Informatik II (D-ITET)

Tutorial 8

TA: Anwar Hithnawi, E-mail: hithnawi@inf.ethz.ch
Distributed Systems Group, ETH Zürich
Administrative

Substitution for the Tutorial Sessions on 1\textsuperscript{st} of May and 22 of May

- 1\textsuperscript{st} of May: Visit Thursday’s tutorials (*Check course website) in case of conflicts.
  - For questions on Exercise Sheet #9 visit office hours (10:00 am, Monday 6\textsuperscript{th} May or per Email)
  - Sheet 8 solution discussion will be combined with Sheet 9 in the following week
- 22 of May: Visit other tutorials that take place at the same time on Wednesday (*Check course website for more information on location)

*http://www.vs.inf.ethz.ch/edu/FS2013/I2/*
Outlook

- Exercise 7: Solution discussion

- Exercise 8: Overview (Generics, Binary Search, Game Tree, Reversi)
For-each loop

The basic for loop was extended in Java 5 to make iteration over arrays and other collections more convenient

- Arrays and Collections
  - for-each is commonly used to iterate over an array or a Collection of type `E`.
    - It can also iterate over anything that implements the `Iterable<E>` interface.
  - Many of the Collections classes (e.g., `ArrayList`) implement `Iterable<E>`, which makes the for-each loop very useful. You can also implement `Iterable<E>` for your own data structures.

http://leepoint.net/notes-java/flow/loops/foreach.html
http://download.oracle.com/javase/tutorial/collections/interfaces/index.html
for-each loop

Example:

```
for( type var : arr )
{
    //body of loop
}
```

```
for( int i = 0; i < arr.length; i++)
{
    type var = arr[i];
    //body of loop
}
```

```
for( type var : coll )
{
    //body of loop
}
```

```
for( Iterator<type> iter = coll.iterator();
    iter.hasNext(); )
{
    type var = iter.next();
    //body of loop
}
```

empty!
for-each loop

Interface Collection<T>

We say: “for each currentThing (of type T) in myCollectionOfThings (do) funnyMethod”

```java
Collection<T> myCollectionOfThings = ... ; // fill with objects

for( T currentThing : myCollectionOfThings )
{
    funnyMethod( currentThing );
}
```
for-each loop

Although the enhanced for loop can make code much clearer, it can't be used in some common situations.

- **Only access.** Elements can not be assigned to, e.g., not to increment each element in a collection.
- **Only single structure.** It's not possible to traverse two structures at once, e.g., to compare two arrays.
- **Only single element.** Use only for single element access, e.g., not to compare successive elements.
- **Only forward.** It's possible to iterate only forward by single steps.
- **At least Java 5.** Don't use it if you need compatibility with versions before Java 5.
Solution Ex7.Q1 – IFilter

Possible solution with for-each loops

```java
public ArrayList<Student> filterGeneric( ArrayList<ArrayList<Student>> groups )
{
    ArrayList<Student> result = new ArrayList<Student>();

    for(ArrayList<Student> group : groups )
        for( Student student : group )
            if( filter( student ) )
                result.add( student );

    return result;
}
```
Outlook

- Exercise 7: Solution discussion
- Exercise 8: Overview (Generics, Binary Search, Game Tree, Reversi)
Hints to Sheet 8

- Generics (Part 2)
- Binary Search
- Game Tree from Tic-Tac-Toe
- Reversi (Part 2)
More Generics: extends

- Example from last tutorial session
  - class MyPair<T> { public T first, second; }
  - An Object pair from the type MyPair<Float> contains two Float-References: pair.first and pair.second

- Sort by MyPairs after pair.first?
  - Already seen: After Comparable casten
    - Requires many Checks (or Exception-handling)
  - Better
    - class MyPair<T extends Comparable<T>> { ... }
    - For T only types can be used, the Comparable<T> implements (e.g. Float, Integer, String)
    - pair.first is now determined int compareTo(T other)
    - Without Casts
    - Compiler treats all exceptions (Compilation errors)
More Generics: Maps

- Often indexed data is needed
  - Particular for AHV-Number (unique id)
  - Document Identifiers (e.g. file name)
  - ...

- Such data structures are called Maps
  - Identifier (key) is "mapped" on content (value)
  - In the java standard library
    - interface Map<Key extends Comparable<Key>, Value>
    - Implementation: TreeMap, HashMap, ...

Hints to Ex8.Q1 – Binary Search

- Binary Search Algorithm (illustrated)

- Decision Tree

query: 16
sub-array-length: 1
return: null

query: 23
sub-array-length: 1
return: "23"
Hints to Ex8.Q1 – Binary Search

- Draw the decision tree and make some thoughts
  - Superposition, factors

- Implementation:
  - `find(List<Unit<Key, Value>> haystack, Key needle)`
  - `setFactor(int factor)`
    - Generalize the search → unbalanced search trees
  - `getNumberOfCalls()`
    - Benchmarking with various factors
    - Average # of recursive calls to various factors
Hints to Ex8.Q2 – Tic-Tac-Toe

- Reflections on game trees…
- Think about how the attribute of a node is calculated based on the attributes of the successor, when you are for example in your opponent’s place
Hints to Ex8.Q3 – Reversi (Part 2)

**HumanPlayer**

nextMove()

Wait for input from The command line

**RandomPlayer**

nextMove()

Random selection (but valid move!) Next move

**GreedyPlayer**

nextMove()

Select the next move using a simple, None-recursive Evaluation function

Download

Excercise 7

Excersise 8
Hints to Ex8.Q3 – Reversi (Part 2)

a. Implementing ICheckMove without Framework-Function.
   Ideas?
   Learn from GameBoard class. What methods available what could be useful?

b. Implement a player that selects the best move among all possible moves
   
   Best move: Move, after it's execution one owns max. more stones than the opponent: «Depth = 1»
   
   Given: Game Tree is not needed!

   Determining the best move: Copy Board (clone), run the move, count...
Hints Ex8.Q3a – checkMove()

```java
boolean checkMove(GameBoard gb, int player, Coordinates c) {
    //Check all directions

    //Unless at least one direction is valid
    //...

    //GameBoard.checkMove is not allowed to be used!
}
```
Hints Ex8.Q3a – greedyPlayer()

- Simple computer opponents
  - Move selection: Best first
  - Search depth: 1 (my move)
  - Evaluation Function: The difference of stone numbers after the move

- Tips
  - A GameBoard can be copied with `gb.clone()`
  - You are allowed to use `GameBoard.checkMove()` here
Have Fun!