Programmierpraktikum

Claudius Gros, SS2012

Institut für theoretische Physik
Goethe-University Frankfurt a.M.

Java - Basic Data Types

primitive data types

- **`int, double, char, ...`** are primitive (predefined) data types
  arbitrary complex data types may be defined by the user
- **`String`** is (however) an object, like everything else

Typ	Size	Domain	Example
boolean	1	{true, false}	
short	2		
int	4	$[-2^{31}, 2^{31}-1] =$	
long	8	$[-2^{63}, 2^{63}-1]$	
float	4	$[1.40239846 \cdot 10^{-45}, 3.40282347 \cdot 10^{38}]$	
double	8	$[4.9406564584124654 \cdot 10^{-324}, 1.7976931348623157 \cdot 10^{308}]$	

- **short** / **long** for specialized applications only
- **float** not used anymore, standard is **double** for floating point operations
- 32/64 bits (**int**/**double**) used for memory addressing by 32/64 bit processors,
  $\rightarrow$ memory restrictions

example: primitive data types

- casting of one data type to another will be disussed later

```
 1. public class PrimitiveDataTypes {
 2.
 3.   public static void main(String[] args)
 4.     {
 5.
 6. // --- pi with double (16 digits) and float (8 digits)
 7. // --- Math.PI is a double constant
 8.     double double_PI = Math.PI;
 9.     float   float_PI = (float)Math.PI;    // casting double to float
10.
11.     System.out.printf("pi with double: %24.20f\n",double_PI);
12.     System.out.printf("pi with  float: %24.20f\n",float_PI);
13.     System.out.printf("%25s ^\n","");   // pointer
14.     System.out.println(" ");            // empty line
15.
16. // --- int and long integer numbers
17.     int   int_int = 1;
```

```
18.     long long_int = 1;
19.
20.     for (int i=0; i<34; i++)
21.       {
22.        int_int =   int_int * 2;
23.       long_int =  long_int * 2;
24.       System.out.printf("i+1, short, int, long: %4d %12d %12d\n",
25.                         i+1, int_int, long_int);
26.       }
27.
28.     }
29. }
```

strings

- a string (word) is an aribitrary sequence of characters
- the **String** class *java.lang.String*
  has many built-in member functions for string manipulation
- **String** concatenation with +

```
 1.
 2. public class ExampleStrings {
 3.   public static void main(String[] args)
 4.   {
 5.
 6. // --- defining and printing
 7. // --- toUpperCase() is a member function of the String class
 8.   String cdde = "cdde";
 9.   System.out.println("abc");
10.   System.out.println(cdde);
11.   System.out.println(cdde.toUpperCase());
12.
```

```
13. // --- String concatenation with '+'
14.   System.out.println("abc" + " " + cdde);
15.
16. // --- substring() is a member function of the String class
17.   String c  = "abc".substring(2,3);
18.   String dd = cdde.substring(1,3);
19.   System.out.println(" c: " + c);
20.   System.out.println("dd: " + dd);
21.
22. // --- valueOf() casts numbers into strings
23.   System.out.println(String.valueOf(Math.PI));
24.
25.   } // end of ExampleStrings.main()
26. } // end of class ExampleStrings
27.
```

casting vs parsing

- here: conversion of primitive data types and strings
- later: casting of objects

```
 1. import java.util.*;
 2. import java.math.*;
 3.
 4. public class CastingDemo {
 5.
 6. public static void main(String args[]) {
 7.
 8.    double rr = Math.random();
 9.    String ss = String.valueOf(rr);         // converting double into string
10.    System.out.println("a random number as a string: " + ss);
11.    System.out.printf( "                   and back: %f\n\n",
12.               Double.parseDouble(ss));     // parsing
13. //
14.    int ii = (int)(Math.random()*100.0);    // casting double into int
15.    System.out.printf( " random integer and string : %d %s\n",
```

```
16.                 ii, String.valueOf(ii));
17.
18.   }  // end of CastingDemo.main()
19. }    // end of CastingDemo
```

arrays

- *n*-component array

```
1. int[] array = new int[n];
```

- value of array-component *k*

```
1. int b = array[k];
```

- array with predefined values

```
1. int[] array = {1, 2, 3};
```

- length *n* of array

```
1. int b = array.length;
```

- multi-dimensional arrays

```
1. int[][] array = new int[n][m];
```

example: arrays

- arrays start always with *[0]*

```
 1. public class ExampleArrays {
 2.   public static void main(String[] args) {
 3.
 4. // --- array of integers
 5.     int[] values = {2, 3, 5, 7, 11, 13};
 6.     System.out.println(values[3]);     // 7
 7.
 8. // --- array of strings
 9.     String[] colors = {"yellow", "red", "green", "blue"};
10.     colors[0] = "black";
11.     for (int i=0; i<colors.length; i++)
12.       System.out.printf("This is color[%d]: %s \n", i, colors[i]);
13.     System.out.println(" ");           // empty line
14.
15. // --- splitting a string, typically used when reading from file
16.     String inputString = "age height weight";
17.     String[] words = inputString.split(" ");
```

```
18.     for (int i=0; i<words.length; i++)
19.       System.out.printf("This is words[%d]: %s \n", i, words[i]);
20.     System.out.println(" ");           // empty line
21.
22. // --- array of chars
23.   char[] chars = "black sun".toCharArray();
24.     for (int i=0; i<chars.length; i++)
25.       System.out.printf("This is chars[%d]: %c \n", i, chars[i]);
26.   }
27. }
```

