Praktikum Datamining und Sequenzanalyse

Einführung

Markus Fleischhauer – markus.fleischauer@uni-jena.de

Organisatorisches

Vorträge

Zu bestimmten Terminen via Zoom

freies Arbeiten

Montag und Freitag via Zoom (Breakout Rooms)

Projekte

1. Exakte Suche

Naive Suche, KMP, Boyer-Moore

2. Alignments

globales/lokales Alignment, Alignmentscore mit linearem Speicher

3. Clustering, phylogenetische Bäume

UPGMA, WPGMA , Neighbor-Joining

Projekte bauen aufeinander auf!

Gruppen und Projekte

- **3 Gruppen mit 3 oder 4 Teilnehmern**
  - jede Gruppe bekommt einen gemeinsamen Projektbereich („Group“) im Gitlab.
    - Projekte **immer** in eurer „Group“ (nicht im User Name-space) anlegen
    - Neues Projekt: `Settings → Repository → Protected Branches → Unprotect Master`
  - Jedes Projekt bekommt ein eigenes git-Repository

`https://git.bio.informatik.uni-jena.de`

Gruppen und Projekte

In der heutigen Veranstaltung:

- Gitlab Account erstellen
- Unsername beliebig, Echten Namen angeben,
  Uni E-Mail-Adresse Verwenden
- Gruppen bilden
- Pro Gruppe eine E-Mail mit den Gitlab
  Benutzernamen der Gruppenmitglieder an:

  `markus.fleischauer@uni-jena.de`

`https://git.bio.informatik.uni-jena.de`

Lernziele

- Softwareprojekten im Team umsetzen
- In der Theorie bekannte Algorithmen implementieren
- Auswirkung der Implementierung auf die Performance
- Arbeiten mit:
  - Versionskontrolle (Git)
  - Build-Management (Gradle)
  - IDE - Integrated Development Environment (IntelliJ)

Aufgaben

Ein Aufgabenblatt pro Projekt (online)

1. Programmieren
   - Wartbarer Code – Objektorientiert (OOP)
   - Effizienter Code
   - Command Line Interface (**CLI**)
   - Dokumentation
2. Evaluation (Protokoll)
3. Präsentation (Vorträge)
   - Siehe Leitfaden Projektpräsentation (online)

Tools für die Aufgaben

- Objektorientierte Programmierung mit **`Java`**
- Dokumentation mit **`Javadoc`**
- Versionskontrolle mit **`git`**
- Projektmanagement mit **`Gradle`**
- Als IDE verwenden wir **`IntelliJ IDEA`**
- Benutzerinterface (CLI) mit **`picocli`**

Auswertung und Vortrag

Auswertung (Aufgabenblatt online)

- Aufgabenzettel bearbeiten
- Laufzeitmessungen
- auf mögliche Fehler/Probleme eingehen
- Protokoll **vor** Vortrag abgeben! (`git push`)

Vortrag (Leitfaden online)

- Jede Gruppe trägt einmal vor (Dauer **ca. 30min**)
  - Vorstellung und Diskussion der Ergebnisse
  - Vorführen der Benutzerschnittstelle
  - Überblick zur Implementierung

Java und OOP

Grundlagen

Klassen definieren Eigenschaften v. Objekten (Blaupausen)

