Java Programming

Albert Weichselbraun

Agenda

- Lecture aims
- Schedule and assessment
  - PC7 accounts and passwords
- Introduction to the program environment (Dia, Eclipse)
- Java Collections

Lecture aims

- repetition and verification of the required programming and modeling skills
- introduction of the game framework
- software project
  - requirement specification
  - project plan
  - modeling
  - implementation
  - deployment and presentation

Schedule

- **8 March 2011**: introduction, Java Collections
- **15 March 2011**: Java GUIs with Swing
- **22 March 2011**: introduction of the project framework; Java 2D; assessment of the programming topics
- **29 March 2011**: test, discussion of the requirement specification
- **5 April 2011**: completed project plan and modelling
- **31 May 2011**: presentation of the software project

Assessment:

- participation and work in labs: 20%
- test: 25%
- software project: 55%
  - documentation: 20%
  - implementation: 35%

Introduction of the Software

- Dia
- Eclipse
  - create your first Java project
  - classes and interfaces
  - running Java applications
  - auto completion, refactoring

Java Collection

- lists, sets and maps - different implementations of these types
- examples:
  - lists: ArrayList, Vector
  - Set: HashSet
  - Map: HashMap
- used to program aggregation/composition (compare UML diagram)

```
1  package simplecollection1;
2  import java.util.*;

4  // Java Collection and Iterator example
5  public class SimpleCollection {

7    public static void main(String[] args) {
8      List<String> testList= new ArrayList<String>();
9      testList.add("Ta'id Holmes");
10     testList.add("Vicky Weichsler");
11     testList.add("Christian Toth");
12     Collections.sort( testList );

14     // option 1: compact output Java 5.0
15     System.out.println("Java 5.0");
16     for (String name: testList) {
```

```
17          System.out.println(name);
18        }

20        // option 2: indices
21        System.out.println("Indices:");
22        for (int i=0; i<testList.size(); i++) {
23          System.out.println(testList.get(i));
24        }

26        // option 3: iterators
27        System.out.println("Iterators:");
28        Iterator<String> it = testList.iterator();
29        while (it.hasNext()) {
30          System.out.println(it.next());
31        }
32    }
```

```
33 }
```

The Comparable Interface

- implementing the `Comparable` Interfaces allows sorting using `Collections.sort()`
- many data types have a natural order and therefore implement a default version of the comparable interfaces
  (e.g. Integer, String alphabetized, . . .)
- Examples: phone book, accounting, . . .

```
1  package collections2;

3  public class Address implements Comparable<Address>{
4    // a simple address demo class
5    private String givenName, surname, phone;

7    public Address(String givenName, String surname,
8        String phone) {
9      this.givenName=givenName;
10     this.surname=surname;
11     this.phone=phone;
12   }
13   public String toString() {
14     return this.givenName+" "+this.surname +
15       ":"+this.phone;
16   }
```

```
17    public int compareTo(Address a) {
18      /* this method implements the Comparable
19       * Interface yielding
20       *    -1  ... if this is smaller than a,
21       *     0  ... if this equals a, and
22       *    +1  ... if this is larger than a
23       */
24      return this.surname.compareTo(
25          a.surname );
26    }
27  }
```

```
1  package collections2;
2  import java.util.*;

4  // Java Collections and Iterators example
5  public class PhoneBook {

7    public static void main(String[] args) {
8      List<Address> phoneList= new ArrayList<Address>();
9      phoneList.add( new Address("Ta'id",
10         "Holmes", "+43 1 58801-49360") );
11     phoneList.add( new Address("Vicky",
12         "Weichsler", "+43 1 783 212") );
13     phoneList.add( new Address("Christian",
14         "Toth", "+43 1 587 27 63") );

16     Collections.sort( phoneList );
```

```
18        // output
19        for (Address a: phoneList) {
20          System.out.println(a);
21        }
22      }
23  }
```

Exercise: Accounting Application

1. Create a class `Transaction`. Each transaction contains a date (String value of type 2010-10-10), a value (e.g. 13.5) and a description (e.g. rent).
2. the program handles two types of transactions: revenues and expenditures. Expenditures contain a numeric cost center number, revenues a text field with the customer's name. Implement the two corresponding classes `Expenditure` and `Income`.
3. implement an accounting application using the two classes (Expenditure and Income). The application should list (i) the total number of transactions, (ii) the number of expenditure and income transaction, (iii) the total sum of expenditures and incomes, and (iv) the total account balance.
4. implement the `Comparable` interface for the class `Transaction` to sort the transactions by their date.