ArrayList: Dynamisk lista med objekt av parameterklassen E.

paket: java.util

```
ArrayList<E>( );                    // skapa tom lista
int size( );                        // antal element
boolean isEmpty( );                 // undersök om listan är tom
E get(int index);                   // tag reda på elementet på plats index

void add(E obj);                    // lägg in obj sist (efter existerande element)
void add(int index, E obj);         // lägg in obj på plats index (efterföljande
                                    // element flyttas)
E set(int index, E obj);            // ersätt elementet på plats index med obj
E remove(int index);                // tag bort elementet på plats index
void clear( );                      // tag bort alla element i listan

int indexOf(Object obj);            // ger index för elementet Obj, –1 om inte finns
```

Hjälpklasser

Finns inte i Standard Java, men används i boken Objektorienterad programmering och Java.

SimpleWindow: SimpleWindow(int width, int height, String title); // skapa fönster

paket: se.lth.cs.window

```
int getWidth( );                    // tag reda på storlek i x-led
int getHeight( );                   // tag reda på storlek i y-led
void clear( );                      // radera innehållet
void close( );                      // stäng fönstret tillfälligt
void open( );                       // öppna fönstret igen

void moveTo(int x, int y);          // flytta pennan till x,y utan att rita
void lineTo(int x, int y);          // flytta och rita rät linje
int getX( );                        // tag reda på pennans position – x
int getY( );                        // – y
void writeText(String txt);         // skriv texten txt
void setLineWidth(int w);           // ändra linjebredd
void setLineColor(Color c);         // ändra färg: c = java.awt.Color.black, white,
                                    // blue, gray, green, red, yellow, …

void waitForMouseClick( );          // vänta tills användaren klickat med musen
int getMouseX( );                   // musens position vid musklick – x
int getMouseY( );                   // – y
int getMouseButton( );              // musknapp (1=vänster, 2=mitten, 3=höger)

void waitForEvent( );               // vänta tills händelse inträffat
int getEventType( );                // 1 = tangentnedtryckning, 2 = musklick
char getKey( );                     // tangent som trycktes ned

static void delay(int mSec);        // vänta mSec millisekunder
```

SimpleInput:

paket: se.lth.cs.util

```
char readChar( );                   // läs nästa tecken
int readInt( );                     // läs nästa heltal
double readReal( );                 // läs nästa reella tal
String readString( );               // läs nästa teckensträng fram till radslut
void readln( );                     // ignorera resten av raden
boolean eof( );                     // tag reda på om indata är slut
```

SimpleRandom:

paket: se.lth.cs.util

```
int randInt(int a, int b);          // rektangelfördelat heltal i [a,b]
double randReal(double a, double b);        // – reellt tal
boolean draw(double a);             // true med sannolikheten a
double normal(double avg, double sDev);  // – normalfördelat, medelvärde avg,
                                    // standardavvikelse sDev
```

Java snabbreferens – med generisk ArrayList

av Per Holm, Per.Holm@cs.lth.se

Här beskrivs stora delar av Java på ett kortfattat sätt. Tecknet ”|” står för ”eller”. Vanliga parenteser ”( )” används för att gruppera alternativ. Med ”[ ]” markeras sådant som inte alltid finns med.

Satser

```
Block:              {                               // fungerar ”utifrån” sett som en sats
                      sats1; sats2; …;              // även deklarationer är satser
                    }

Tilldelningssats:   variabel = uttryck;             // variabeln och uttrycket av kompatibel typ

Speciella tilldeln: variabel += uttryck;            // variabel=variabel+uttryck; även –=, *=, /=
                    variabel++;                     // variabel=variabel+1; även – –

if-sats:            if (villkor)                    // villkoret är ett logiskt uttryck
                      sats1                         // sats1 utförs om uttrycket är true
                    [ else
                      sats2 ]                       // sats2 utförs om uttrycket är false

switch-sats:        switch (x) {                    // x är ett heltalsuttryck
                      case A: sats1; break;         // utförs om x=A (A konstant)
                      case B: sats2; break;         // utförs om x=B
                      …
                      default: satsN; break;        // utförs om inget case utförs
                    }

for-sats:           for (int k = start; k<=slut; k++)   // satsen utförs för k = start, start+1, …,
                      sats                          // slut (ingen gång om start>slut)
                                                    // k++ kan ersättas med k = k+steg

while-sats:         while (villkor)
                      sats                          // utförs så länge villkoret är true

do-while-sats:      do
                      sats                          // utförs så länge villkoret är true
                    while (villkor);

return-sats:        return uttryck;                 // returnera funktionsresultat

Void-metod-anrop:   objektuttryck.metod( [param, …] ); // anropa ”vanlig” metod (utför operation)
                    Klassnamn.metod( [param, …] );  // – statisk metod
```

Uttryck

Aritmetiskt uttryck: Skrivs som i ”normal” matematisk notation. Operanderna är konstanter, variabler eller funktionsanrop. Dessutom ingår parenteser, operatorerna *, /, % rest, +, –. Om operanderna är heltal är / helstalsdivision dvs resultatet avkortas mot noll.

