Parallele Programmierung FS25

Exercise Session 2

Jonas Wetzel

Plan für heute

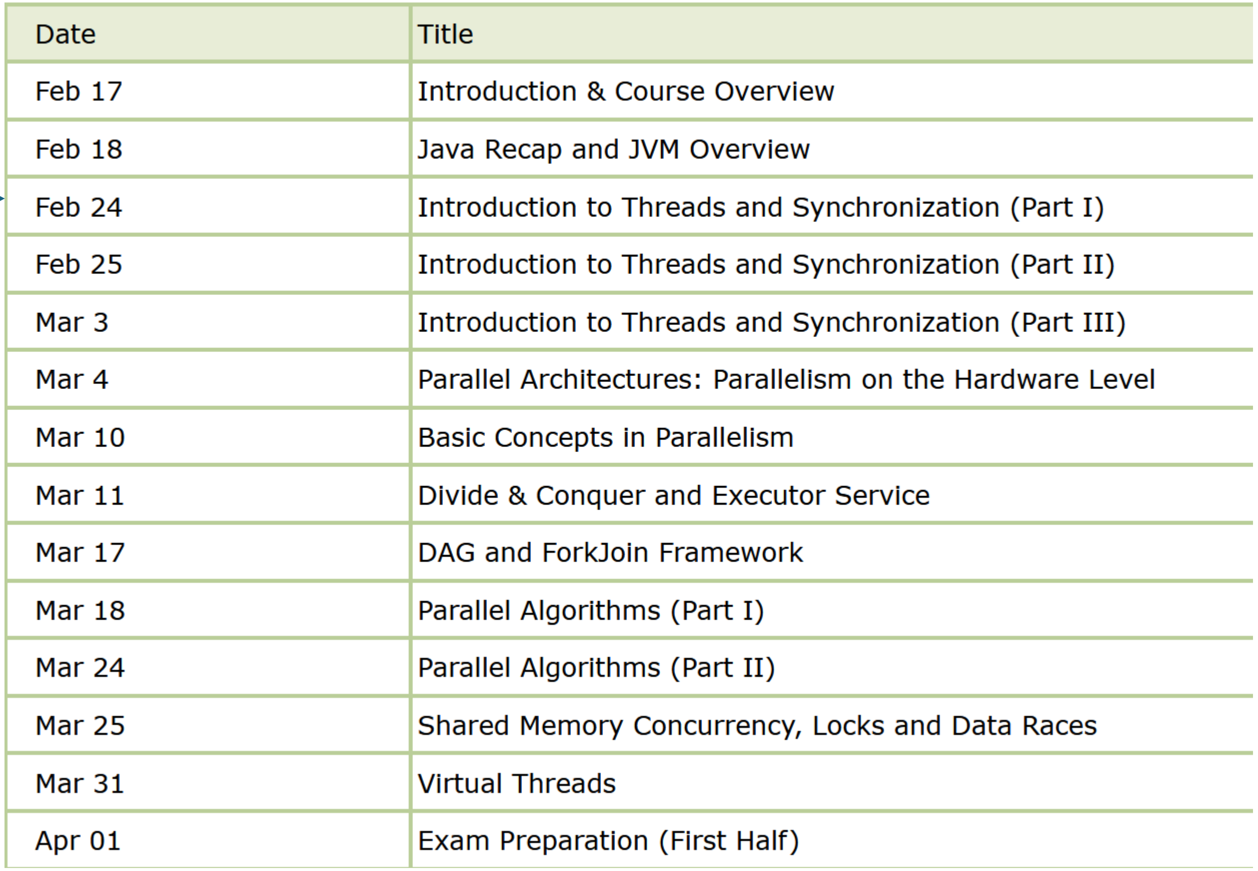
- Organisation
- Theory Recap
- Einstieg in Exercise 2
- Demo
- Kahoot
- Exam Questions

Organisation

- Mein Name ist Jonas Wetzel
- Meine Website (Materialien und Inhalt der Übungen): n.ethz.ch/~jwetzel
- Meine Email: jwetzel@ethz.ch
- Discord: @jonas.too

Organisation

- Wo sind wir jetzt?



Date	Title
Feb 17	Introduction & Course Overview
Feb 18	Java Recap and JVM Overview
Feb 24	Introduction to Threads and Synchronization (Part I)
Feb 25	Introduction to Threads and Synchronization (Part II)
Mar 3	Introduction to Threads and Synchronization (Part III)
Mar 4	Parallel Architectures: Parallelism on the Hardware Level
Mar 10	Basic Concepts in Parallelism
Mar 11	Divide & Conquer and Executor Service
Mar 17	DAG and ForkJoin Framework
Mar 18	Parallel Algorithms (Part I)
Mar 24	Parallel Algorithms (Part II)
Mar 25	Shared Memory Concurrency, Locks and Data Races
Mar 31	Virtual Threads
Apr 01	Exam Preparation (First Half)

Motivation

Wieso brauchen wir paralleles Programmieren?

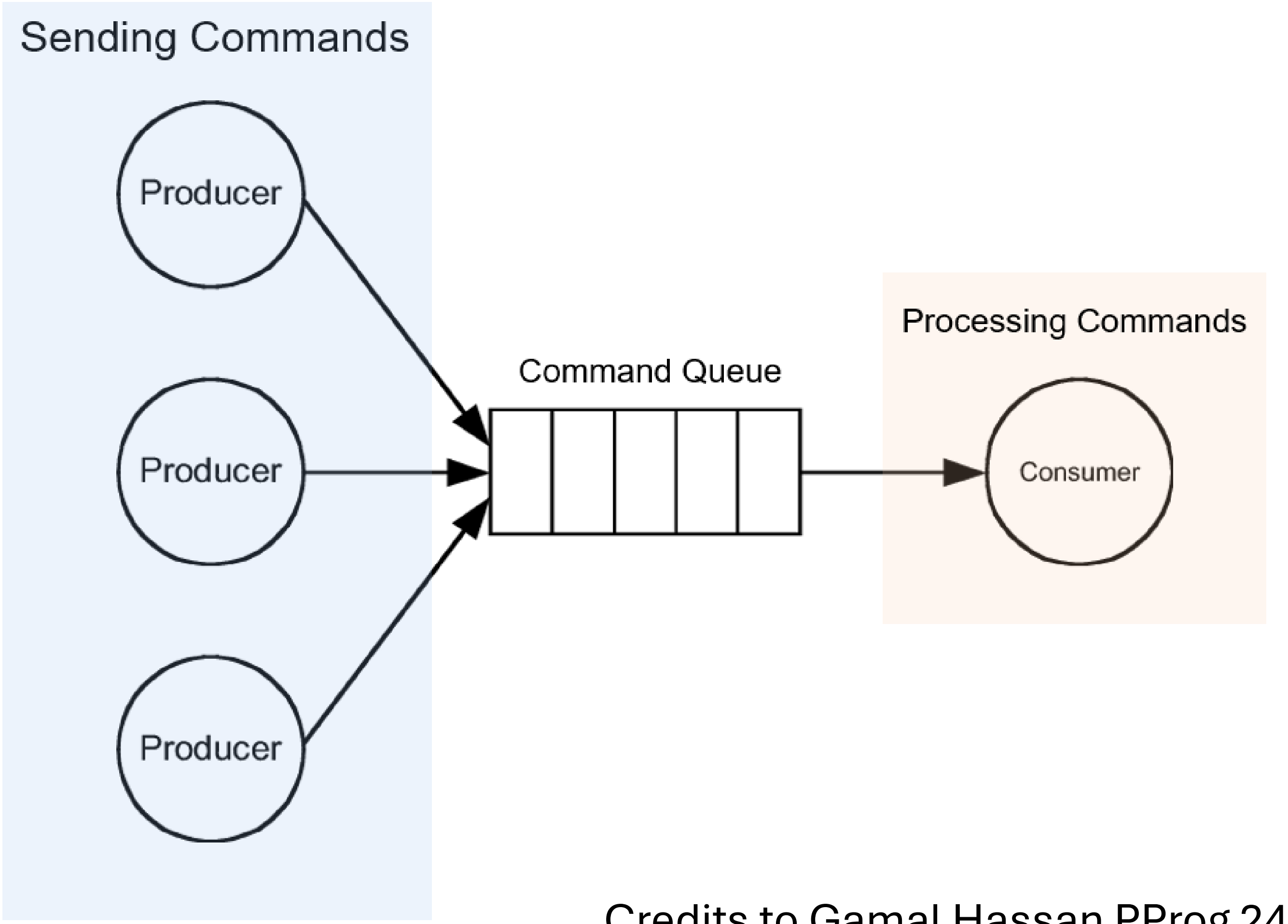
- Stellen wir uns vor, wir haben einen Discord Bot programmiert
- Nimmt commands wie !pprog, !info an
- Verarbeitet den Command und antwortet

Wieso brauchen wir paralleles Programmieren?

- Stellen wir uns vor, wir haben einen Discord Bot programmiert
- Nimmt commands wie !pprog, !info an
- Verarbeitet den Command und antwortet

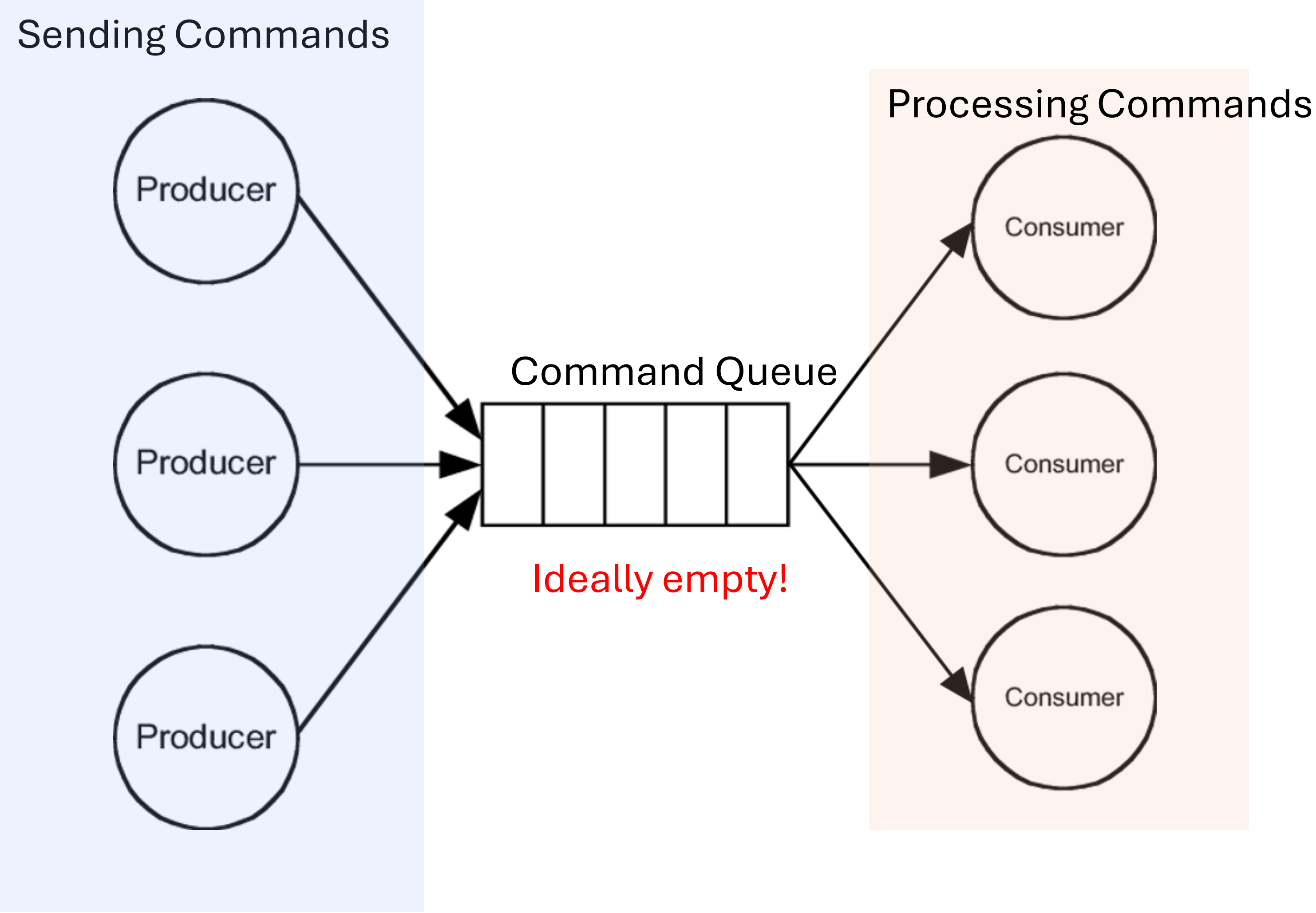
- **Problem:** Beim testen des Bots funktioniert alles einwandfrei mit geringer Latency. Nachdem der Bot aber veröffentlicht wird dauert es sehr lange bis der Bot auf Commands reagiert. Wieso?

Produce Consumer Problem



Credits to Gamal Hassan PProg 24

Solution? More Threads!



Ideally empty!

Credits to Gamal Hassan PProg 24

Theory Recap

Terminology

Overview:
https://cgl.ethz.ch/teaching/parallelprog25/pages/terminology.html

Thread Definition

An independent (i.e., capable of running in parallel) unit of computation that executes code.

Each thread is like a running sequential program,

but a thread can create other threads

that are then part of the same program.

Those threads can create more threads etc.

Thread Definition Advanced

Concept of threads exists on various levels:

- Hardware (CPU)
- Operating systems
- Programming languages
  - Java: `Thread` class

Thread Properties (in our course)

- Threads can create other threads
- **Shared memory** (changes to variables by threads are visible to other Threads)
- Threads (from same class) execute same program *but* with different arguments
- Communication between threads: Writing fields of shared objects

custom

Create Java Threads: Option 1 (oldest)

Instantiate a subclass of `java.lang.Thread` class

- Override run method (must be overridden)
- run() is called when execution of that thread begins
- A thread terminates when run() returns
- start() method invokes run()
- Calling run() does not create a new thread

```
class ConcurrWriter extends Thread { …
    public void run() {
        // code here executes concurrently with caller
    }
}
ConcurrWriter writerThread = new ConcurrWriter();
writerThread.start();   // calls ConcurrWriter.run()
```

Creating the Thread object does not start the thread!

