

Parallel Programming

Exercise Session 2

Spring 2026

anguhl@ethz.ch

annikaguhl.com

Schedule

Preparation assignment 2 5'

Coding Remarks 5'

Theory Recap 15'

Pre-Discussion assignment 2 20'

Break 15'

Quiz 15'

Empfehlung: Lernt den Umgang mit der Command Line

Nicht relevant für PP ausserhalb vom Erhalten/Abgeben von Übungen, aber es wird euch im weiteren Studium helfen (SPCA, etc.)

Einen guten Überblick findet ihr hier:

<https://www.codecademy.com/article/command-line-commands>

Um den Umgang mit der Command Line zu üben, empfehle ich OverTheWire Bandit:

<https://overthewire.org/wargames/bandit/>.

Wie wird man TA?

Grundsätzlich in jedem Fach unterschiedlich

Meistens wird eine gute Note (5.75+) benötigt

- Nach Fach mehr oder weniger wichtig, in manchen Fächern ist gutes Unterrichten wichtiger

Manche Fächer schicken Anfragen per Mail an Leute mit guten Noten

Meine Empfehlung, falls ihr TA werden wollt: Schickt 1-2 Monate im neuen Semester ein Mail an die Head-TAs (und die Dozenten im CC), um zu fragen, wie der Bewerbungsprozess funktioniert

Preparation Exercise 2

Preparations

1. Fetch Assignment 2 from Gitlab upstream
2. Run the projects unit-tests in IntelliJ
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

Pulling Assignment 02

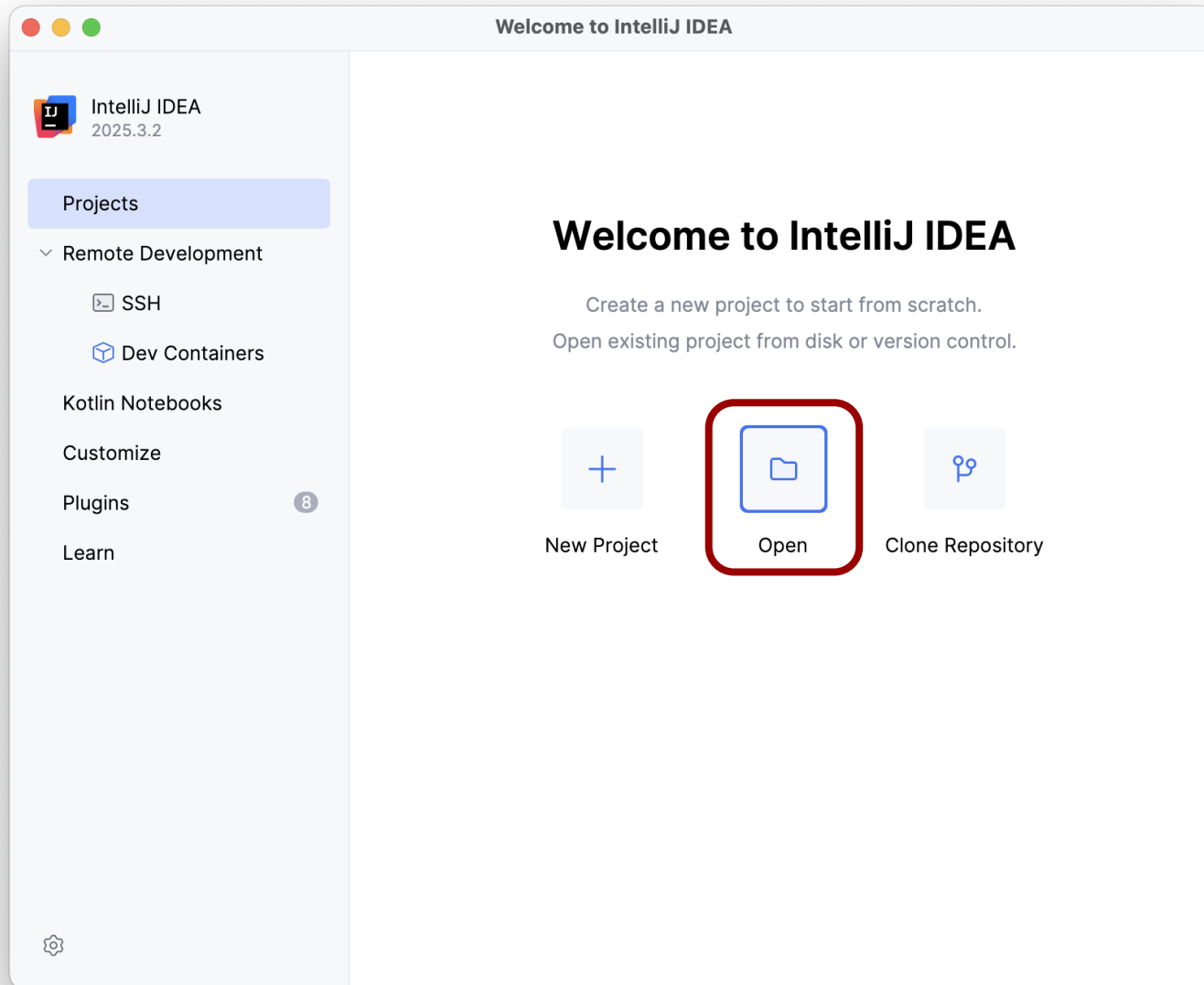
Each week, when new assignments are released, you will have to pull from an upstream repository.