example: matrices

- *matrix.length*: the number of rows
- *matrix[i].length*: the number of elements in row[i]
- the number of elements per row may be variable (ragged matrix)

```
 1. public class ExampleMatrices {
 2.   public static void main(String[] args)
 3.   {
 4.
 5. // --- defining a 3x2 matrix (3 rows and 2 columns)
 6.   String[][] strMatrix = new String[3][2];
 7.   String[] rowZero = { "rowZero[0]", "rowZero[1]"};
 8.   String[] rowOne  = { "rowOne[0]", "rowOne[1]"};
 9.   strMatrix[0] = rowZero;             // setting row [0]
10.   strMatrix[1] = rowOne;               // setting row [1]
11.
12. // --- printing matrix
13. // --- note: row[2] is undefined (null)
14.   for (int i = 0; i<strMatrix.length; i++)
```

```
15.     for (int j = 0; j<strMatrix[i].length; j++)
16.       System.out.printf("  strMatrix[%d][%d]:  %s\n",
17.                         i, j, strMatrix[i][j]);
18.   System.out.println(" ");
19.
20. // --- a matrix needs not to be rectangular, may be ragged
21.   int[][] intMatrix = new int[3][];    // allocate the number of rows
22.   int[] row_0 = {0,1,2,3,4};
23.   int[] row_1 = {5,6,7};
24.   int[] row_2 = {8,9};
25.   intMatrix[0] = row_0;
26.   intMatrix[1] = row_1;
27.   intMatrix[2] = row_2;
28.
29.   for (int i = 0; i<intMatrix.length; i++)
30.     {
31.     System.out.printf("  strMatrix[%d]:",i);
32.     for (int j = 0; j<intMatrix[i].length; j++)
33.       System.out.printf(" %d",intMatrix[i][j]);
34.     System.out.printf("\n");
35.     }
36.
37.   } // end of ExampleMatrices.main()
```

```
38. } // end of class ExampleMatrices
```

exercise: matrix operations

training suggestion

write a code for calculating the scalar product between two vectors and for matrix multiplication

- $n \times m$-Matrix

```
1. int[][] matrix = new int[n][m];
```

- value of M_{ij}

```
1. matrix[i][j]
```

- initialize matrix with predefined values

```
1. int[][] matrix = new int[][] {{1, 2, 3}, {4, 5, 6}};
```

write a code for evaluating the trace of a given matrix

- $tr(M) \coloneqq \sum_i M_{ii}$

arithmetic operators

- assignment and arithmetics

Symbol	Description	Examples
=	assignment	`int i=5;`
+ -	addition, subtraction	`int i=5+3; int i=5-3;`
* /	multiplication, integer division	`int i=5*3; int i=9/3;`
%	modulo	`int i=14%3;`
+ -	unary plus/minus	`int i=+3; int i=-2;`
+= -= *= /=	assignment and operation	`i += 5;` // i = i + 5
++ --	pre-/postfix increment/decrement	`i++;` // i = i + 1

- naming of C++

mathematical functions

import for basic mathematics

```
1. import java.lang.Math;
```

- $\pi \approx 3.14159265358979323846$
  $e \approx 2.71828182845904523 54$, Euler's number

```
1. static final double PI;
2. static final double E;
```

- absolute value $|x|$

```
1. static int abs(int x);
```

- minimum/maximum value

```
1. static int min(int x, int y);
2. static int max(int x, int y);
```

- (arcus-, hyperbolic-) sinus/cosine/tangent

```
1. static double sin(double x);
```

```
2. static double cos(double x);
3. static double tan(double x);
4. static double asin(double x);
5. static double acos(double x);
6. static double atan(double x);
7. static double hsin(double x);
8. static double hcos(double x);
9. static double htan(double x);
```

- square/cubic root

```
1. static double sqrt(double x);
2. static double cbrt(double x);
```

- exponential e^x

```
1. static double exp(double x);
```

- power x^y

```
1. static double pow(double x, double y);
```

- logarithm (base e/10)

```
1. static double log(double a);
2. static double log10(double a);
```

- round

```
1. static double round(double a);
2. static long round(double a);
```