Need to actually call start() to start it.

Create Java Threads: Option 2 (better)

Implement `java.lang.Runnable`

- Single method: public void run()
- Class implements `Runnable`

```
public class ConcurrWriter implements Runnable {
     …
     public void run() { …
        // code here executes concurrently with caller
      }
}

ConcurrWriter writerThread = new ConcurrWriter();
Thread t = new Thread(writerThread);
t.start();    // calls ConcurrWriter.run()
```

Alternativ

```
Runnable incrementTask = new
Runnable() {
    @Override
    public void run() {
        for (int i = 0; i < 10000; i++) {
            counter++;
        }
    }
};
```

```
Runnable incrementTask = () -> {
    for (int i = 0; i < 10000; i++) {
        counter++;
    }
};
```

Why is it better?

- In Java, a class can extend only one class. By implementing Runnable, your class remains free to extend another class if needed.

- See code example

Java Threads: some key points

Every Java program has at least one execution thread

- First execution thread calls `main()`

Each call to **`start()`** method of a `Thread` object creates an actual execution thread

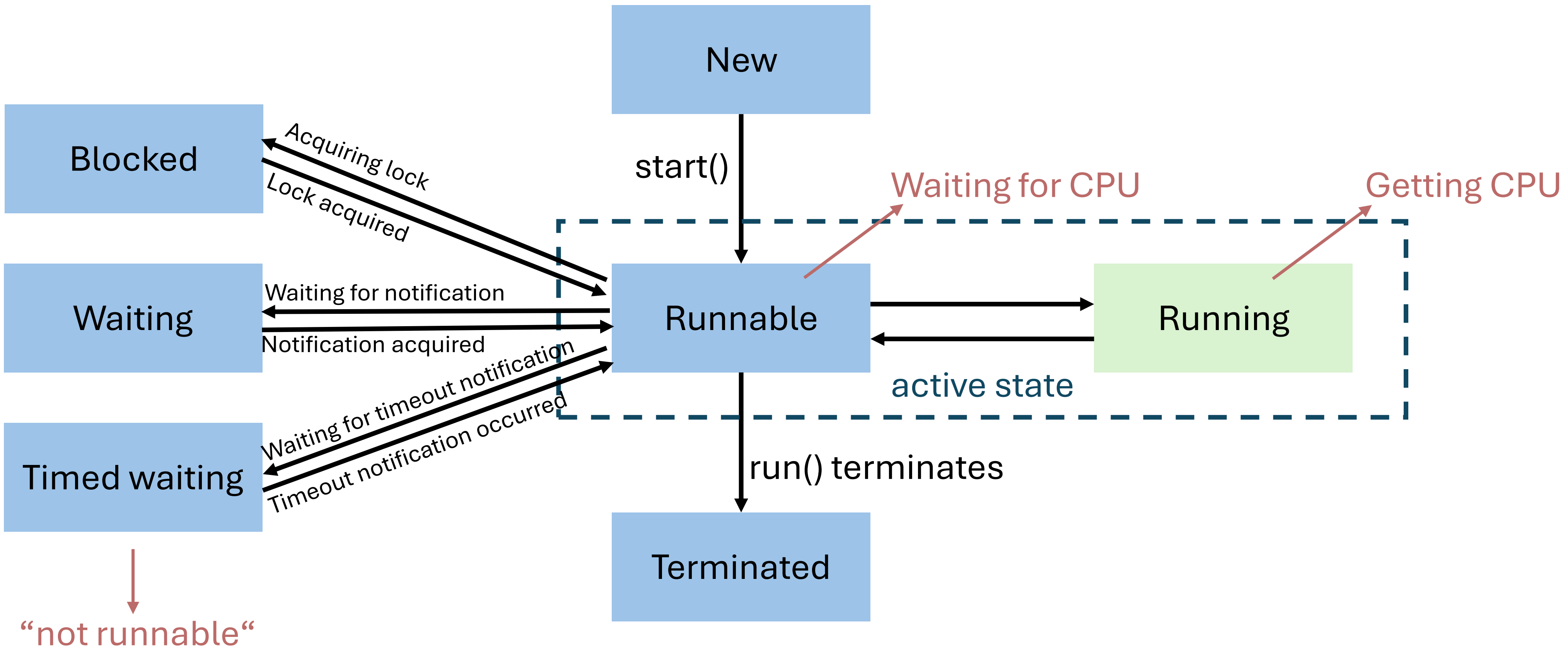
Program ends when all threads (non-daemon threads) finish.

Threads can continue to run even if `main()` returns

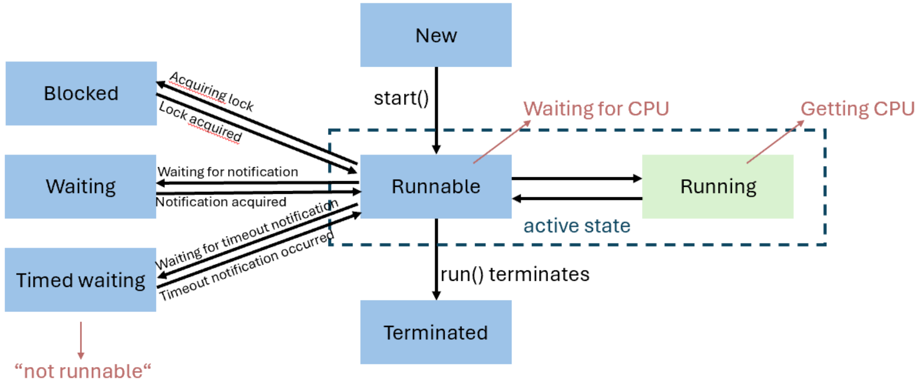
Creating a `Thread` object does not start a thread

Calling `run()` doesn't start thread either (need to call start()!)

Life cycle of a Thread



Thread State Model



NEW: The thread has been created but not yet started.

RUNNABLE: The thread is ready to run as soon as it gets CPU time.

RUNNING: The tread is running.

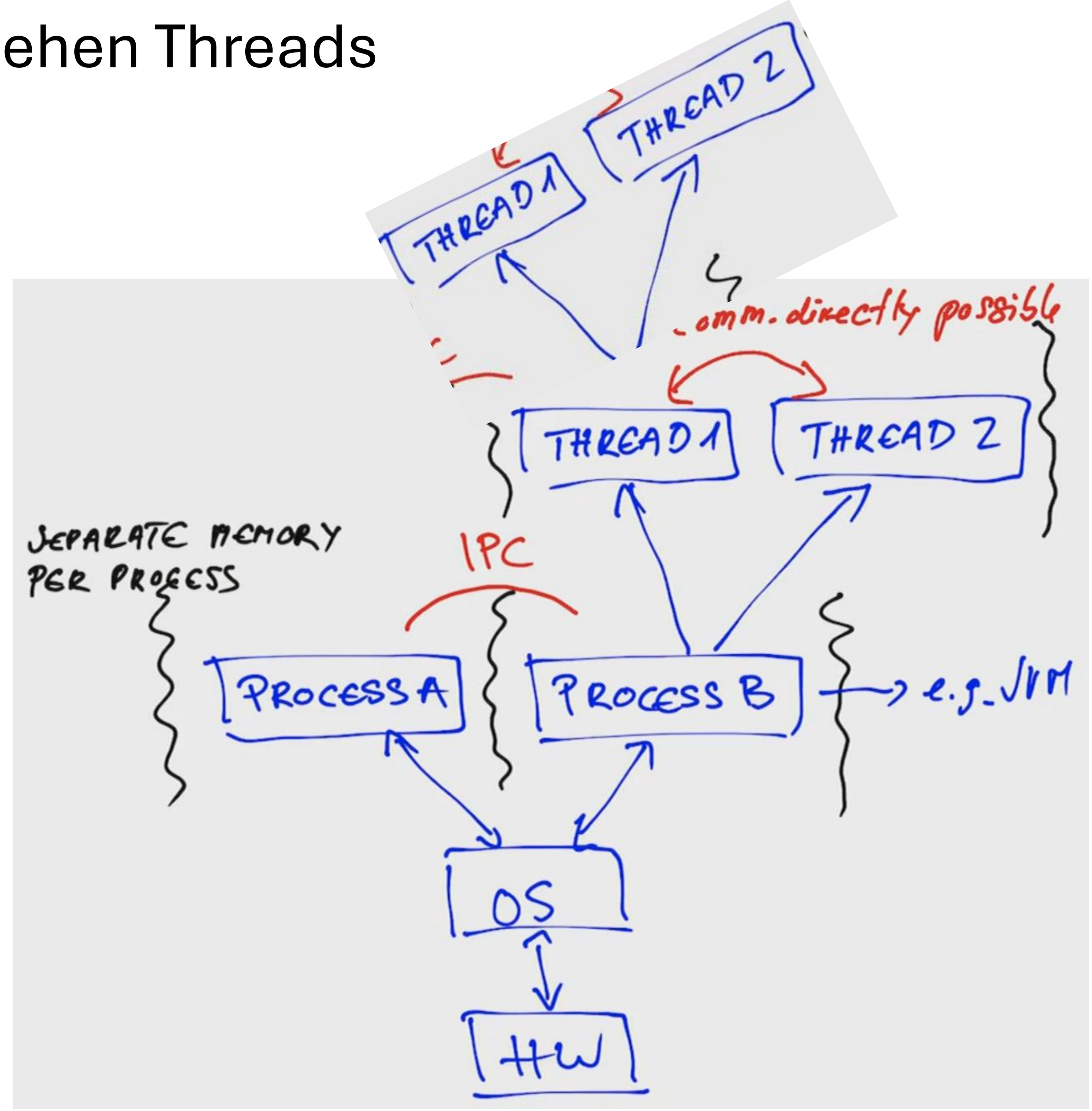
BLOCKED: The thread is waiting to acquire a monitor lock (for example, trying to enter a synchronized block/method).

WAITING: The thread is waiting indefinitely for another thread to perform a particular action (e.g., calling Object.wait() without a timeout).

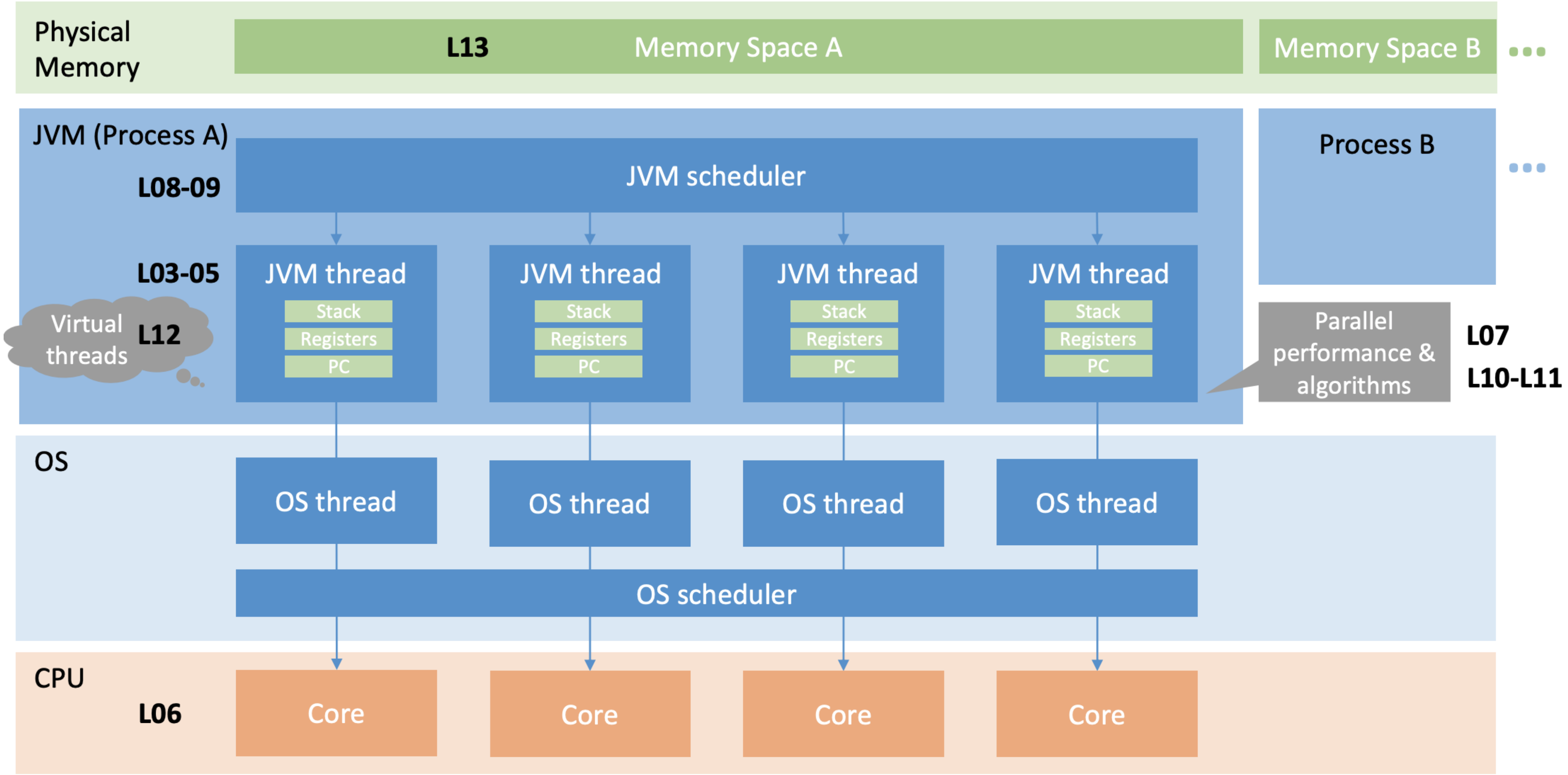
TIMED_WAITING: The thread is waiting for another thread’s action for up to a specified period (e.g., Thread.sleep(1000) or wait(1000)).