To do this, please use the following commands:

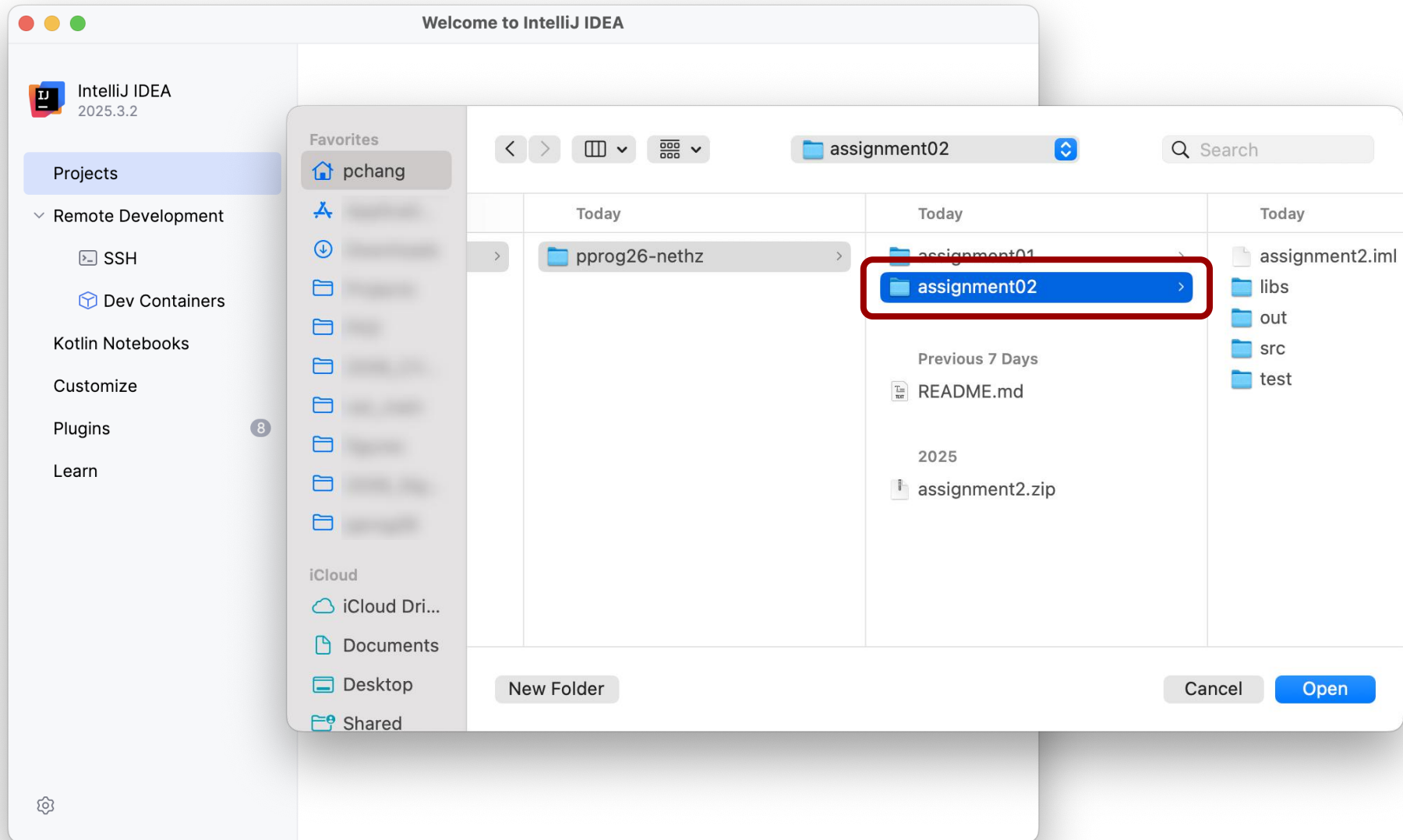
```
# check to see if there is a new assignment  
git fetch upstream
```

```
# merge your assignment into your repository  
git merge upstream/main
```

