

Random()	skapar "slumpmässig" slumpalgsgenerator
Random(long seed);	- med bestämt slumpfelsfrö
int nextInt(int n);	heltal i intervallet [0, n)
double nextDouble();	double-tal i intervallet [0.0, 1.0)
Scanner(File f);	läser från filen f, ofta System.in
Scanner(String s);	läser från strängen s
String next();	läser nästa sträng fram till whitespace
boolean hasNext();	ger true om det finns mer att läsa
int nextInt();	nästa heltal; också nextDouble(), ...
boolean hasNextInt();	också hasNextDouble(), ...
String nextLine();	läser resten av raden

File, import java.io.File/FileNotFoundException/PrintWriter

Läsa från fil	Skapa en Scanner med new Scanner(new File(filename)), Ger FileNotFoundException om filen inte finns. Sedan läser man "som vanligt" från scanneren (nextInt och liknande).
Skriva till fil	Skapa en PrintWriter med new PrintWriter(new File(filename)), Ger FileNotFoundException om filen inte kan skapas. Sedan skriver man "som vanligt" på PrintWriter-objektet (println och liknande).
Fånga undantag	Så här gör man för att fånga FileNotFoundException: <pre>Scanner scan = null; try { scan = new Scanner(new File("data.txt")); } catch (FileNotFoundException e) { ... ta hand om felet }</pre>

Specialtecken

\n	ny rad, radframmatningsstecken
\t	ny kolumn, tabulatorstecken (eng. tab)
\\	bakåtsnedstreck: \ (eng. backslash)
\"	citationstecken: "
\"	apostrof: '

Nedan 50 ord kan ej användas som identifierare i Java. Orden **goto** och **const** är reserverade men används ej.

```
abstract assert boolean break byte case catch char class const
continue default do double else enum extends final finally float for
goto if implements import instanceof int interface long native new
package private protected public return short static strictfp super
switch synchronized this throw throws transient try void volatile while
```

Reserverade ord

Aritmetiskt uttryck	(x + Z) * ! / Z + ! % Z	för heltal är / heltalsdivision, % "rest"
Objektuttryck	new Classname(..) ref-var null function-call this super	
Logiskt uttryck	i cond cond && cond cond cond relationsuttryck true false	
Relationsuttryck	expr (< > <= >= == != < > <= >=) expr	för objektuttryck bara == och !=, också typtest med expr instanceof Classname
Funktionsanrop	obj-expr.method(..) (Classname.method(..))	anropa "vanlig metod" (utför operation)
Array	new int[size]	skapar int-array med size element
Matris	new int[i][j][k] m.length m[i].length	//Skapar matris med r rader och c kolonner //Ger matrisens längd (d.v.s. antalet rader) //Ger antalet element (längden) på raden i
Typkonvertering	(newType) expr (int) real-expr (Square) aShape	konverterar expr till typen newType - avkortar genom att stryka decimaler - ger ClassCastException om aShape inte är ett Square-objekt

Uttryck

return-sats	return expr;	returnerar funktionsresultat
do-while-sats	do { ... stmt; ... } while (cond) { stmt; ... }	utförs minst en gång, så länge cond är true
while-sats	while (cond) { stmt; ... }	utförs så länge cond är true
for-each-sats	for (int x: xs) { ... stmt; ... }	x blir ett element i taget ur xs fungerar även med array
for-sats	for (int i = a; i < b; i++) { ... stmt; ... }	satserna görs för i = a, a+1, ..., b-1 Görs ingen gång om a >= b !+ kan ersättas med i = i + step
switch-sats	switch (expr) { ... case A: stmt1; break; ... default: stmtN; break; }	expr är ett heltalsuttryck utförs om expr = A (A konstant) "faller igenom" om break saknas sats efter default: utförs om inget case passar
if-sats	if (cond) { stmt; ... } else { stmt; ... }	utförs om cond är true utförs om false
Förkortade	x += expr; x -= expr; även *=, /= x ++; även x --	variabeln och uttrycket av kompatibel typ
Block	{ stmt1; stmt2; ... }	fungerar "utföra" som en sats

Satser

Vertikalstreck | används mellan olika alternativa. Parenteser () används för att gruppera en mängd alternativa. Hakparenteser [] markerar valfria delar. En sats betecknas s tmt medan x, i, s, ch är variabler, expr är ett uttryck, cond är ett logiskt uttryck. Med ... avses valfri, extra kod.