TERMINATED: The thread has completed its execution.

Was sehen Prozesse und was sehen Threads



Big Picture


Physical Memory
L13
Memory Space A
Memory Space B
...
JVM (Process A)
L08-09
JVM scheduler
Process B
...
L03-05
JVM thread
Stack
Registers
PC
JVM thread
Stack
Registers
PC
JVM thread
Stack
Registers
PC
JVM thread
Stack
Registers
PC
Virtual threads
L12
Parallel performance & algorithms
L07
L10-L11
OS
OS thread
OS thread
OS thread
OS thread
OS scheduler
CPU
L06
Core
Core
Core
Core

(Bad) Interleavings, what can go wrong?

Assume we have two threads executing increment() n-times concurrently.

```
• public class Counter {
•       int count = 0;

•       public void increment() {
•               count++;
•       }
• }
```

Bad Interleaving

- Siehe code example

Bad Interleaving

- Was ist passiert?

(Bad) Interleavings, what can go wrong?

Assume we have two threads executing increment() n-times concurrently.

```
public class Counter {
    int count = 0;

    public void increment() {
        count = count + 1;
    }
}
```

Bad Interleaving

- Count++ ist äquivalent zu count = count + 1
- Read, increment and write back -> separate Handlungen
- Passieren nicht in einem “einzigen” Schritt

Bad Interleaving

- Zum Beispiel:
- Thread 1 reads counter (say, counter = 42).
- Thread 2 also reads counter (still 42).
- Thread 1 writes 43.
- Thread 2 writes 43 (overwriting Thread 1’s update to 43).

- As a result, the counter might end up with a value less than the total expected (in this example, less than 20,000).
- Das nennen wir ein Bad Interleaving!

Bad Interleaving

- As a result, the counter might end up with a value less than the total expected (in this example, less than 20,000).
- Das nennen wir ein Bad Interleaving!

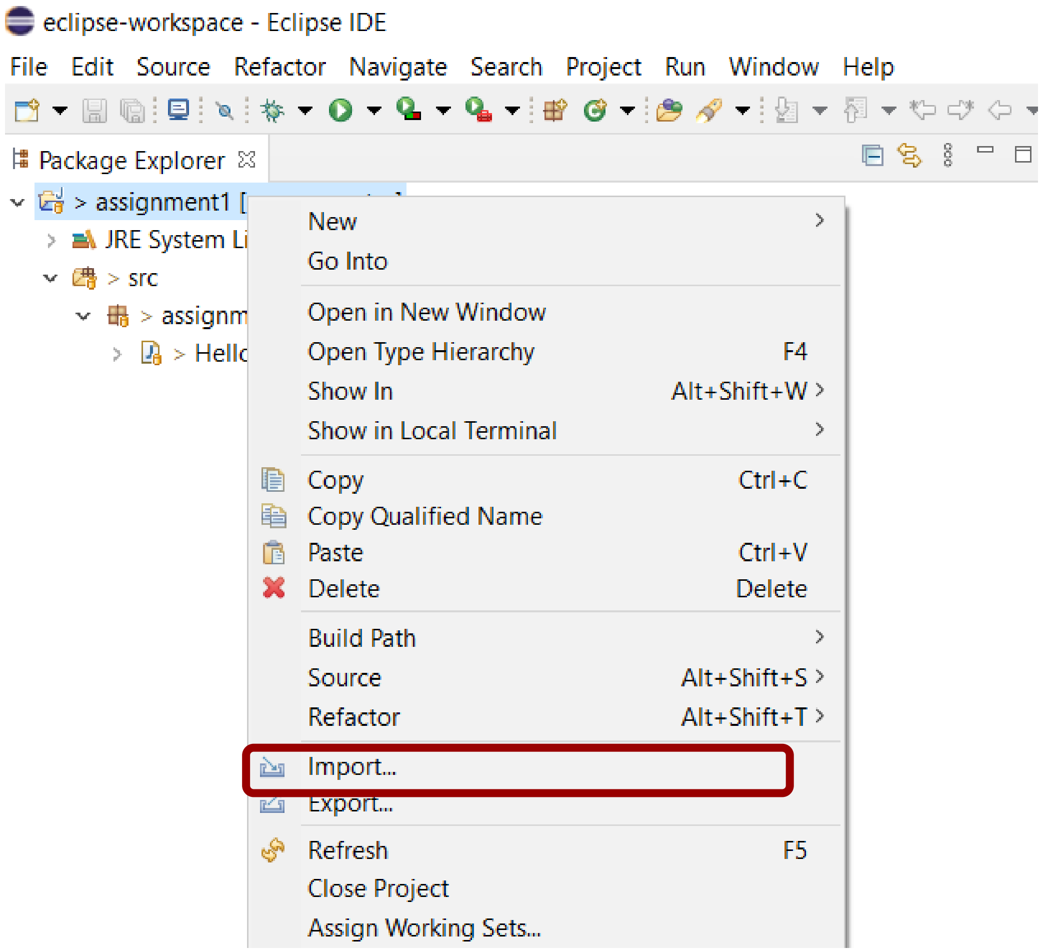
- Es gibt aber verschiedene Wege um das Problem zu fixen
- (synchronized, atomic integer), sehen wir noch

Einstieg in Exercise 2

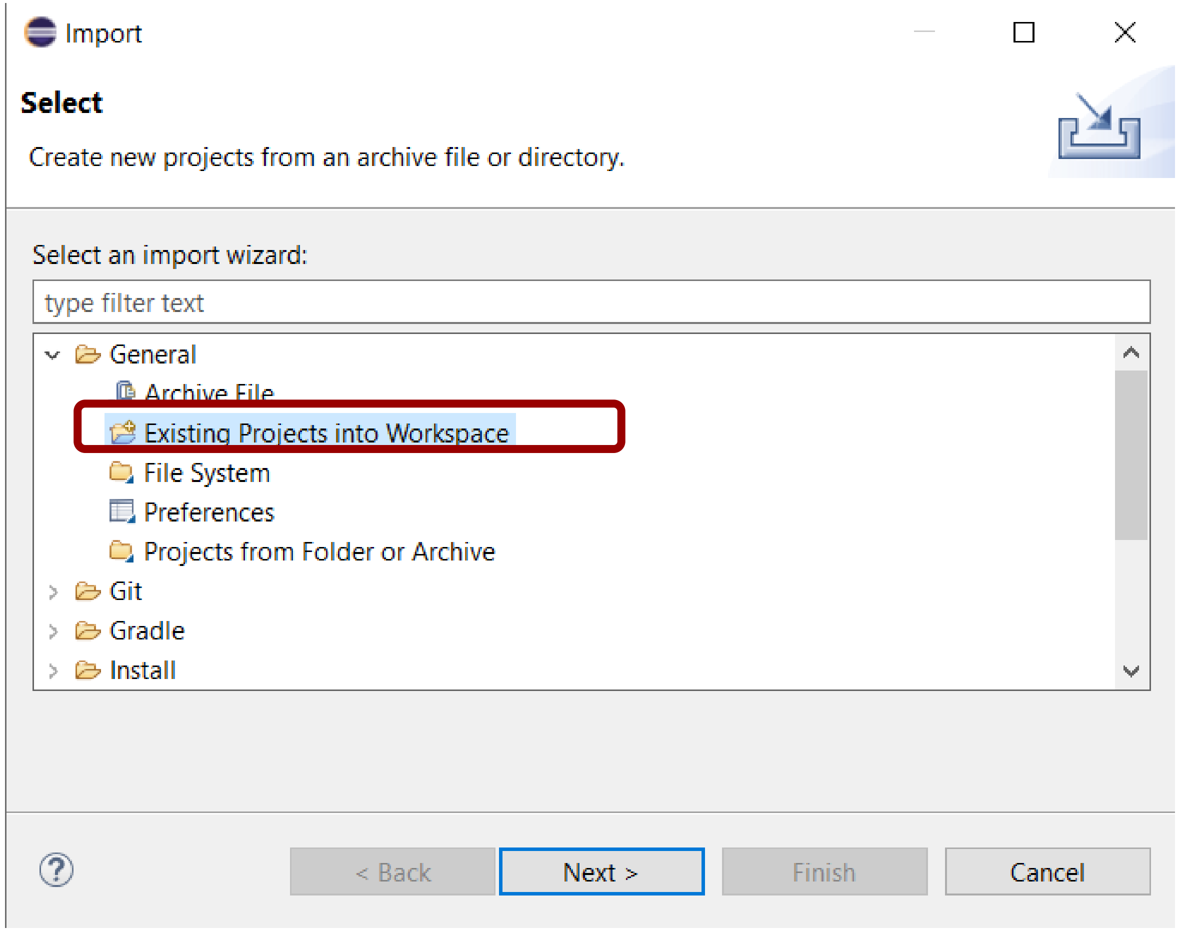
Preparations

1. Import assignment2.zip in Eclipse
2. Run the projects unit-tests in Eclipse
3. Understand output of unit-tests
   - Did the test fail or succeed?
   - Why did the test fail?
4. Start coding and keep checking if tests pass

Eclipse: import project



Eclipse: import project



Import

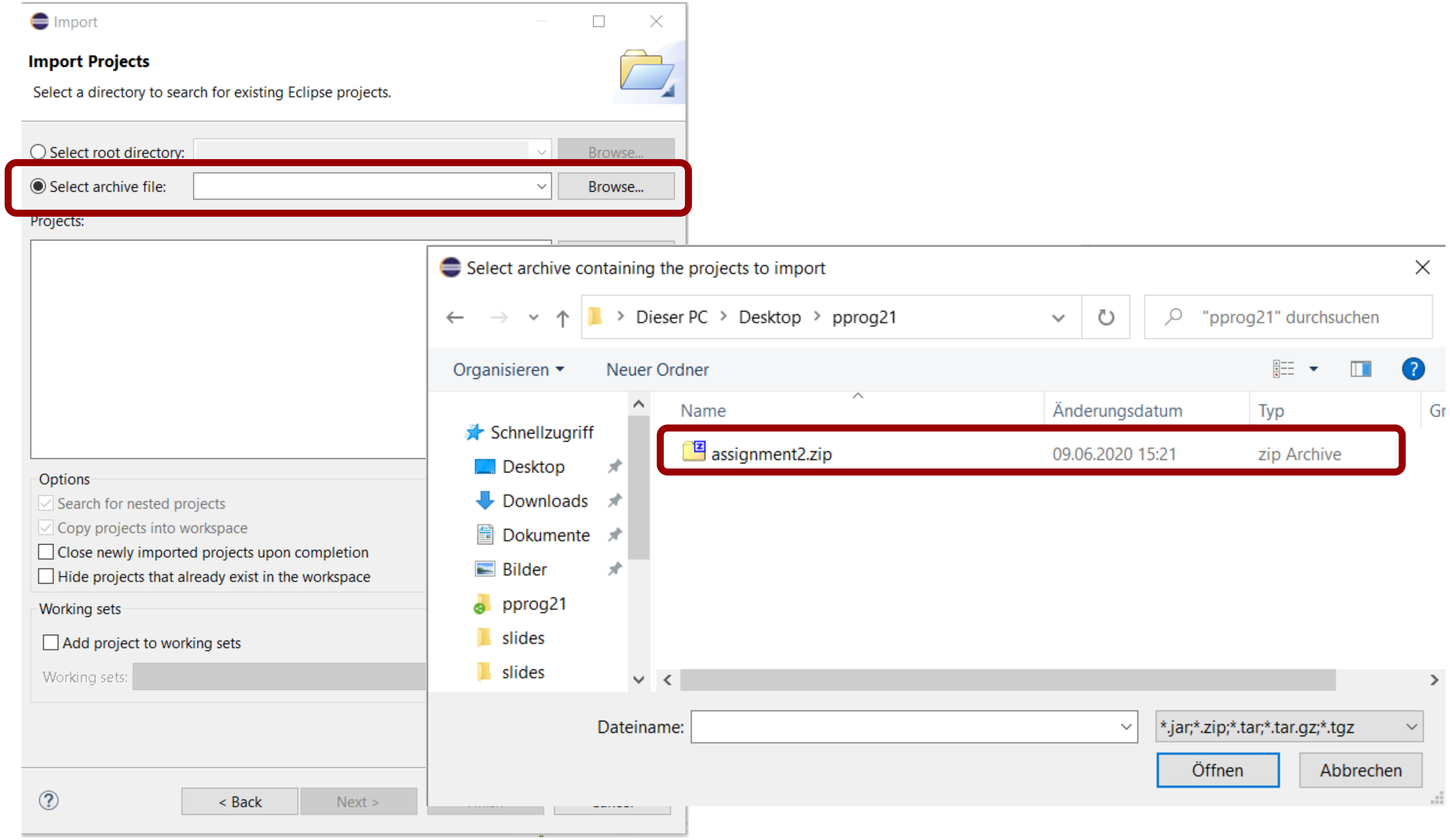
Select

Create new projects from an archive file or directory.