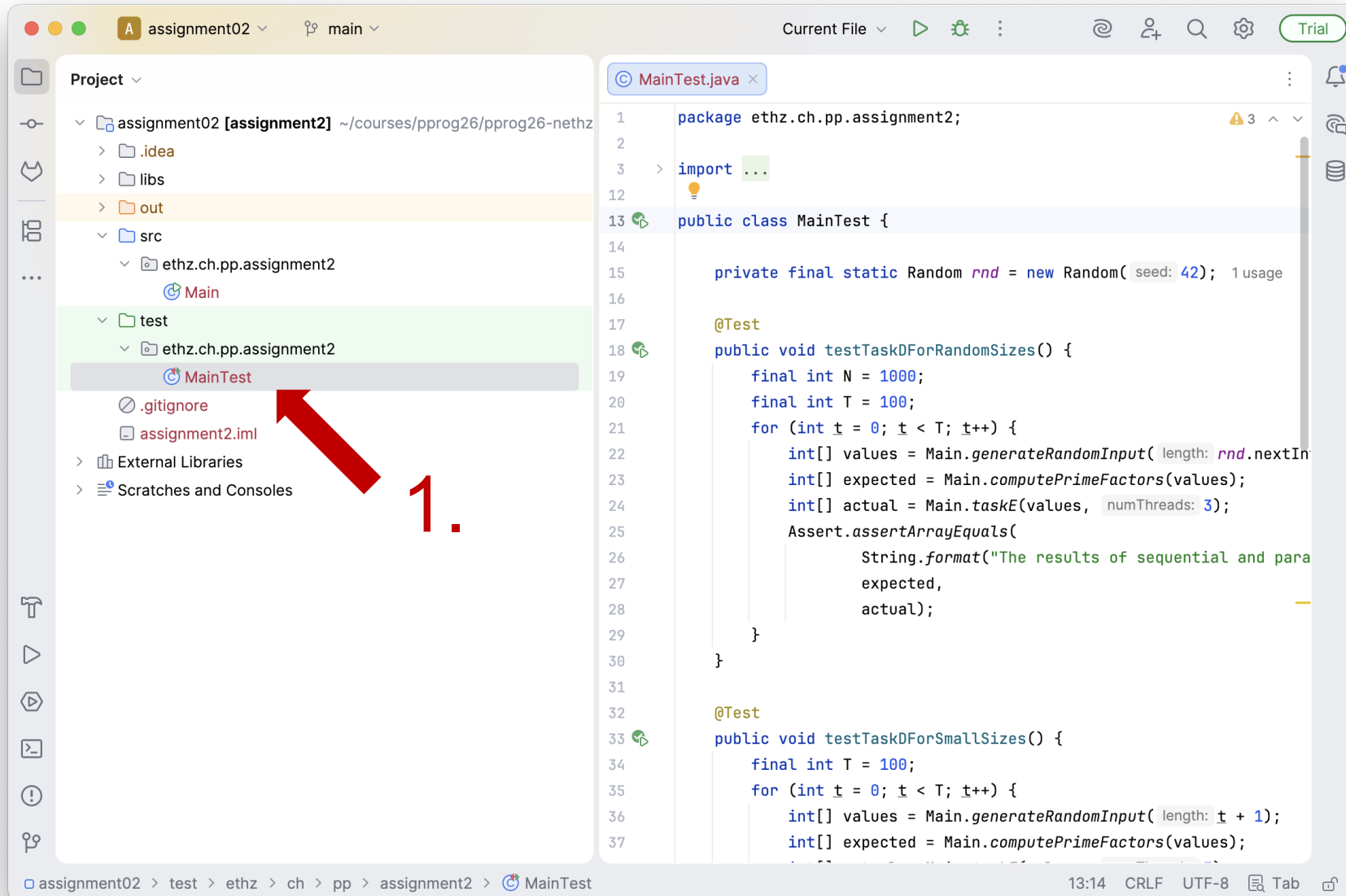
IntelliJ IDEA: Import Project



IntelliJ IDEA: Import Project



IntelliJ: running JUnit tests (1)



