Recap

- Git: Open source software to manage versions
- Commit: One specific version that knows its predecessor
- Branch: Multiple different commits can have the same predecessor, allowing parallel development
- Merging
  - Re-integrate parallel development
  - Mostly automatic, but sometimes not

How to Ask for Technical Support

Howto

- You may need to write to various people to get technical support
- Take a moment to think before clicking “send”

How to Ask for Technical Support

Howto

- You may need to write to various people to get technical support
- Take a moment to think before clicking “send”

Ensure that

- you make it easy for the other person
  - e.g., by including information the other person might first need to look up
- all relevant information is given (as far as you know)
- you use proper terminology (as far as you can)
- the context is still conceivable
  - I.e., click on reply instead of writing a new mail, keep the old mail text in there
- references in your text are clear
  - For instance: “this exercise” is not a clear reference
- you’re concise – long e-mails tend to be put on the read-later-pile (which may never happen)

UNIVERSITÄT
ZU KÖLN

Session 4: Iterable and Iterators

Fortgeschrittene Programmierung (Java 2)

Nils Reiter
nils.reiter@uni-koeln.de

8. Mai 2024

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Section 1

Introduction and Motivation

Iterating

- Programs with only single variables are not very powerful
- Power comes from possibility to group things of the same type
  - E.g., arrays: `int[] myArray = new int[1,2,3,4,5,6,7,8,9];`

Iterating

- Programs with only single variables are not very powerful
- Power comes from possibility to group things of the same type
  - E.g., arrays: `int[] myArray = new int[1,2,3,4,5,6,7,8,9];`
- Arrays allow treating many things the same way, because they have a common name
  - E.g.: `myArray[i] = myArray[i] * 2;`

Iterating

- Programs with only single variables are not very powerful
- Power comes from possibility to group things of the same type
  - E.g., arrays: `int[] myArray = new int[1,2,3,4,5,6,7,8,9];`
- Arrays allow treating many things the same way, because they have a common name
  - E.g.: `myArray[i] = myArray[i] * 2;`
- For this, we need a method to *iterate* over the elements of the array
  - E.g.: `for (int i = 0; i < myArray.length; i++) { }`

Iterating

- Programs with only single variables are not very powerful
- Power comes from possibility to group things of the same type
  - E.g., arrays: `int[] myArray = new int[1,2,3,4,5,6,7,8,9];`
- Arrays allow treating many things the same way, because they have a common name
  - E.g.: `myArray[i] = myArray[i] * 2;`
- For this, we need a method to *iterate* over the elements of the array
  - E.g.: `for (int i = 0; i < myArray.length; i++) { }`
- Iterating is such a central activity that Java offers different ways to do it
- `for (...) {...}`, `while (...) {...}`, `do {...} while (...)`

Loops: `for` and `while`

- How to decide which loop to use?

Loops: `for` and `while`

- How to decide which loop to use?
- No technical difference, it's about *code clarity*
  - I.e., for future code readers, potentially yourself

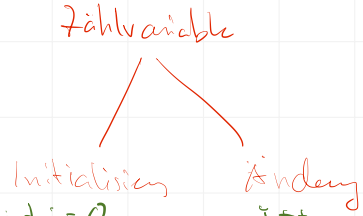
Loops: `for` and `while`

- How to decide which loop to use?
- No technical difference, it's about *code clarity*
  - I.e., for future code readers, potentially yourself

Example

```
for (int i = 0; i < myArray.length; i++) { ... }
int i = 0; while (i < myArray.length) { i++; ... }
```

What are Key Elements of Any Loop?



Bedingung
i < array.length

for(; Festplatte nicht voll Platz ;)
{
Daten speichern
}

Real-Life Problems

Example (File Search)

- 1000s of files
- Search term is a single word
- We're interested in the first file with the word

Real-Life Problems

Example (File Search)

- 1000s of files
- Search term is a single word
- We're interested in the first file with the word
- Solution so far
  - Create an array with all contents of the files
  - Iterate over the array
  - Return the one we want, disregard all others

Real-Life Problems

Example (File Search)

- 1000s of files
- Search term is a single word
- We're interested in the first file with the word
- Solution so far
  - Create an array with all contents of the files
  - Iterate over the array
  - Return the one we want, disregard all others
- Wasteful: Most file contents will probably never be read
- Incomplete: A file might be added on disk after having created the array
- Better: After inspecting each file, see if you need to load another

Section 2

Iterator

Iterator

- An interface in the Java library: `java.util.Iterator`
- A iterator iterates once over a collection of objects

java.util.Iterator

Iterator

- An interface in the Java library: `java.util.Iterator`
- A iterator iterates once over a collection of objects
- Four methods (two have a default implementation):

  `boolean hasNext()`: Returns `true` if there are more elements in the sequence

  `E next()`: Returns the next element in the collection

  `void remove()`: Removes the last element returned (optional)

  `void forEachRemaining(Consumer<? super E> a)`: Applies action to elements not yet returned (optional)

java.util.Iterator

Iterator

- An iterator object represents a specific iteration over a specific collection
- Iterators can (mostly) not be used twice
- Iterators are most naturally used in combination with while-loops:

```
1 Iterator iter = ...
2 while(iter.hasNext()) {
3     Object myObject = iter.next();
4 }
```

System.out.println(myObject.name);

Iterator

- An iterator object represents a specific iteration over a specific collection
- Iterators can (mostly) not be used twice
- Iterators are most naturally used in combination with while-loops:

```
1 Iterator iter = ...
2 while(iter.hasNext()) {
3     Object myObject = iter.next();
4 }
```

Benefits

- We only inspect/load as many elements as needed
- Object-oriented iteration: The iterator object represents the iteration itself
- Iterators make iterating easier (and object oriented) – they do not add something what would be impossible otherwise

demo

Student.java, StudentIterator.java

Section 3

Iterable

Iterable

- An interface in the Java library: `java.lang.Iterable`
- Provides a single (non-default) method: `Iterator<T> iterator()`
  - I.e.: the method returns an Iterator

Iterable

- An interface in the Java library: `java.lang.Iterable`
- Provides a single (non-default) method: `Iterator<T> iterator()`
  - I.e.: the method returns an Iterator
- An object that implements `Iterable`
  - is iterable, i.e., can be iterated on
  - can be used in a for-loop like this:

```
1 for (Object o : myIterable) {
2     o.doSomething();
3 }
```

demo

Generics

Topic for next week, but:

- Some classes are written with angle brackets: `Iterator<Student>` / `Iterable<Student>`
- Angle brackets contain the type that we iterate over
- This allows us to re-use the same code to iterate over different tyes!

Exercise

https://github.com/idh-cologne-java-2-summer-2024/exercise-04