Select an import wizard:

type filter text

Eclipse: import project



○ Select root directory:

● Select archive file:

☑ Search for nested projects

☑ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

☐ Add project to working sets

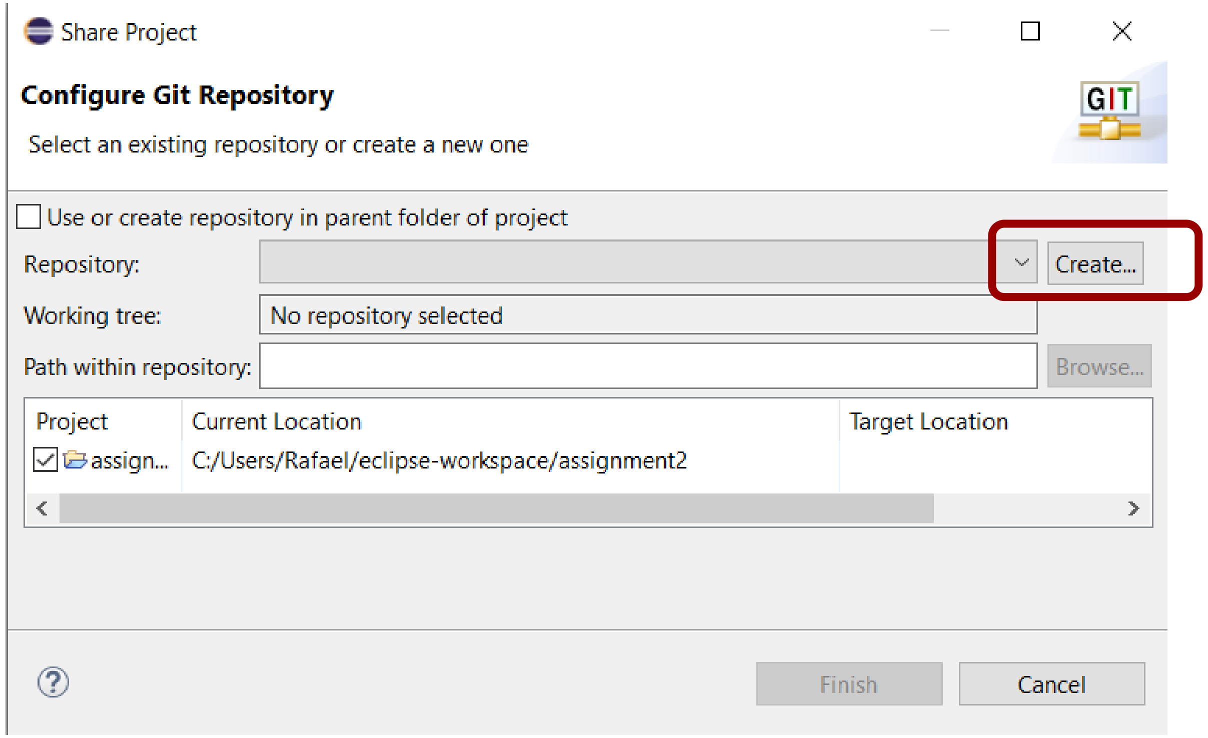
Name	Änderungsdatum	Typ
assignment2.zip	09.06.2020 15:21	zip Archive

Dateiname:

.jar;.zip;*.tar;*.tar.gz;*.tgz

Eclipse: add to git

Team -> Share Project ...



Share Project

Configure Git Repository

Select an existing repository or create a new one

☐ Use or create repository in parent folder of project

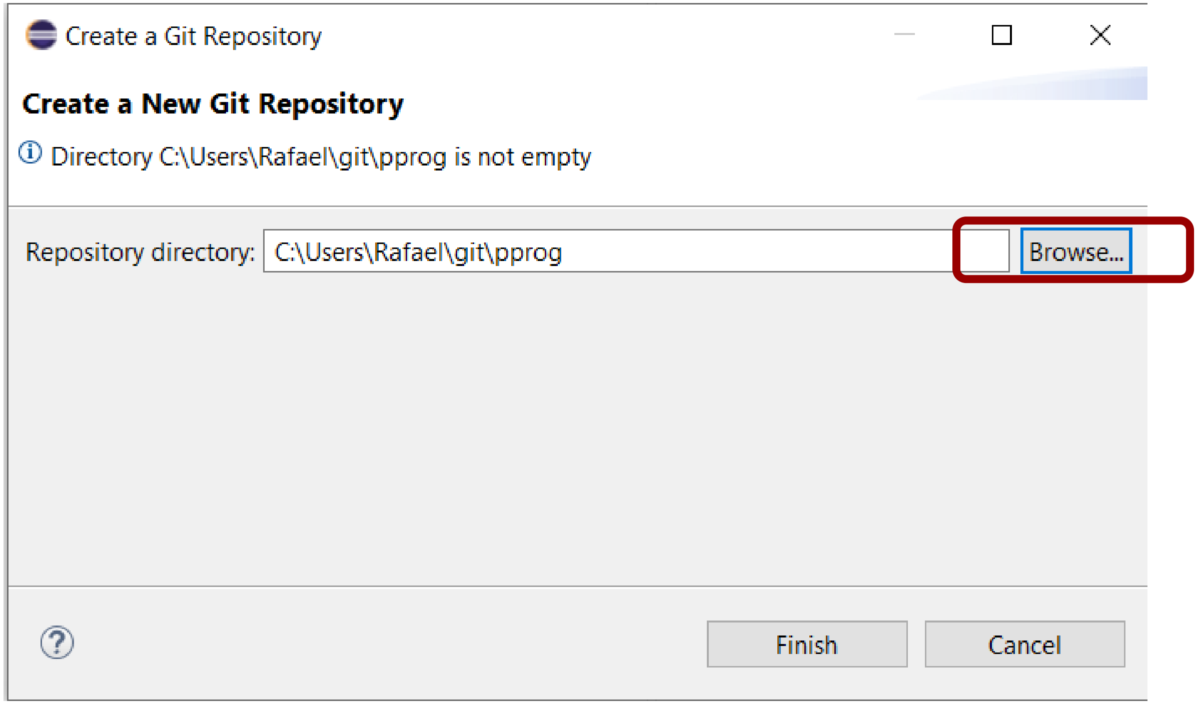
Repository:

Working tree: No repository selected

Path within repository:

Project	Current Location	Target Location
☑ assign...	C:/Users/Rafael/eclipse-workspace/assignment2	

Eclipse: add to git



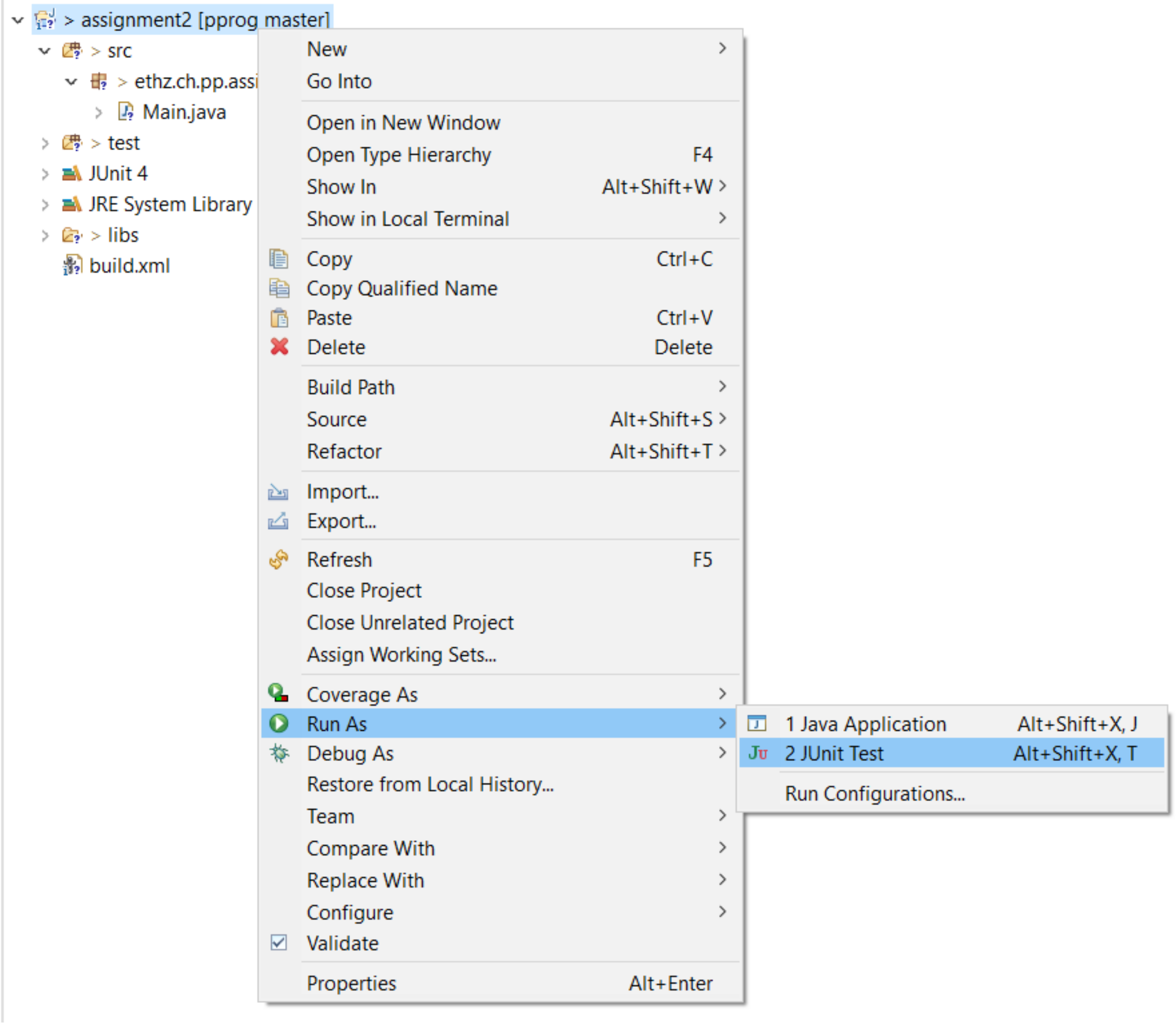
Create a New Git Repository

Directory C:\Users\Rafael\git\pprog is not empty

Repository directory: C:\Users\Rafael\git\pprog

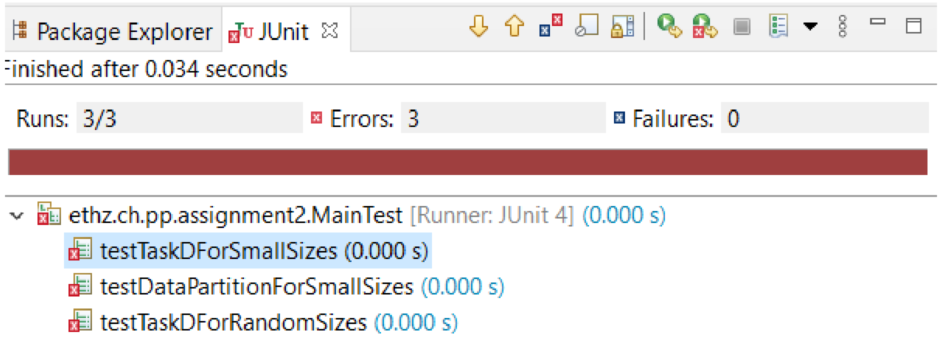
Important: Select same directory as for assignment 1
If you don’t have a repo yet, contact your TA

Eclipse: running JUnit tests (1)



Eclipse: running JUnit tests (2)

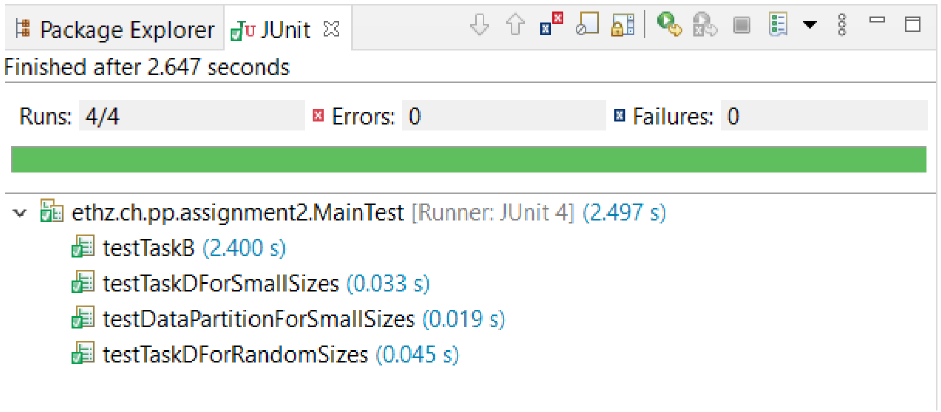
Template



```
Finished after 0.034 seconds
Runs: 3/3    Errors: 3    Failures: 0

ethz.ch.pp.assignment2.MainTest [Runner: JUnit 4] (0.000 s)
    testTaskDForSmallSizes (0.000 s)
    testDataPartitionForSmallSizes (0.000 s)
    testTaskDForRandomSizes (0.000 s)
```

Your solution (ideally)



```
Finished after 2.647 seconds
Runs: 4/4    Errors: 0    Failures: 0

ethz.ch.pp.assignment2.MainTest [Runner: JUnit 4] (2.497 s)
    testTaskB (2.400 s)
    testTaskDForSmallSizes (0.033 s)
    testDataPartitionForSmallSizes (0.019 s)
    testTaskDForRandomSizes (0.045 s)
```