IntelliJ: running JUnit tests (1)

The screenshot shows the IntelliJ IDEA interface. On the left, the Project tool window displays the project structure. A red arrow labeled '1.' points to the 'MainTest' file in the 'test' directory. The main editor shows the code for 'MainTest.java'. The code includes package declarations, imports, and two test methods: 'testTaskDForRandomSizes()' and 'testTaskDForSmallSizes()'. The 'Run' button (a green play icon) in the top toolbar is highlighted with a red box and labeled '2.'.

```
package ethz.ch.pp.assignment2;

import java.util.Random;

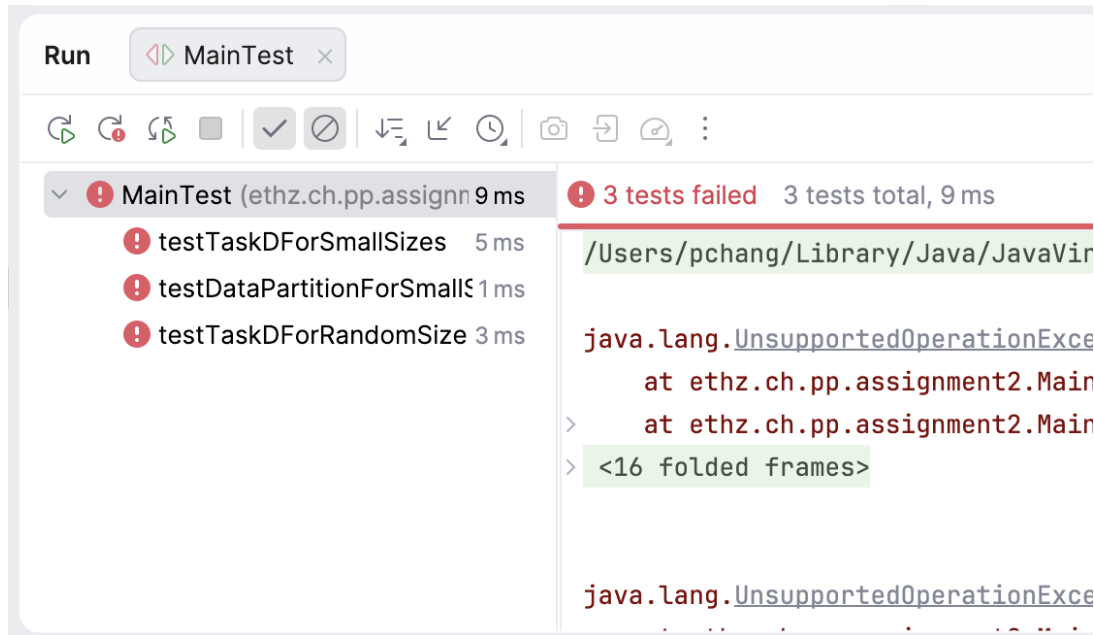
public class MainTest {

    private final static Random rnd = new Random();

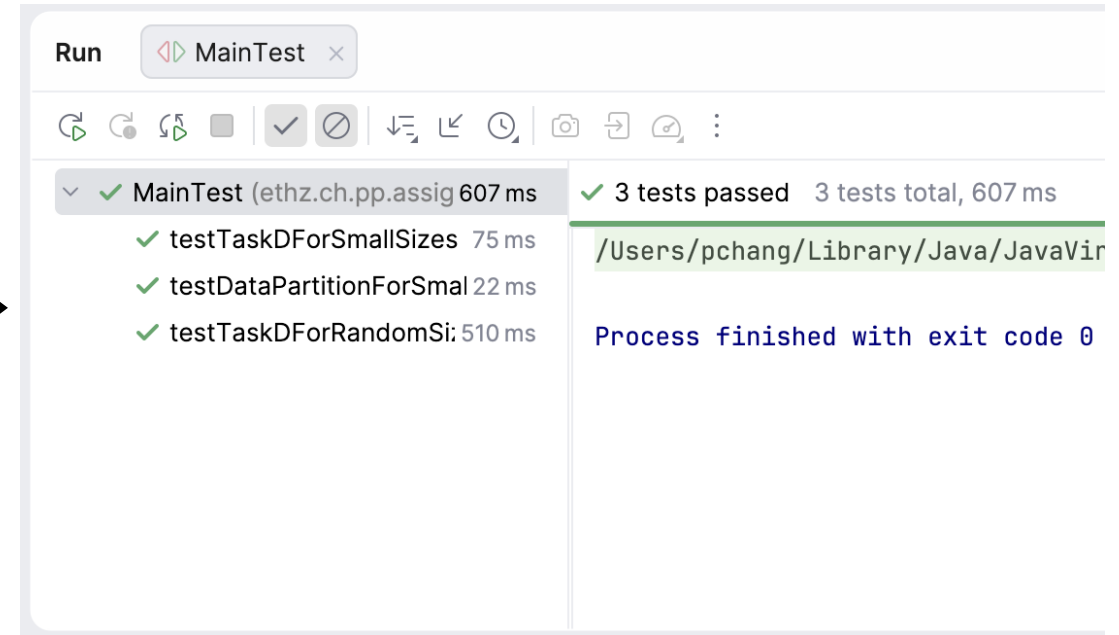
    @Test
    public void testTaskDForRandomSizes() {
        final int N = 1000;
        final int T = 100;
        for (int t = 0; t < T; t++) {
            int[] values = Main.generateRandomInput( length: rnd.nextInt(N));
            int[] expected = Main.computePrimeFactors(values);
            int[] actual = Main.taskE(values, numThreads: 3);
            Assert.assertEquals(
                String.format("The results of sequential and parallel computation are not equal for N=%d, t=%d", N, t),
                expected,
                actual);
        }
    }

    @Test
    public void testTaskDForSmallSizes() {
        final int T = 100;
        for (int t = 0; t < T; t++) {
            int[] values = Main.generateRandomInput( length: t + 1);
            int[] expected = Main.computePrimeFactors(values);
            int[] actual = Main.taskE(values, numThreads: 3);
            Assert.assertEquals(
                String.format("The results of sequential and parallel computation are not equal for N=%d, t=%d", t + 1, t),
                expected,
                actual);
        }
    }
}
```

