* `:` *expr*`) {`
	statements
	`}`

`switch (`*expr*`) {`
`case` *lit1*`:` *stmt*
...
`case` *litN*`:` *stmt*
`}`
	`try {`
	statements
	`} catch (`*exc-type var*`) {`
	statements
	`}`

lambda-uttryck: *params* `->` *funktionskropp*	

Klasser

- Funktionsbibliotek (Ex: `Math`, `Arrays`). Innehåller bara statiska funktioner/subrutiner.

- Mallar från vilka objekt skapas (de flesta klasser). Innehåller oftast inte statiska metoder. I stället instansvariabler, konstruerare, metoder.

- Huvudklassen i en applikation. Innehåller `public static void main(String[] args)`

Funktionsbibliotek

Alla rutiner i detta avsnitt är `static` även om det inte är utskrivet.

java.lang.Math
Bland andra: `abs`, `max`, `min`, `sin`, `cos`, `exp`, `log`, `pow`, `sqrt`, `round`, `floor`, `random`.

Funktionsbibliotek, forts.

java.util.Arrays

```
int binarySearch(X [] a, X key)
X [] copyOf(X [] orig, int len)
boolean equals(X [] a1, X [] a2)
void fill(X [] a, X val)
void sort(X [] a)
String toString(X [] a)
```

java.lang.Character

```
int digit(char ch, int radix)
char forDigit(int dig, int radix)
boolean isDigit(char ch)
boolean isLetter(char c)
char toLowerCase(char ch)
char toUpperCase(char ch)
```

java.lang.Double

```
double parseDouble(String str)
```

java.lang.Integer

```
int parseInt(String str)
```

java.lang.System

```
InputStream in
PrintStream out
PrintStream err
void exit(int status)
long currentTimeMillis()
```

Object och String

java.lang.Object

```
boolean equals(Object obj)
String toString()
```

java.lang.String

```
implements Comparable
char charAt(int index)
byte[] getBytes()
int indexOf(String str)
int length()
String[] split(String regex)
boolean startsWith(String prefix)
String subString(int beginIndex,
                 int endIndex)
String toLowerCase()
String toUpperCase()
```

In- och utmatning

java.util.Scanner

```
Scanner(File source)
  throws FileNotFoundException
Scanner(InputStream source)
boolean hasNextX()
X nextX()
boolean hasNext()
String next()
String nextLine()
boolean hasNextLine()
Stream<String> tokens()
```

java.io.BufferedReader

```
BufferedReader(Reader in)
  throws FileNotFoundException
String readLine()
int read()
Stream<String> lines()
void close()
```

Exempel: initiering av BufferedReader:

```
BufferedReader in=new BufferedReader(
        new FileReader("foo.in"));
```

java.io.PrintStream

```
PrintStream(File file)
  throws FileNotFoundException
void print(X x)
void println(X x)
void print(String str)
void println(String str)
```

Collections

interface java.util.Iterator<T>

```
boolean hasNext()
T next()
void remove()   optional method
```

interface java.lang.Iterable<T>

```
Iterator<T> iterator()
```

interface java.lang.Comparable<T>

```
int compareTo(T o)
```

interface java.util.Collection<T>
extends Iterable<T>

```
boolean add(T e)
int size()
default Stream<E> stream()
```

interface java.util.List<T>
extends Collection

```
void add(int index, T element)
T get(int index)
T set(int index, T element)
T remove(int index)
```

class java.util.ArrayList<T>
implements List<T>

```
ArrayList<T>()
```

class java.util.LinkedList<T>
implements List<T>

```
LinkedList<T>()
```

interface Set<E>

```
boolean contains(Object o)
boolean remove(Object o)
```

class HashSet<E>
implements Set<E>

```
HashSet<E>()
```

interface Map<K, V>

```
V get(K key)
V put(K key, V value)
V remove(K key)
Set<K> keySet()
```

class HashMap<K,V>
implements Map<K,V>

```
HashMap<K,V>()
```

Stream

interface java.util.Stream<T>

```
Stream<T> filter
        (Predicate<? super T> p)
void forEach (Consumer<? super T> a)
boolean allMatch
        (Predicate<? super T> p)
boolean anyMatch
        (Predicate<? super T> p)
```

Exempel: En `Stream<String> s` kan konverteras till en lista med följande kod:

```
List<String> asList =
         s.collect(Collectors.toList());
```

---

java.util.function

Interfacen nedan är s.k. *funktions-interface.*

interface Predicate<T>
Representerar ett predikat (funktion som returnerar en boolean) med ett argument.

```
boolean test(T t)
default Predicate<T> and
   (Predicate<? super T> otherP)
default Predicate<T> or
   (Predicate<? super T> otherP)
```

interface Consumer<T>
Representerar en operation som tar ett argument och inte returnerar något resultat.

```
void accept(T t)
```

interface Function<T,R>
Representerar en funktion som tar ett argument och producerar ett resultat av typ R.

```
R apply (T t)
```

AWT/Swing (förenklat)

class java.awt.Component

```
void addMouseListener(MouseListener l)
void repaint()
void setBackground(Color c)
void setPreferredSize(Dimension d)
void setVisible(boolean b)
```

class java.awt.Container
extends Component

```
Component add(Component comp)
void setLayout(LayoutManager mgr)
```

javax.swing.JFrame extends Container

```
JFrame()
JFrame(String title)
Container getContentPane()
void pack()
```

javax.swing.JPanel
extends Container

```
JPanel()
void paintComponent(Graphics g)
```

javax.swing.JButton
extends Container

```
JButton(String text)
void addActionListener(
   ActionListener l)
```

interface java.awt.event.MouseListener

```
void MouseClicked(MouseEvent e)
```

interface java.awt.event.ActionListener

```
void actionPerformed(ActionEvent e)
```

java.awt.event.MouseEvent

```
int getX()
int getY()
```

java.awt.BorderLayout
implements LayoutManager

```
static String CENTER, WEST, SOUTH, ...
```

java.awt.Graphics

```
void setColor(Color c)
void drawLine(int x1,int y1,
              int x2,int y2)
void drawOval(int x,int y,int w,int h)
void drawRect(int x,int y,int w,int h)
void fillOval(int x,int y,int w,int h)
void fillRect(int x,int y,int w,int h)
void drawString(String str,int x,int y)
```