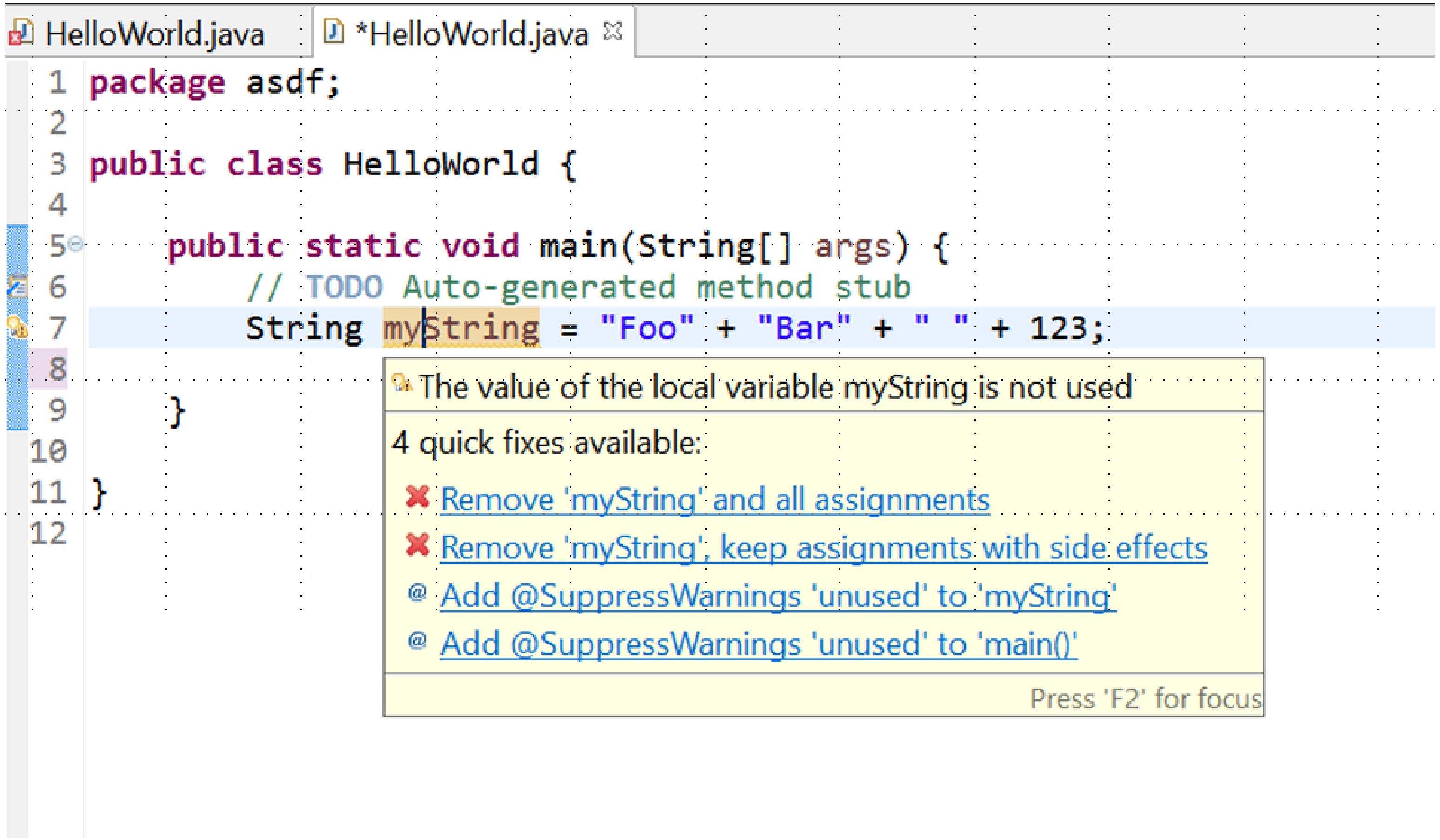
Coding Remarks

Code Style

- Try to make your code as readable as possible
- Include high-level comments that explain why you are doing something (much better than a line-by-line commentary of your code)

Code Style / Errors

Keep attention what Eclipse reports:



```
package asdf;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String myString = "Foo" + "Bar" + " " + 123;

    }

}
```

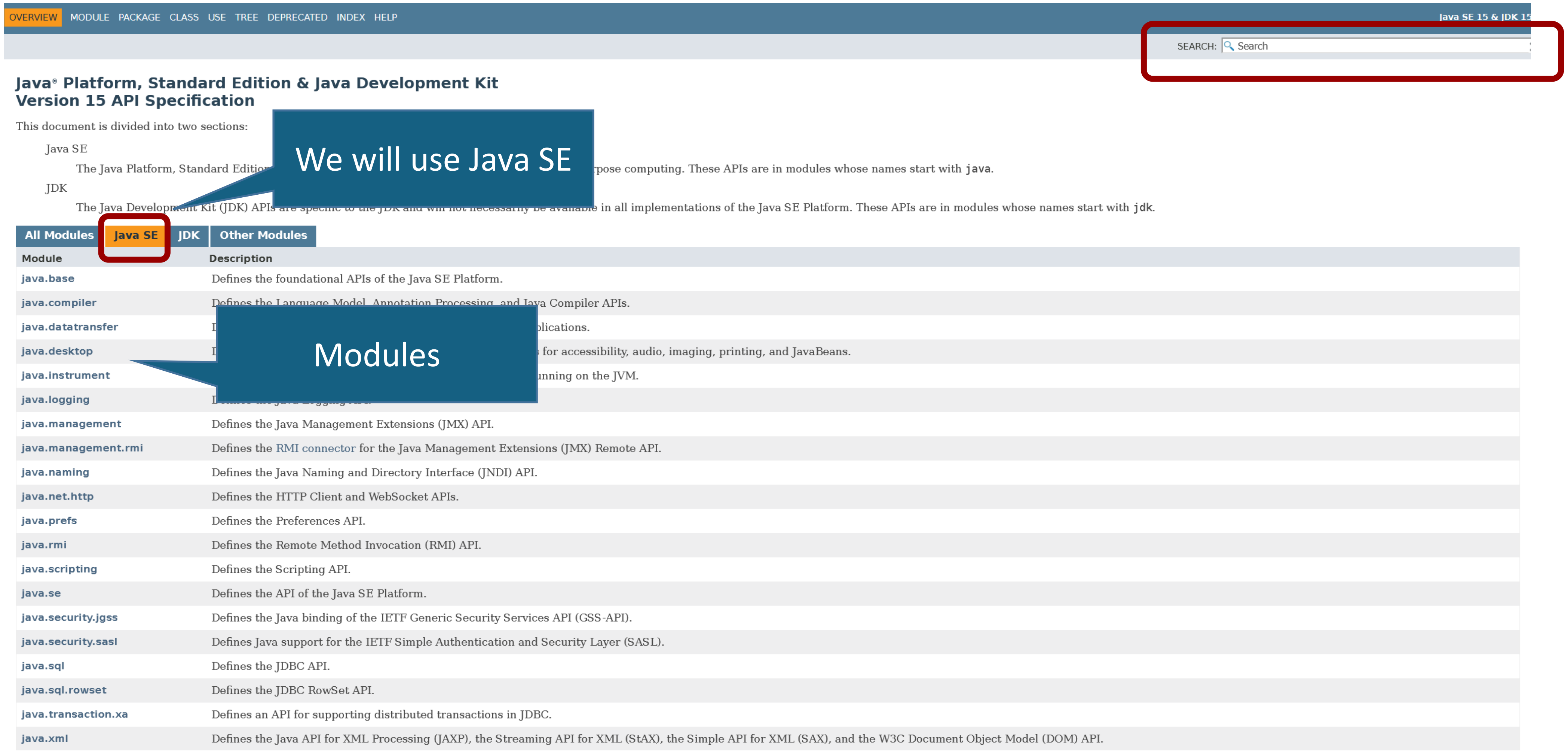
The value of the local variable myString is not used

4 quick fixes available:

- Remove 'myString' and all assignments
- Remove 'myString', keep assignments with side effects
- Add @SuppressWarnings 'unused' to 'myString'
- Add @SuppressWarnings 'unused' to 'main()'

Press 'F2' for focus

Java Doc (https://docs.oracle.com/en/java/javase/21/docs/api/index.html)



Java® Platform, Standard Edition & Java Development Kit Version 15 API Specification

This document is divided into two sections:

Java SE

The Java Platform, Standard Editio … rpose computing. These APIs are in modules whose names start with `java`.

JDK

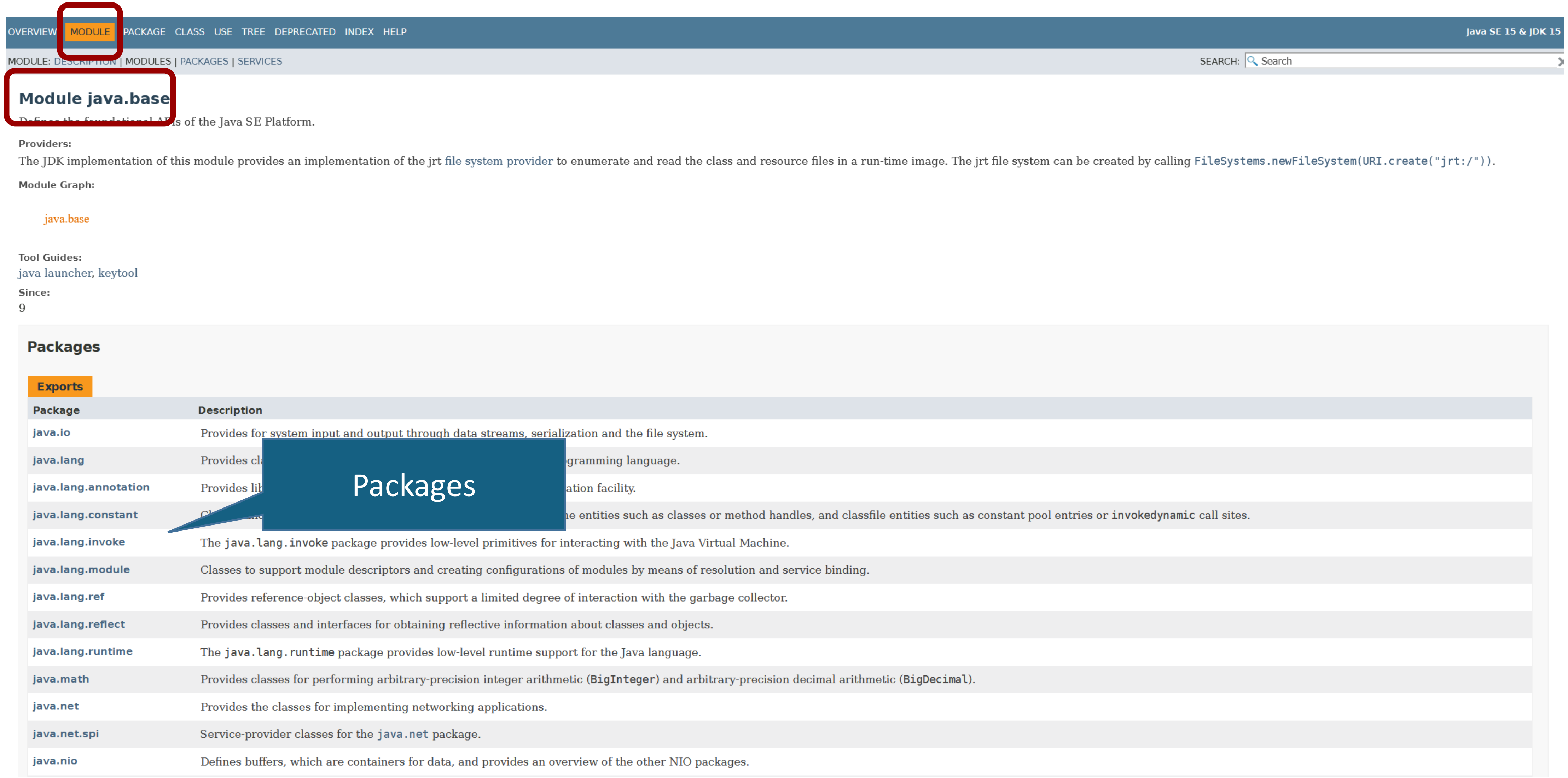
The Java Developm … in all implementations of the Java SE Platform. These APIs are in modules whose names start with `jdk`.

Module	Description
java.base	Defines the foundational APIs of the Java SE Platform.
java.compiler	Defines the Language Model, Annotation Processing, and Java Compiler APIs.
java.datatransfer	D … plications.
java.desktop	D … for accessibility, audio, imaging, printing, and JavaBeans.
java.instrument	… nning on the JVM.
java.logging	D …
java.management	Defines the Java Management Extensions (JMX) API.
java.management.rmi	Defines the RMI connector for the Java Management Extensions (JMX) Remote API.
java.naming	Defines the Java Naming and Directory Interface (JNDI) API.
java.net.http	Defines the HTTP Client and WebSocket APIs.
java.prefs	Defines the Preferences API.
java.rmi	Defines the Remote Method Invocation (RMI) API.
java.scripting	Defines the Scripting API.
java.se	Defines the API of the Java SE Platform.
java.security.jgss	Defines the Java binding of the IETF Generic Security Services API (GSS-API).
java.security.sasl	Defines Java support for the IETF Simple Authentication and Security Layer (SASL).
java.sql	Defines the JDBC API.
java.sql.rowset	Defines the JDBC RowSet API.
java.transaction.xa	Defines an API for supporting distributed transactions in JDBC.
java.xml	Defines the Java API for XML Processing (JAXP), the Streaming API for XML (StAX), the Simple API for XML (SAX), and the W3C Document Object Model (DOM) API.

We will use Java SE

Modules

Java Doc (https://docs.oracle.com/en/java/javase/21/docs/api/index.html)



Module java.base

Defines the foundational APIs of the Java SE Platform.

Providers:

The JDK implementation of this module provides an implementation of the jrt file system provider to enumerate and read the class and resource files in a run-time image. The jrt file system can be created by calling `FileSystems.newFileSystem(URI.create("jrt:/"))`.