IntelliJ: running JUnit tests (2)



Template



Your solution (ideally)

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 IntelliJ reports:



```
1 package asdf;
2
3 public class HelloWorld {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         String myString = "Foo" + "Bar" + " " + 123;
8     }
9 }
10
```

Variable 'myString' is never used

Remove local variable 'myString'   More actions...  

```
String myString = "Foo" + "Bar" + " " + 123
```

 assignment2  

Java Doc (<https://docs.oracle.com/en/java/javase/25/docs/api/index.html>)

The screenshot shows the Oracle Java SE 25 & JDK 25 API Specification page. At the top, there is a navigation bar with links: OVERVIEW, TREE, PREVIEW, NEW, DEPRECATED, INDEX, SEARCH, HELP. The version 'Java SE 25 & JDK 25' is displayed in the top right. A search bar is highlighted with a red box and contains the text 'Search documentation (type /)'. The main heading is 'Java® Platform, Standard Edition & Java Development Kit Version 25 API Specification'. Below this, it states 'This document is divided into two sections:'. Under 'Java SE', it says 'The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with java.' Under 'JDK', it says 'The Java Development Kit (JDK) APIs are not necessarily available in all implementations of the Java SE Platform. These APIs are in modules whose names start with java.'. A blue callout box with the text 'We will use Java SE' points to the 'Java SE' tab in the 'All Modules' section. Below this is a table of modules. A blue callout box with the text 'Modules' points to the table. The table has two columns: 'Module' and 'Description'. The 'All Modules' tab is selected and highlighted with a red box.

OVERVIEW TREE PREVIEW NEW DEPRECATED INDEX SEARCH HELP Java SE 25 & JDK 25

Search documentation (type /)

Java® Platform, Standard Edition & Java Development Kit Version 25 API Specification

This document is divided into two sections:

Java SE

The Java Platform, Standard Edition (Java SE) APIs define the core Java platform for general-purpose computing. These APIs are in modules whose names start with java.

