# Programming using Java language

March, 20 – 24, 2006, Kaunas

dr Jaanus Pöial

Estonian Information Technology College, Tallinn

# Topics (8h)

- Data structures, collections
- Exceptions
- Threads
- Files

Website:
http://www.itcollege.ee/~jpoial/Kaunas/

# Data structures in Java

- Simple variables: primitive types and object types
- Arrays: base type, index, length
- Objects: encapsulate different fields into one instance (like records in “old” imperative languages + methods)
- Collections – built-in tools in Java API to manipulate group of objects: Vector, Hashtable, etc. Java collections framework

# Types

- Primitive types: byte, short, int, long, float, double, boolean, char
- Object types:
  - Wrappers: Byte, Short, Integer, Long, Float, Double, Boolean, Character
  - Other API types: String, Object, StringBuffer, …
  - Interface types: Comparable, Runnable, …

# Arrays

Array creation and initialization

```
int [] a = { 1, 5, 8 };
```

consists of 3 steps:

```
int [] a;                    // variable declared
a = new int [3];             // memory allocated
a[0]=1; a[1]=5; a[2]=8;      // values assigned
int n = a.length;            // array size
```

Array expression:

```
int [] a = new int [] { 1, 5, 8};
```

# Multi-dimensional arrays

- 2-dimensional array is an array of 1-dimensional arrays (NB! these can be of different length):

```
int [][] m;                         // 2-dim array
m = new int [2][];                  // first level
System.out.println (m.length);
m[0] = new int [4];                 // second level
m[0][0] = -8;
m[1] = new int [3];                 // different size
m[1][0] = 9;
```

# Objects

Object fields:

```
class Person {
    String surname;
    String firstName;
    Calendar birthDate;
    // etc. whatever we want to record
```

# Constructors

```
Person (String sn, String fn, Calendar bd) {
    surname = sn;
    firstName = fn;
    birthDate = bd;
} // constructor

Person() {
    this (“*”, “*”, Calendar.getInstance());
} // default constructor
```

# Instance methods

```
  public String toString() {
    return (firstName + " " + surname
      + " " + String.valueOf (
      birthDate.get (Calendar.YEAR)
      ) + " " + String.valueOf (
      birthDate.get (Calendar.MONTH)
      ) + " " + String.valueOf (
      birthDate.get(Calendar.DAY_OF_MONTH)));
  } // toString

} // Person
```

# Usage of objects

```
public class PersonMain {
   public static void main (String[] args) {
      Calendar bd1 = Calendar.getInstance();
      bd1.set (1959, 04, 30);
      Person p1 = new Person ("Smith",
         "John", bd1);
      System.out.println (p1);
      Person p2 = new Person();
      System.out.println (p2);
   } // main
} // PersonMain
```

# Collections

- *Collection*
  - *Set* (set, unique elements)
    - HashSet
      - LinkedHashSet
    - *SortedSet* (ordered set, unique elements)
      - TreeSet
  - *List* (dynamic, indexed, multiple copies allowed)
    - ArrayList
    - LinkedList
    - Vector (legacy API, similar to ArrayList)
  - *Queue* (Java 5, not discussed here)

# Collections

*Map* ("key-value" pairs)

- HashMap
  - LinkedHashMap
- *SortedMap*
  - TreeMap
- Hashtable (legacy API)
- WeakHashMap (allow garbage collection)

*Iterator* (to find the next element)

*Enumeration* (legacy API, similar to Iterator)

*Iterable* (has iterator)

- *Collection*

# Collections

*Comparable*

which of two elements is "bigger"

```
public int compareTo (Object o)
```

o1.compareTo (o2) =( -1, if o1 < o2 / 0, if o1 == o2 \ 1, if o1 > o2

Arrays

static utilities – asList, search, sort, fill, ...

Collections

static utilities – search, sort, copy, fill, replace, min, max, reverse, shuffle, ...

# Examples

- Use cases
- Word counter

# Homework

- Write static methods to calculate union and intersection of two sets (of type HashSet)