Module Graph:

java.base

Tool Guides:

java launcher, keytool

Since:

9

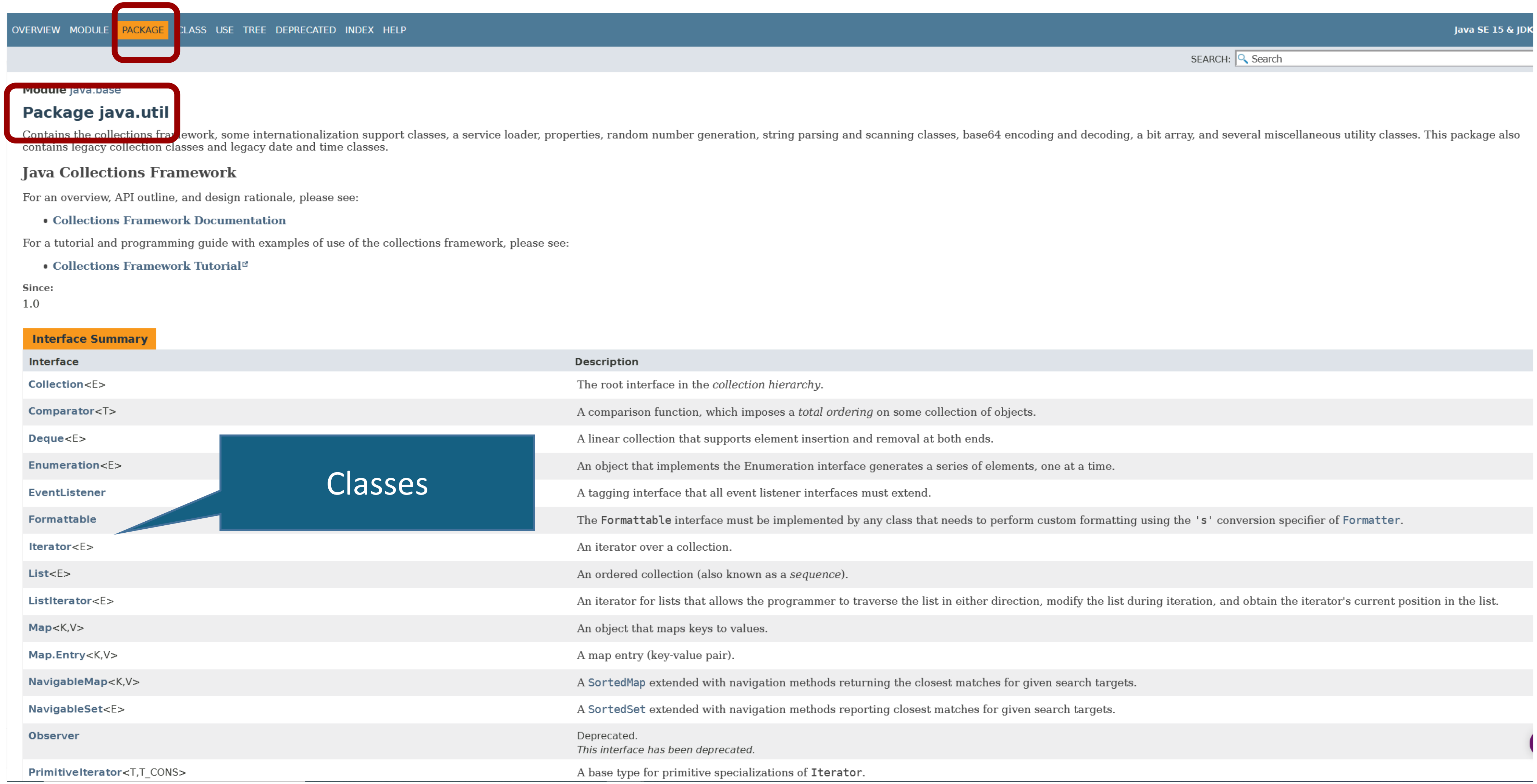
Packages

Exports

Package	Description
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides cl gramming language.
java.lang.annotation	Provides lib ation facility.
java.lang.constant	Cl ne entities such as classes or method handles, and classfile entities such as constant pool entries or `invokedynamic` call sites.
java.lang.invoke	The `java.lang.invoke` package provides low-level primitives for interacting with the Java Virtual Machine.
java.lang.module	Classes to support module descriptors and creating configurations of modules by means of resolution and service binding.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.lang.runtime	The `java.lang.runtime` package provides low-level runtime support for the Java language.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (`BigInteger`) and arbitrary-precision decimal arithmetic (`BigDecimal`).
java.net	Provides the classes for implementing networking applications.
java.net.spi	Service-provider classes for the `java.net` package.
java.nio	Defines buffers, which are containers for data, and provides an overview of the other NIO packages.

Packages

Java Doc (https://docs.oracle.com/en/java/javase/21/docs/api/index.html)


OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
Java SE 15 & JDK
SEARCH: Search

Module java.base

Package java.util

Contains the collections framework, some internationalization support classes, a service loader, properties, random number generation, string parsing and scanning classes, base64 encoding and decoding, a bit array, and several miscellaneous utility classes. This package also contains legacy collection classes and legacy date and time classes.

Java Collections Framework

For an overview, API outline, and design rationale, please see:

- Collections Framework Documentation

For a tutorial and programming guide with examples of use of the collections framework, please see:

- Collections Framework Tutorial

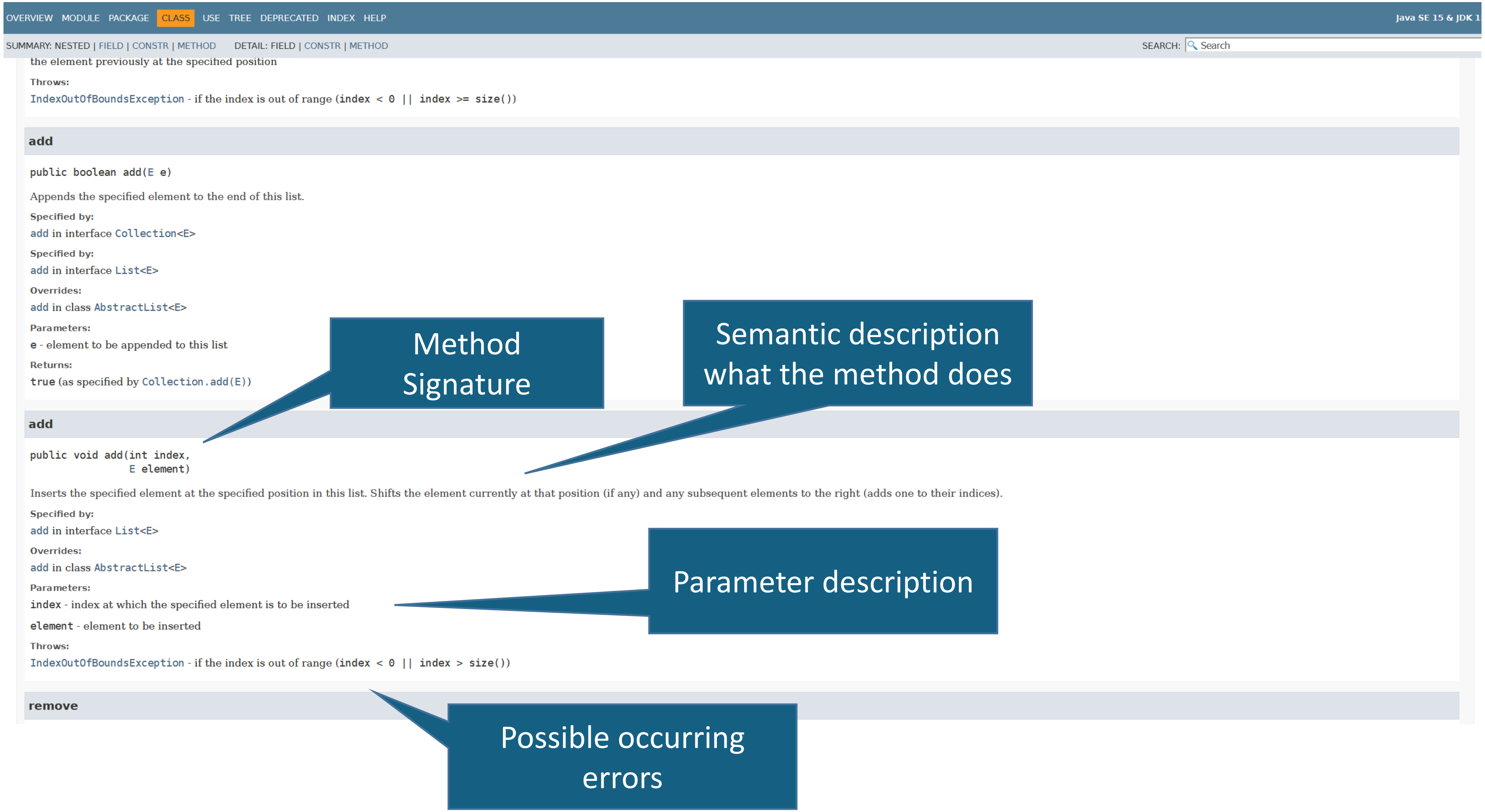
Since:
1.0

Interface Summary

Interface	Description
Collection<E>	The root interface in the *collection hierarchy*.
Comparator<T>	A comparison function, which imposes a *total ordering* on some collection of objects.
Deque<E>	A linear collection that supports element insertion and removal at both ends.
Enumeration<E>	An object that implements the Enumeration interface generates a series of elements, one at a time.
EventListener	A tagging interface that all event listener interfaces must extend.
Formattable	The `Formattable` interface must be implemented by any class that needs to perform custom formatting using the `'s'` conversion specifier of `Formatter`.
Iterator<E>	An iterator over a collection.
List<E>	An ordered collection (also known as a *sequence*).
ListIterator<E>	An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.
Map<K,V>	An object that maps keys to values.
Map.Entry<K,V>	A map entry (key-value pair).
NavigableMap<K,V>	A `SortedMap` extended with navigation methods returning the closest matches for given search targets.
NavigableSet<E>	A `SortedSet` extended with navigation methods reporting closest matches for given search targets.
Observer	Deprecated. *This interface has been deprecated.*
PrimitiveIterator<T,T_CONS>	A base type for primitive specializations of `Iterator`.

Classes

Java Doc (https://docs.oracle.com/en/java/javase/21/docs/api/index.html)



OVERVIEW MODULE PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

the element previously at the specified position

Throws:

`IndexOutOfBoundsException` - if the index is out of range (`index < 0 || index >= size()`)

add

```
public boolean add(E e)
```

Appends the specified element to the end of this list.

Specified by:

`add` in interface `Collection<E>`

Specified by:

`add` in interface `List<E>`

Overrides:

`add` in class `AbstractList<E>`

Parameters:

`e` - element to be appended to this list

Returns:

`true` (as specified by `Collection.add(E)`)

add

```
public void add(int index,
                E element)
```

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Specified by:

`add` in interface `List<E>`

Overrides:

`add` in class `AbstractList<E>`

Parameters:

`index` - index at which the specified element is to be inserted

`element` - element to be inserted

Throws:

`IndexOutOfBoundsException` - if the index is out of range (`index < 0 || index > size()`)

remove

Method Signature

Semantic description what the method does

Parameter description

Possible occurring errors

Exercise Preview

Task A

To start with, print to the console "Hello Thread!" from a new thread. How do you check that the statement was indeed printed from a thread that is different to the main thread of your application? Furthermore, ensure that your program (i.e., the execution of main thread) finishes only after the thread execution finishes.

Task A: How to create and start a new thread?

option 1: Extend class Thread

```
class ConcurrWriter extends Thread { …
     public void run() { … }
}
ConcurrWriter writerThread = new ConcurrWriter();
writerThread.start();   // calls ConcurrWriter.run()
```

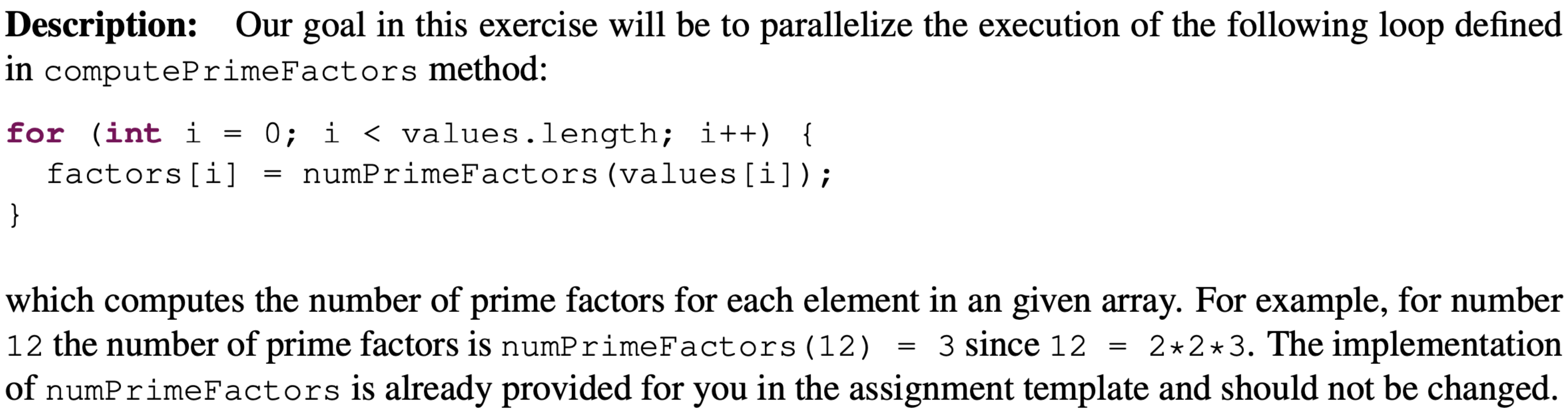
option 2: Implement Runnable

```
public class ConcurrReader implements Runnable {
     …
     public void run() { …
           … code here executes concurrently with caller … }
}

ConcurrReader readerThread = new ConcurrReader();
Thread t = new Thread(readerThread);
t.start();   // calls ConcurrReader.run() automatically
```

Demo see code examples

Task B



Description: Our goal in this exercise will be to parallelize the execution of the following loop defined in `computePrimeFactors` method:

```
for (int i = 0; i < values.length; i++) {
  factors[i] = numPrimeFactors(values[i]);
}
```

which computes the number of prime factors for each element in an given array. For example, for number `12` the number of prime factors is `numPrimeFactors(12) = 3` since `12 = 2*2*3`. The implementation of `numPrimeFactors` is already provided for you in the assignment template and should not be changed.