Deklarationer

Allmänt	[<protection>] [static] [final] <type> name1, name2, ...;	
<type>	byte short int long float double boolean char Classname	
<protection>	public private protected	för attribut och metoder i klasser (paketskydd om inget anges)
Startvärde	int x = 5;	startvärde bör alltid anges
Konstant	final int N = 20;	konstantnamn med stora bokstäver
Array	<type>[] vname = new <type>[10];	deklarerar och skapar array
Matris	<type>[][] m = new <type>[4][5];	// deklarerar och skapar 4x5 matrisen m

Klasser

Deklaration	[public] [abstract] class Classname [extends Classname1] [implements Interface1, Interface2, ...] { <deklaration av attribut> <deklaration av konstruktörer> <deklaration av metoder> }	
Attribut	Som vanliga deklarerationer. Attribut får implicita startvärden, 0, 0.0, false, null.	
Konstruktör	<prot> Classname(param, ...) { stmt; ... }	Parametrarna är de parametrar som ges vid new Classname(...). Satserna ska ge attributen startvärden
Metod	<prot> <type> name(param, ...) { stmt; ... }	om typen inte är void måste en return-sats exekveras i metoden
Huvudprogram	public static void main(String[] args) { ... }	
Abstrakt metod	Som vanlig metod, men abstract före typnamnet och { . . . } ersätts med semikolon. Metoden måste implementeras i subclasserna.	

Standardklasser, java.lang, behöver inte importeras

Object	Superklass till alla klasser. boolean equals(Object other); int hashCode(); String toString();	ger true om objektet är lika med other ger objektets hashkod ger en läsbar representation av objektet
Math	Statiska konstanter Math.PI och Math.E. Metoderna är statiska (anropas med t ex Math.round(x)). long round(double x); int abs(int x); double hypot(double x, double y); double sin(double x); double exp(double x); double pow(double x, double y); double log(double x); double sqrt(double x); double toRadians(double deg);	avrundning, även float → int $ x $, även double, ... $\sqrt{x^2 + y^2}$ $\sin x$, liknande: cos, tan, asin, acos, atan e^x x^y $\ln x$ \sqrt{x} $deg \cdot \pi / 180$
System	void System.out.print(String s); void System.out.println(String s); void System.exit(int status);	skriv ut strängen s som print men avsluta med ny rad avsluta exekveringen, status != 0 om fel
	Parameter till print och println kan vara av godtycklig typ: int, double, ...	

Wrapperklasser

För varje datatyp finns en wrapperklass: char → Character, int → Integer, double → Double, ... Statiska konstanter MIN_VALUE och MAX_VALUE i klassen Integer ger minsta respektive största heltalsvärde. För klassen Double ger MIN_VALUE minsta flyttalet som är större än noll. Exempel med klassen Integer:

```
Integer(int value);           skapar ett objekt som innehåller value
int intValue();              tar reda på värdet
```

String

Teckensträngar där tecknen inte kan ändras. "asdf" är ett String-objekt. s1 + s2 för att konkatenera två strängar. StringIndexOutOfBoundsException om någon position är fel.

```
int length();                antalet tecken
char charAt(int i);          tecknet på plats i, 0..length()-1
boolean equals(String s);    jämför innehållet (s1 == s2 fungerar inte)
int compareTo(String s);     < 0 om mindre, = 0 om lika, > 0 om större
int indexOf(char ch);        index för ch, -1 om inte finns
int indexOf(char ch, int from); som indexOf men börjar leta på plats from
String substring(int first, int last); kopia av tecknen first..last-1
String[] split(String delim); ger array med "ord" (ord är följder av tecken åtskilda med tecknen i delim)
```

Konvertering mellan standardtyp och String (exempel med int, liknande för andra typer):

```
String.valueOf(int x);       x = 1234 → "1234"
Integer.parseInt(String s); s = "1234" → 1234, NumberFormatException om s innehåller felaktiga tecken
```

StringBuilder

Modifierbara teckensträngar. length och charAt som String, plus:

```
StringBuilder(String s);     StringBuilder med samma innehåll som s
void setCharAt(int i, char ch); ändrar tecknet på plats i till ch
StringBuilder append(String s); lägger till s, även andra typer: int, char, ...
StringBuilder insert(int i, String s); lägger in s med början på plats i
StringBuilder deleteCharAt(int i); tar bort tecknet på plats i
String toString();           skapar kopia som String-objekt
```

Standardklasser, import java.util.Classname

List	List<E> är ett gränssnitt som beskriver listor med objekt av parameterklassen E. Man kan lägga in värden av standardtyperna genom att kapsla in dem, till exempel int i Integer-objekt. Gränssnittet implementeras av klasserna ArrayList<E> och LinkedList<E>, som har samma operationer. Man ska inte använda operationerna som har en position som parameter på en LinkedList (i stället en iterator). IndexOutOfBoundsException om någon position är fel. För att operationerna contains, indexOf och remove(Object) ska fungera måste klassen E över-skugga funktionen equals(Object). Integer och de andra wrapperklasserna gör det.	
ArrayList LinkedList	ArrayList<E>(); LinkedList<E>(); int size(); boolean isEmpty(); E get(int i); int indexOf(Object obj); boolean contains(Object obj); void add(E obj); void add(int i, E obj); E set(int i, E obj); E remove(int i); boolean remove(Object obj); void clear();	skapar tom lista skapar tom lista antalet element ger true om listan är tom tar reda på elementet på plats i index för obj, -1 om inte finns ger true om obj finns i listan lägger in obj sist, efter existerande element lägger in obj på plats i (efterföljande element flyttas) ersätter elementet på plats i med obj tar bort elementet på plats i (efterföljande element flyttas) tar bort objektet obj, om det finns tar bort alla element i listan