Objektuttryck: new Klassnamn([param, …]) | ref-variabel | null | funktionsanrop | this | super

Logiskt uttryck: ! logiskt-uttryck | logiskt-uttryck && logiskt uttryck | logiskt-uttryck || logiskt uttryck | funktionsanrop | relation | boolean-variabel | true | false

Relation: uttryck (< | <= | == | >= | > | !=) uttryck

(för objektuttryck bara == och !=, också uttryck instanceof Klassnamn)

Funktionsanrop: objektuttryck.metod([param, …]); // anropa ”vanlig” metod (utför operation)
Klassnamn.metod([param, …]); // – statisk metod

Vektor: new typ[antal] // skapa vektor med plats för antal st element
vektornamn.length // antalet element i vektorn

Typkonvertering: (nytyp) uttryck // konvertera uttrycket till angiven typ, t ex
(int) reellt-uttryck // avkorta genom att stryka decimaler
(Car) aVehicle // konvertera kvalifikation, går bra om
// aVehicle refererar till ett Car-objekt

Deklarationer

Allmänt: [<skydd>] [static] [final] <typ> namn1, namn2, …;

<typ> byte | short | int | long | float | double | boolean | char | Klassnamn

<skydd> public | protected | private // för attribut och metoder i klasser. Om skydd
// inte anges är storheten public inom aktuellt
// paket, private utanför

Startvärde: Deklaration med namn1 = värde1, namn2 = värde2, …

Konstant: Deklaration med final-modifierare och namn = värde.

Vektor: [ ] efter typnamnet i deklarationen

Klasser

Deklaration:
```
[public] [abstract] class Klassnamn
   [extends Klass1]  [implements Interface1, Interface2, …] {
   <Deklaration av attribut>
   <Deklaration av konstruktorer>
   <Deklaration av metoder>
}
```

Attribut: Som vanliga deklarationer. Attribut får implicita startvärden: 0, 0.0, false, null.

Konstruktor: [<skydd>] Klassnamn([<typ> param, …]) Block // anger vilka parametrar
// som ges vid new Klassnamn. I blocket
// skrivs satser som initierar objektets tillstånd

Metod: [<skydd>] (<typ> | void) metodnamn([<typ> param, …]) Block

Abstrakt metod: Som metod, men abstract före typnamnet och Block ersätts med semikolon. Implementationer av metoden måste finnas i subklasserna.

Standardklasser

Math: Statiska konstanter Math.PI och Math.E. Statiska metoder (anropas med t ex Math.round(x)):

```
long round(double x);              // avrundning, även float –> int
int abs(int x);                    // absolutvärde, även double, …
double sin(double x);              // liknande: cos, tan, asin, acos, atan
double exp(double x);              // e upphöjt till x
double pow(double x, double y);    // x upphöjt till y
double log(double x);              // naturliga logaritmen
double sqrt(double x);             // kvadratroten ur x
double random();                   // rektangelfördelat slumptal i [0,1]
```

System: Inläsning och utskrift (se också hjälpklassen SimpleInput):

```
int System.in.read( );                  // läs ett tecken, ger Unicode-numret
void System.out.print(String s);        // skriv strängen s
void System.out.println(String s);      // som print men avsluta med ny rad
void System.out.flush( );               // skicka upplagrade utskrifter till skärmen
                                        // (behövs bara efter print, ej efter println)
```

Parametern till print och println kan vara av godtycklig typ: int, double, …

Typklasser: Till varje datatyp finns en typklass: char –> Character, int –> Integer, double –> Double, osv. Statiska konstanter MIN_VALUE och MAX_VALUE ger minsta resp största värde. Operationer:

```
Type(type val);                 // skapa ett objekt innehållande val
Type(String s);                 // avkoda strängen s –> typen type
String toString( );             // skapa en läsbar representation
type typeValue( );              // tag reda på attributvärdet
```

Character: Statiska metoder i typklassen Character:

```
boolean isLetter(char ch);      // true om ch är en bokstav
boolean isDigit(char ch);       // true om ch är siffra
boolean isSpaceChar(char ch);   // true om ch är blank, tabulator eller ny rad
char toLowerCase(char ch);      // om ch är stor bokstav fås motsvarande lilla
                                // bokstav, annars ch
char toUpperCase(char ch);      // liten bokstav –> stor
```

String: Teckensträngar där de ingående tecknen inte kan ändras. ”asdf” är ett objekt av klassen String. s1+s2 konkatenering av två strängar.

```
String( );                      // tom sträng
String(String s);               // kopia av s

int length( );                  // antalet tecken i strängen
char charAt(int k);             // tecknet på plats k (0..length–1)

boolean equals(String s);       // jämför innehållet
int compareTo(String s);        // <0 om mindre, 0 om lika, >0 om större
String substring(int start, int stop); // kopia av tecknen start..stop–1
String trim( );                 // kopia utan inledande och avslutande blanka
String toLowerCase( );          // kopia med stora bokstäver utbytta mot små
String toUpperCase( );          // kopia med små bokstäver utbytta mot stora
```

StringBuffer: Modifierbara teckensträngar. Konstruktorer, length, charAt som String, plus:

```
void setCharAt(int k, char ch);     // tecknet på plats k sätts till ch
StringBuffer append(String s);      // lägg s sist. Även andra typer: int s, …
StringBuffer insert(int k, String s); // lägg in s med början på plats k

String toString( );                 // skapa kopia som String-objekt
```

Konvertering: 1) från annan typ till String:

```
String.valueOf(int x)           // x –> sträng, liknande för andra typer
```

2) från String till annan typ (exempel):

```
Integer.parseInt(String s)      // s (siffror) –> int-värde
Double.parseDouble(String s);   // s (siffror) –> double-värde
```