Task B

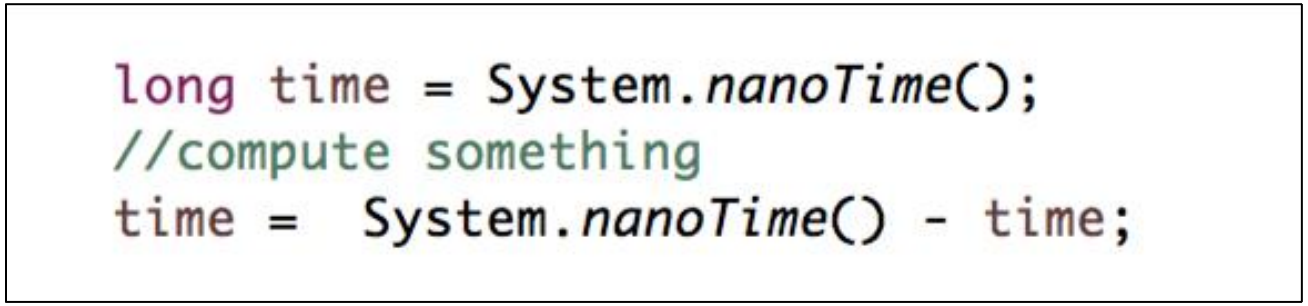
Run the method computePrimeFactors in a single thread other than the main thread. Measure the execution time of sequential execution (on the main thread) and execution using a single thread. Is there any noticeable difference?

Task C

Design and run an experiment that would measure the overhead of creating and executing a thread.

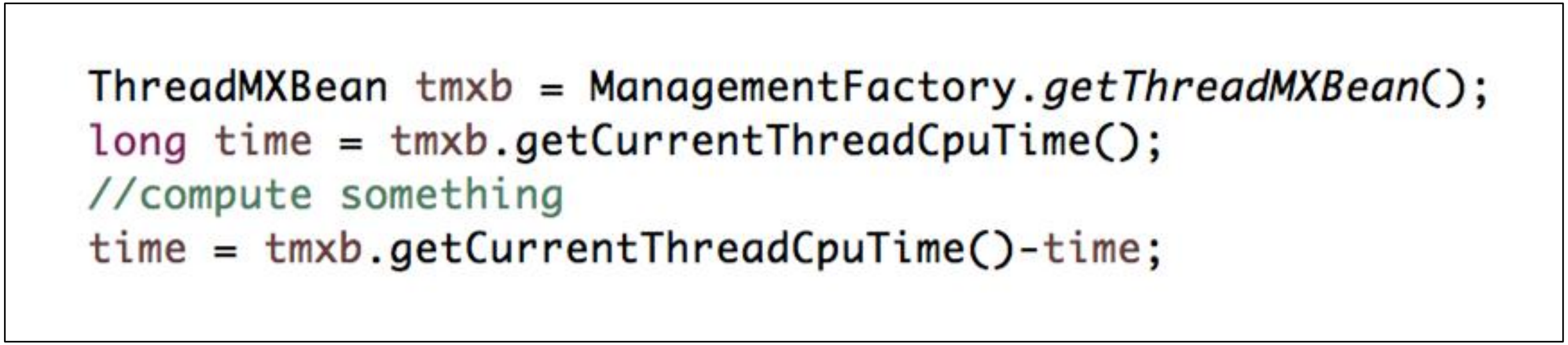
Task C

option 1: Measures real time elapsed including time when the thread is not running.



```
long time = System.nanoTime();
//compute something
time =  System.nanoTime() - time;
```

option 2: Measures thread cpu time excluding time when the thread is not running.



```
ThreadMXBean tmxb = ManagementFactory.getThreadMXBean();
long time = tmxb.getCurrentThreadCpuTime();
//compute something
time = tmxb.getCurrentThreadCpuTime()-time;
```

Task C

- Measured execution time not always the same

→ Average over multiple runs (the more the better)

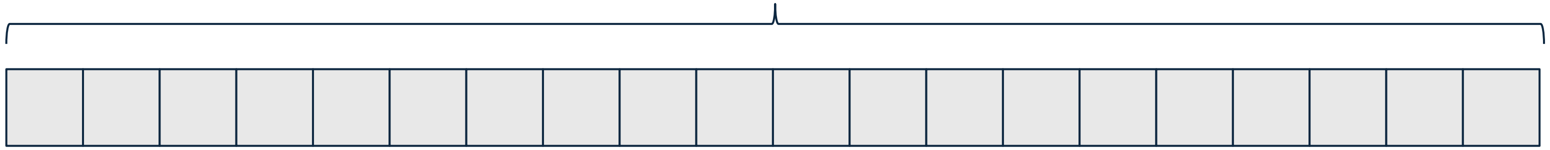
→ Calculate variance

Task D

Before you parallelize the loop in Task E, design how the work should be split between the threads by implementing method PartitionData. Each thread should process roughly equal amount of elements. Briefly describe you solution and discuss alternative ways to split the work.

Task D: Split the work between the threads

```
PartitionData(int length, int numPartitions) { … }
```



length (20)

Input

a) PartitionData(20,1) ?

b) PartitionData(20,2) ?

c) PartitionData(20,3) ?

Task D: Split the work between the threads

```
PartitionData(int length, int numPartitions) {  …  }
```

length (20)

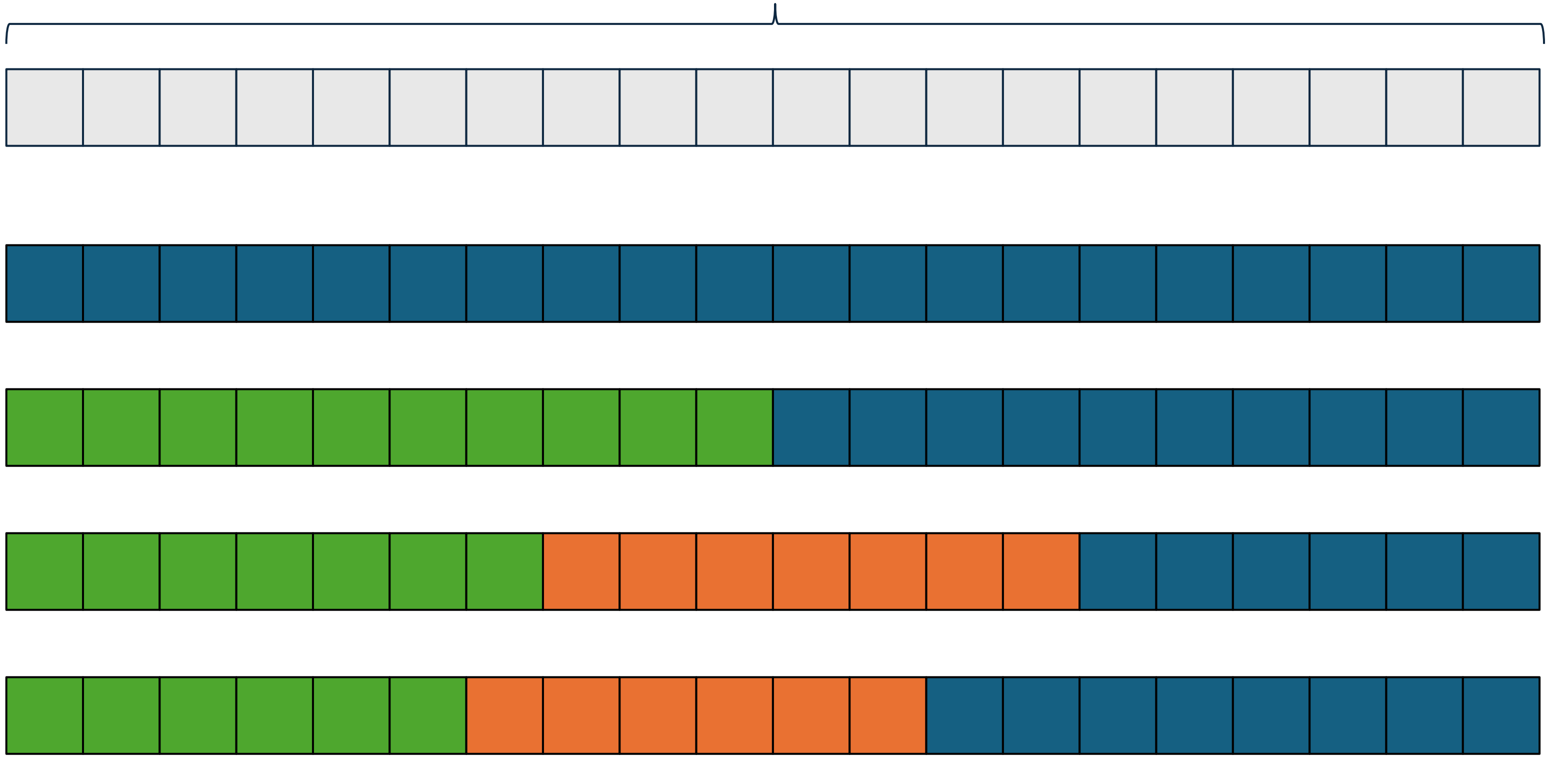
Input

a) PartitionData(20,1)

b) PartitionData(20,2)

c) PartitionData(20,3)

d) PartitionData(20,3)



both c) and d) are correct solutions for this exercise

Task D

Several ways with different performance depending on task and data

If input is random: Splitting the input into half works well

If input is sorted: 1. half finishes faster than 2. half
→ maybe split on odd/even indices

Task D

- What about (length>0 and numPartitions>0) and length<numPartitions?
  - ??
  - ??
- And (length<=0 or numPartitions<=0)?
  - ??
  - ??

```
PartitionData(int length, int numPartitions) {  …  }
```

Task D

- What about (length>0 and numPartitions>0) and length<numPartitions?
    - Throw an exception?
    - Return m = min(m,n) splits?
- And (length<=0 or numPartitions<=0)?
    - Throw an exception?
    - Create a default return value (e.g. new ArraySplit[0])?
- In any case, write your assumptions in JavaDoc

```
PartitionData(int length, int numPartitions) { … }
```

Task E

Parallelize the loop execution in computePrimeFactors using a configurable number of threads.

Task F

Think of how would a plot that shows the execution speed-up of your implementation, for n = 1, 2, 4, 8, 16, 32, 64, 128 threads and the input array size of 100, 1000, 10000, 100000 look like.

Task G

Measure the execution time of your parallel implementation for n = 1, 2, 4, 8, 16, 32, 64, 128 threads and the input array size of input.length = 100, 1000, 10000, 100000. Discuss the differences in the two plots from task F and G.

Speedup

Sub-linear: usually, im besten Fall linear

Super-linear: not possible in theory, *but*

- Modern hardware properties (local/remote memory)
- Bug (this course assumes this)
- Wird als Anomalie betrachtet (zum Beispiel plötzlich bessere cache utilization)

Speedup

Sub-linear: usually

Super-linear: not possible in theory

Wieso?

- Amdahls Law -> Sequentieller Anteil eines Programmes schränkt den Speedup ein, egal wie viele Cores wir haben
- Thread creation, scheduling, and synchronization add extra work that doesn’t exist in a sequential run.
- Context switching and coordination between threads also slow down execution.
- When multiple threads access shared resources (e.g., memory, caches, I/O), contention and delays occur.

Speedup

Sub-linear: usually

Super-linear: not possible in theory

Superlinear speedup (where the speedup is greater than the number of processors) would mean that parallel execution is more than just dividing the work—it would imply that each additional processor gives you an extra benefit beyond the direct division of labor.

Speedup

Sub-linear: usually

Super-linear:

not possible in

theory

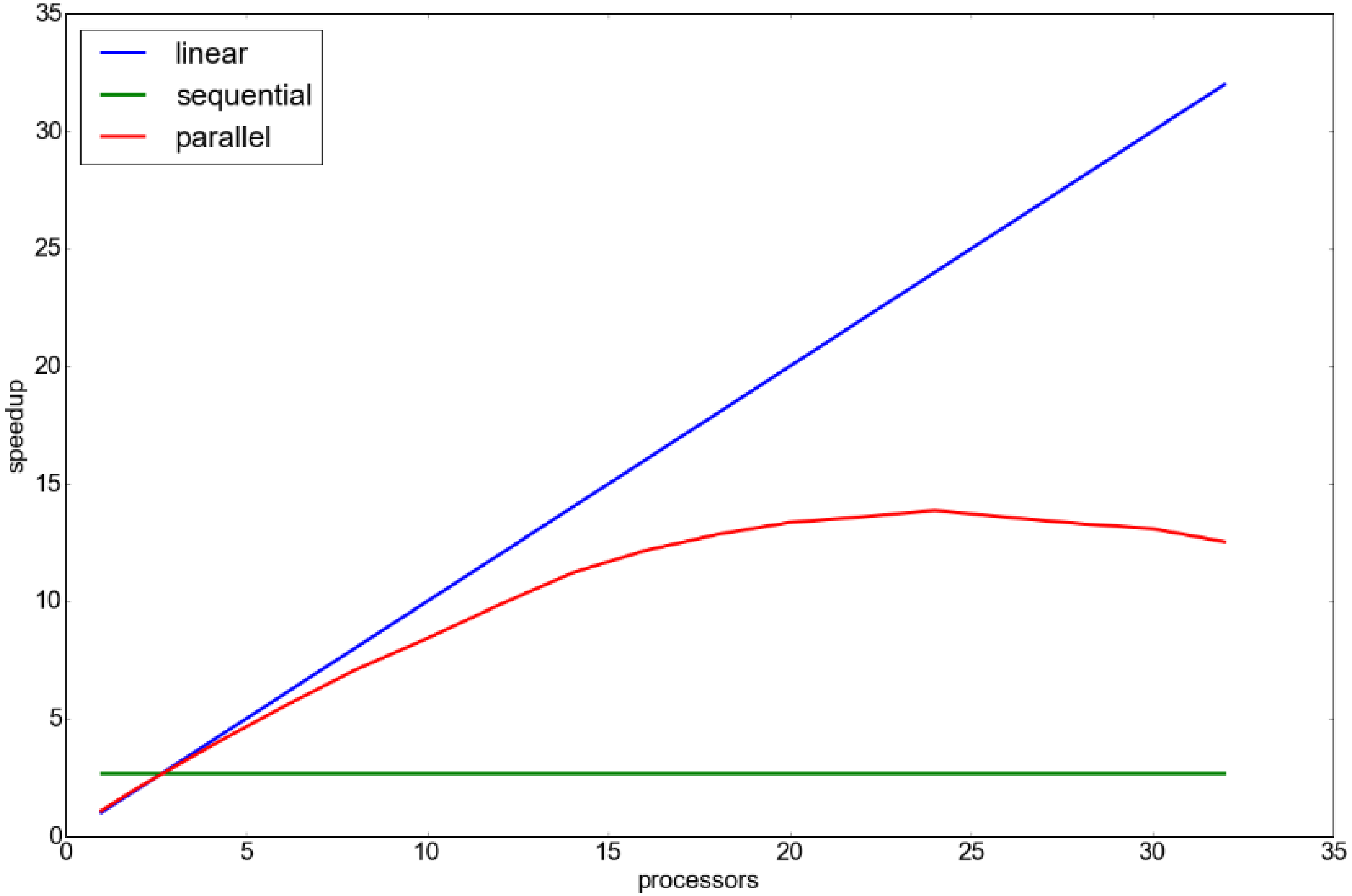
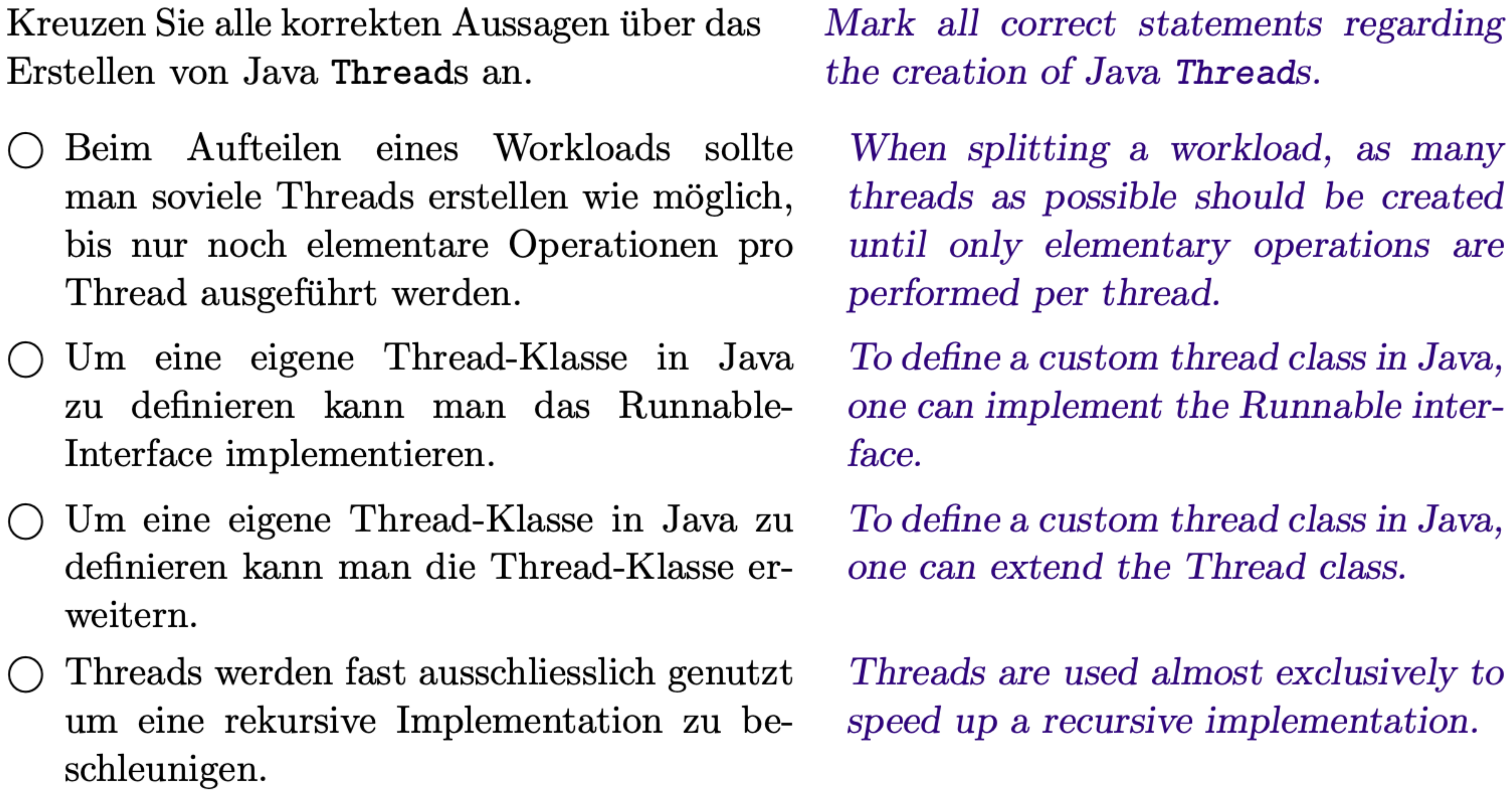


Figure 4.1: A typical graph comparing actual to linear speedup

Past Exam Task



Kreuzen Sie alle korrekten Aussagen über das Erstellen von Java `Threads` an.

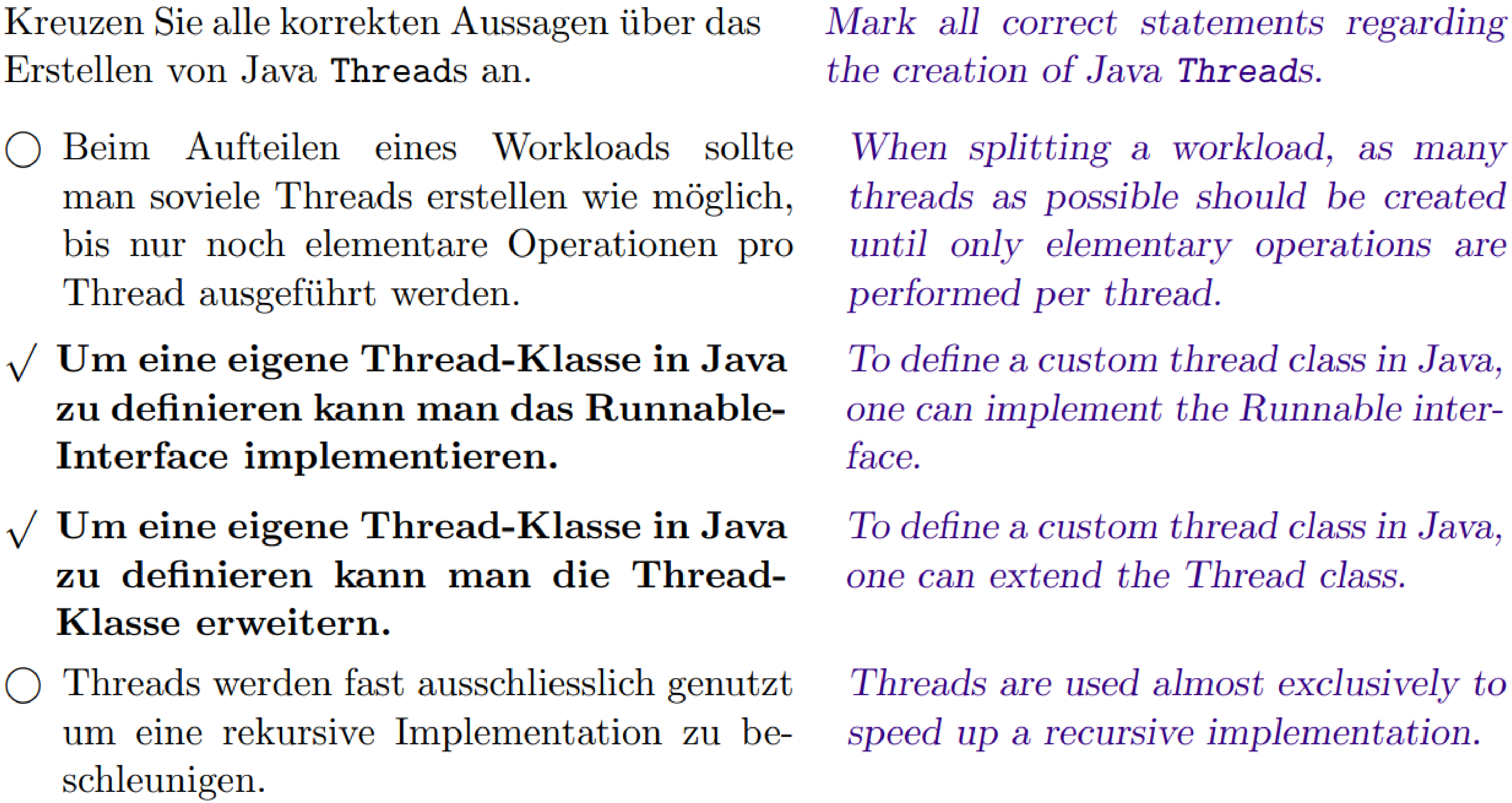
- ○ Beim Aufteilen eines Workloads sollte man soviele Threads erstellen wie möglich, bis nur noch elementare Operationen pro Thread ausgeführt werden.
- ○ Um eine eigene Thread-Klasse in Java zu definieren kann man das Runnable-Interface implementieren.
- ○ Um eine eigene Thread-Klasse in Java zu definieren kann man die Thread-Klasse erweitern.
- ○ Threads werden fast ausschliesslich genutzt um eine rekursive Implementation zu beschleunigen.

Mark all correct statements regarding the creation of Java `Threads`.

- *When splitting a workload, as many threads as possible should be created until only elementary operations are performed per thread.*
- *To define a custom thread class in Java, one can implement the Runnable interface.*
- *To define a custom thread class in Java, one can extend the Thread class.*
- *Threads are used almost exclusively to speed up a recursive implementation.*

Rep. Exam, FS 2023

Past Exam Task



Kreuzen Sie alle korrekten Aussagen über das Erstellen von Java `Threads` an.

Mark all correct statements regarding the creation of Java `Threads`*.*

○ Beim Aufteilen eines Workloads sollte man soviele Threads erstellen wie möglich, bis nur noch elementare Operationen pro Thread ausgeführt werden.

When splitting a workload, as many threads as possible should be created until only elementary operations are performed per thread.

√ **Um eine eigene Thread-Klasse in Java zu definieren kann man das Runnable-Interface implementieren.**

To define a custom thread class in Java, one can implement the Runnable interface.

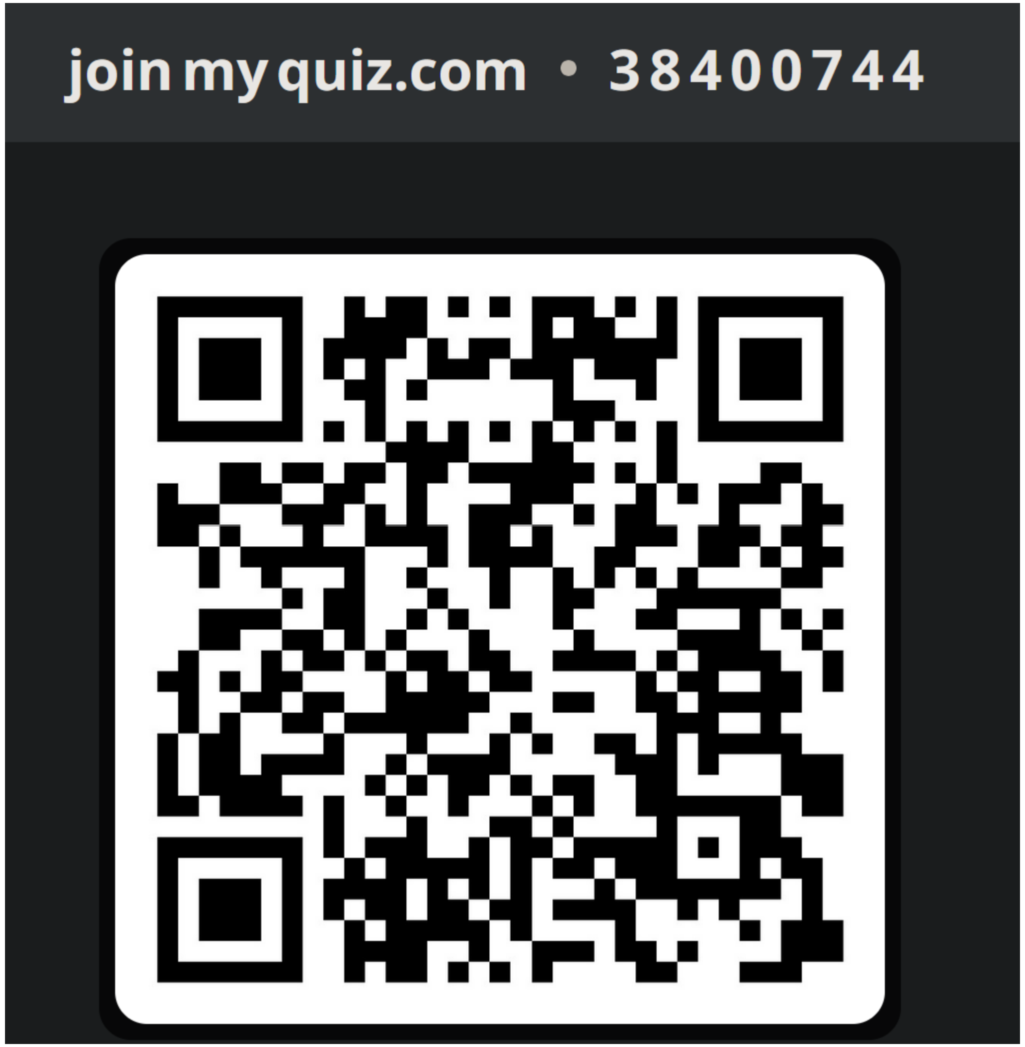
√ **Um eine eigene Thread-Klasse in Java zu definieren kann man die Thread-Klasse erweitern.**

To define a custom thread class in Java, one can extend the Thread class.

○ Threads werden fast ausschliesslich genutzt um eine rekursive Implementation zu beschleunigen.

Threads are used almost exclusively to speed up a recursive implementation.

Rep. Exam, FS 2023



joinmyquiz.com • 38400744

https://quizizz.com/join?gc=38400744

Feedback

- Falls ihr Feedback möchtet sagt mir bitte Bescheid

Danke

- Bis nächste Woche!