JDK

The Java Development Kit (JDK) APIs are not necessarily available in all implementations of the Java SE Platform. These APIs are in modules whose names start with java.

All Modules **Java SE** **JDK** **Other Modules**

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	Defines APIs for transferring data between and within applications.
java.desktop	Defines APIs for user interface toolkits, plus APIs for accessibility, audio, imaging, printing, and JavaBeans.
java.instrument	Defines services that allow agents to instrument programs running on the JVM.
java.logging	Defines the Java Logging API.
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 Doc (<https://docs.oracle.com/en/java/javase/25/docs/api/index.html>)

OVERVIEW **MODULE** TREE PREVIEW NEW DEPRECATED INDEX SEARCH HELP Java SE 25 & JDK 25

java.base

Module java.base

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.getFileSystem(URI.create("jrt:/"))`.

Module Graph:

java.base

Tool Guides:

java launcher, keytool

Since:

9

Packages

Exports

Package	Description
java.io	Provides for custom input and output through data streams, serialization and the file system.
java.lang	Essential to the design of the Java programming language.
java.lang.annotation	Java programming language annotation facility.
java.lang.classfile	Provides classfile parsing, generation, and transformation library.
java.lang.classfile.attribute	Provides interfaces describing class file attributes for the java.lang.classfile library.
java.lang.classfile.constantpool	Provides interfaces describing constant pool entries for the java.lang.classfile library.

Java Doc (<https://docs.oracle.com/en/java/javase/25/docs/api/index.html>)

OVERVIEW **PACKAGE** USE TREE PREVIEW NEW DEPRECATED INDEX SEARCH HELP Java SE 25 & JDK 25

java.base > **java.util** Search documentation (type /)

Package java.util

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

All Classes and Interfaces **Interfaces** **Classes** **Enum Classes** **Exception Classes**

Class	Description
AbstractCollection<E>	This class provides a skeletal implementation of the <code>Collection</code> interface, to minimize the effort required to implement this interface.
AbstractList<E>	This class provides a skeletal implementation of the <code>List</code> interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).
AbstractMap<K,V>	This class provides a skeletal implementation of the <code>Map</code> interface, to minimize the effort required to implement this interface.
AbstractMap.SimpleEntry<K,V>	An <code>Entry</code> maintaining a key and a value.
AbstractMap.SimpleImmutableEntry<K,V>	An unmodifiable <code>Entry</code> maintaining a key and a value.
AbstractQueue<E>	This class provides skeletal implementations of some <code>Queue</code> operations.
AbstractSequentialList<E>	This class provides a skeletal implementation of the <code>List</code> interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

Java Doc (<https://docs.oracle.com/en/java/javase/25/docs/api/index.html>)

The screenshot shows the JavaDoc page for the `add` method in the `AbstractList` class. The page includes a navigation bar with tabs for OVERVIEW, CLASS, USE, TREE, PREVIEW, NEW, DEPRECATED, INDEX, SEARCH, and HELP. The current page is for `java.base > java.util > AbstractList`. A search bar is located in the top right corner.

The `add` method signature is `public void add(int index, E element)`. A blue callout box labeled "Method Signature" points to this signature. Below the signature, the text "Inserts the specified element at the specified position in this list (optional operation). Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices)." is present. A blue callout box labeled "Semantic description what the method does" points to this text.

Under the "Specified by:" section, it says "add in interface List<E>". Under "Implementation Requirements:", it says "This implementation always throws an UnsupportedOperationException." Under "Parameters:", it lists "index - index at which the specified element is to be inserted" and "element - element to be inserted". A blue callout box labeled "Parameter description" points to the "index" parameter description.

Under "Throws:", it lists several exceptions: "UnsupportedOperationException - if the add operation is not supported by this list", "ClassCastException - if the class of the specified element prevents it from being added to this list", "NullPointerException - if the specified element is null and this list does not permit null elements", "IllegalArgumentException - if some property of the specified element prevents it from being added to this list", and "IndexOutOfBoundsException - if the index is out of range (index < 0 || index > size())". A blue callout box labeled "Possible occurring errors" points to this list.

At the bottom of the page, the `remove` method is partially visible.

Theory Recap

Terminology

Overview:

<https://cgl.ethz.ch/teaching/parallelprog26/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

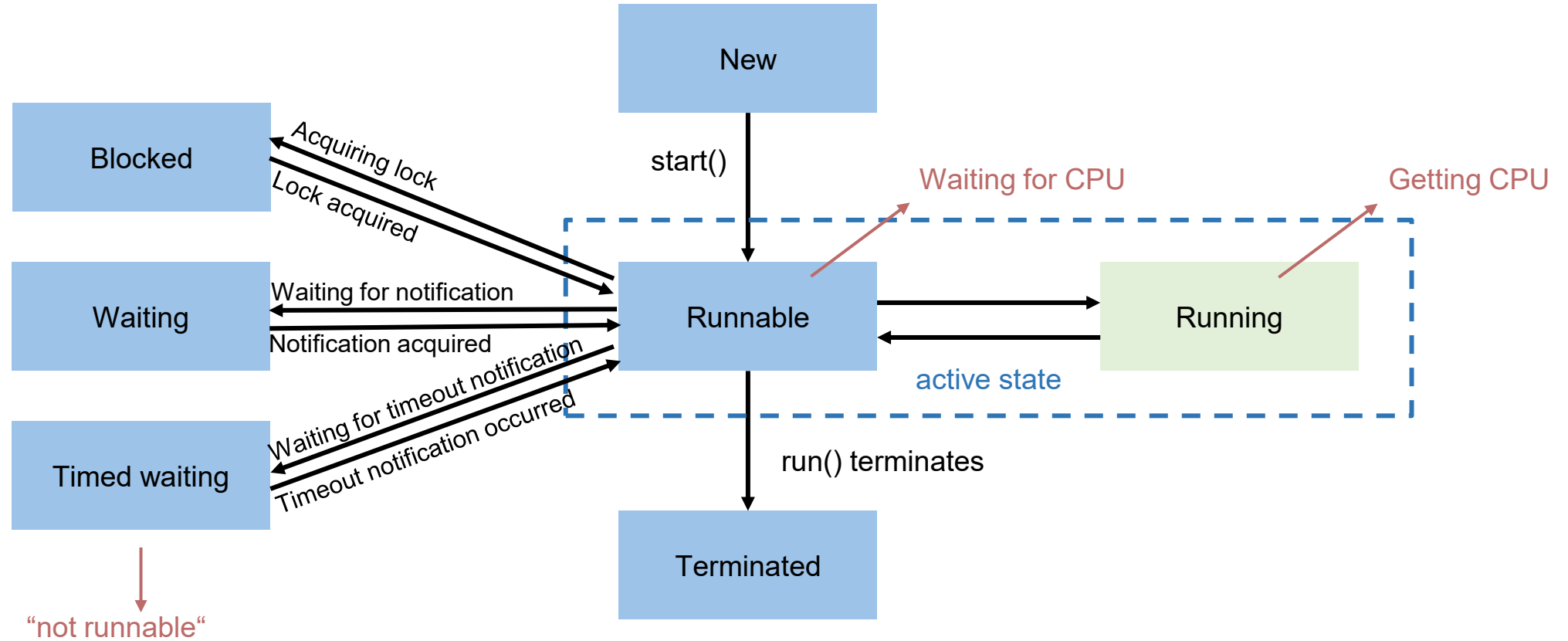
Concept of threads exists on various levels:

- 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

Life cycle of a Thread



Pre-Discussion Exercise 2

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



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

Task C

Mögliche Implementationen:

Measured execution time not always the same

→ Average over multiple runs (the more the better)

→ Calculate variance

Task D

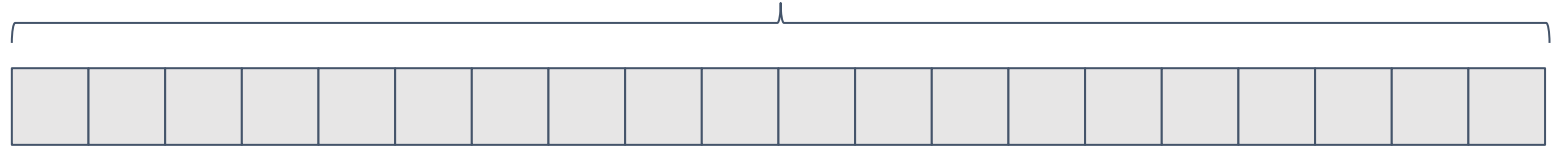
Before you parallelize the loop in Task E, design how the work should be split between the threads by implementing method `PartitionData`. Each thread should process roughly an equal amount of elements. Briefly describe your solution and discuss alternative ways to split the work.

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)

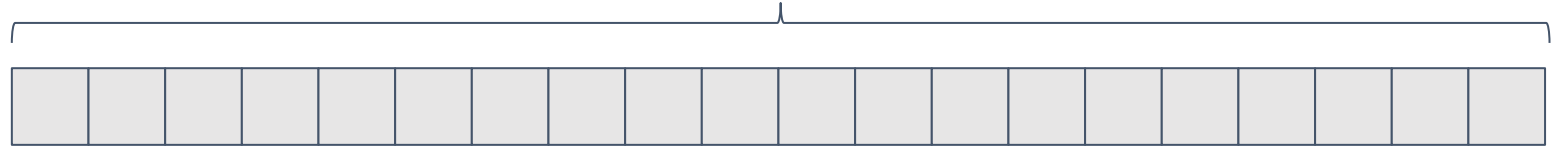
?

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

Mögliche Implementationen:

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 $(\text{length} > 0 \text{ and } \text{numPartitions} > 0)$ and $\text{length} < \text{numPartitions}$?
 - ??
 - ??
- And $(\text{length} \leq 0 \text{ or } \text{numPartitions} \leq 0)$?
 - ??
 - ??

```
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.

Empfehlung

Falls ihr nicht die Zeit habt, alle Aufgaben zu lösen, löst A-E.
F und G sind spannend, falls ihr dazu Zeit habt.

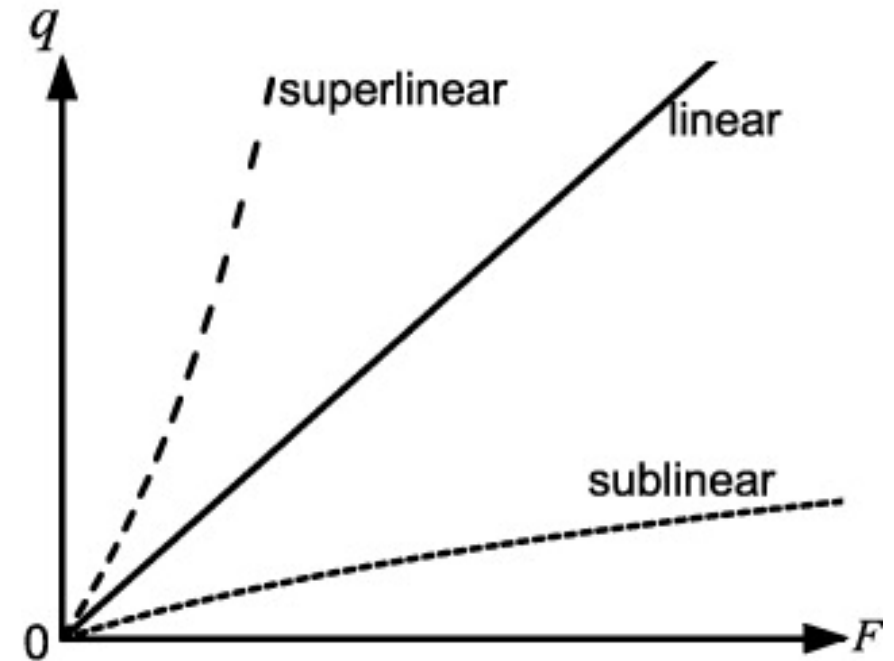
Ihr müsst einige Implementationsdetails selbst entscheiden, nicht alles ist vorgegeben!

Speedup

Sub-linear: usually

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

- Modern hardware properties (local/remote memory)
- Bug (this course assumes this)







<https://softwareengineering.stackexchange.com/questions/347748/can-sub-linear-still-be-a-straight-line>

Feedback-Umfrage



Past Exam Task

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

-  Beim Aufteilen eines Workloads sollte man so viele 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.

Past Exam Task

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

- Beim Aufteilen eines Workloads sollte man so viele 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.



https://quizizz.com/admin/assessment/65d34c567875b472cb3fa7be?source=lesson_share

Replace link with link to quiz