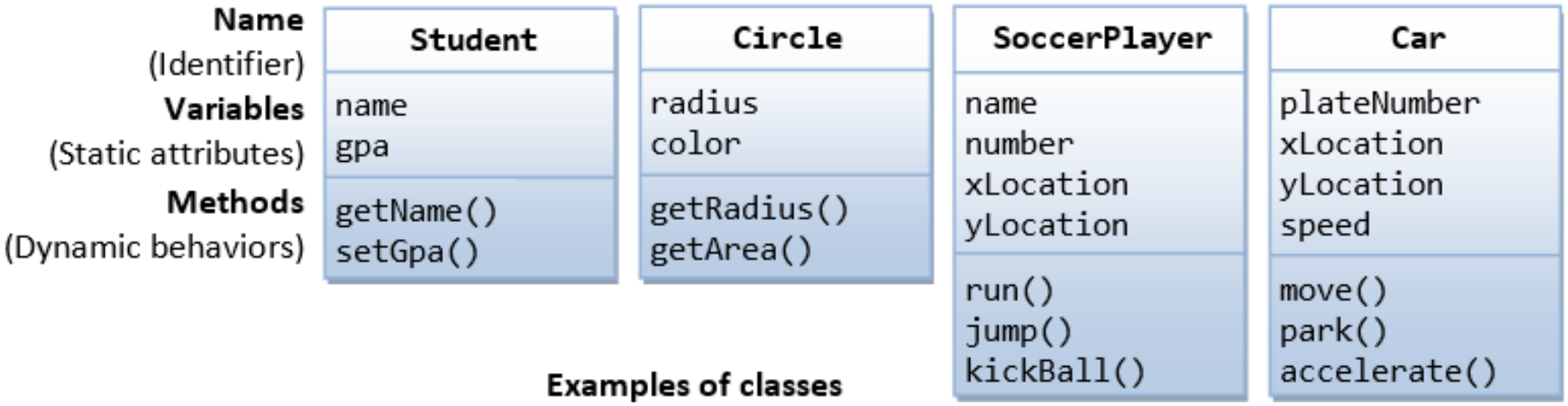
1. Name/Identifier
2. Attribute (Felder, Variablen)
3. Behaviour (Methoden)

Objekte sind Instanzen von Klassen

1. Felder – Belegung
2. Methoden – Abhängig v. Belegung

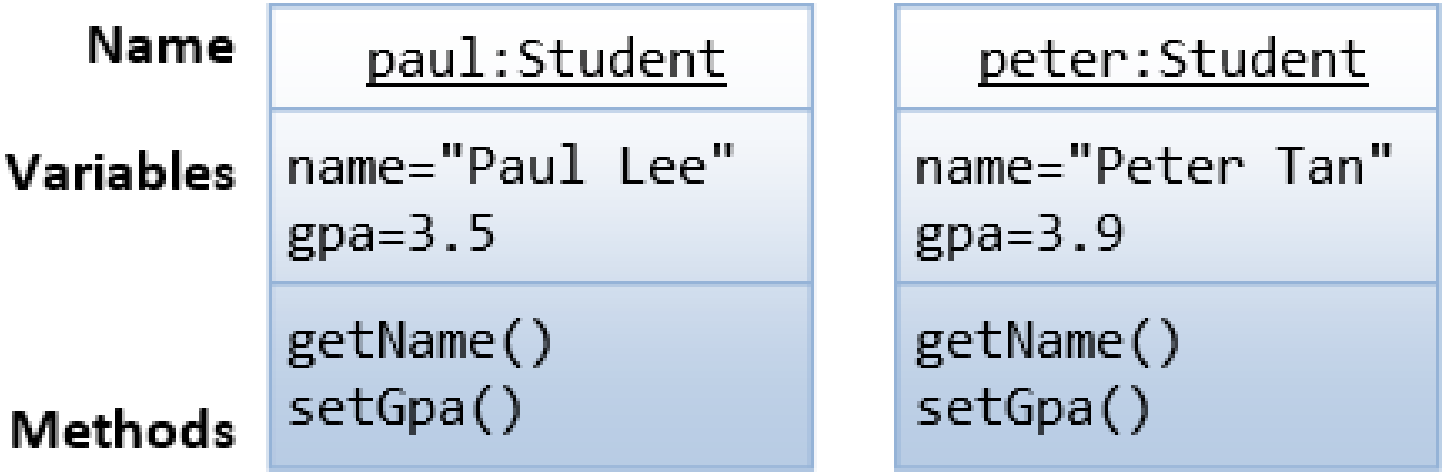
Repräsentation von physikalischen oder logischen Einheiten der Echtwelt

Klassen



Examples of classes

Objekte - Instanzen



Two instances - paul and peter - of the class Student

Sichtbarkeit

Modifier	Class	Package	Subclass	World
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default	✓	✓	✗	✗
private	✓	✗	✗	✗

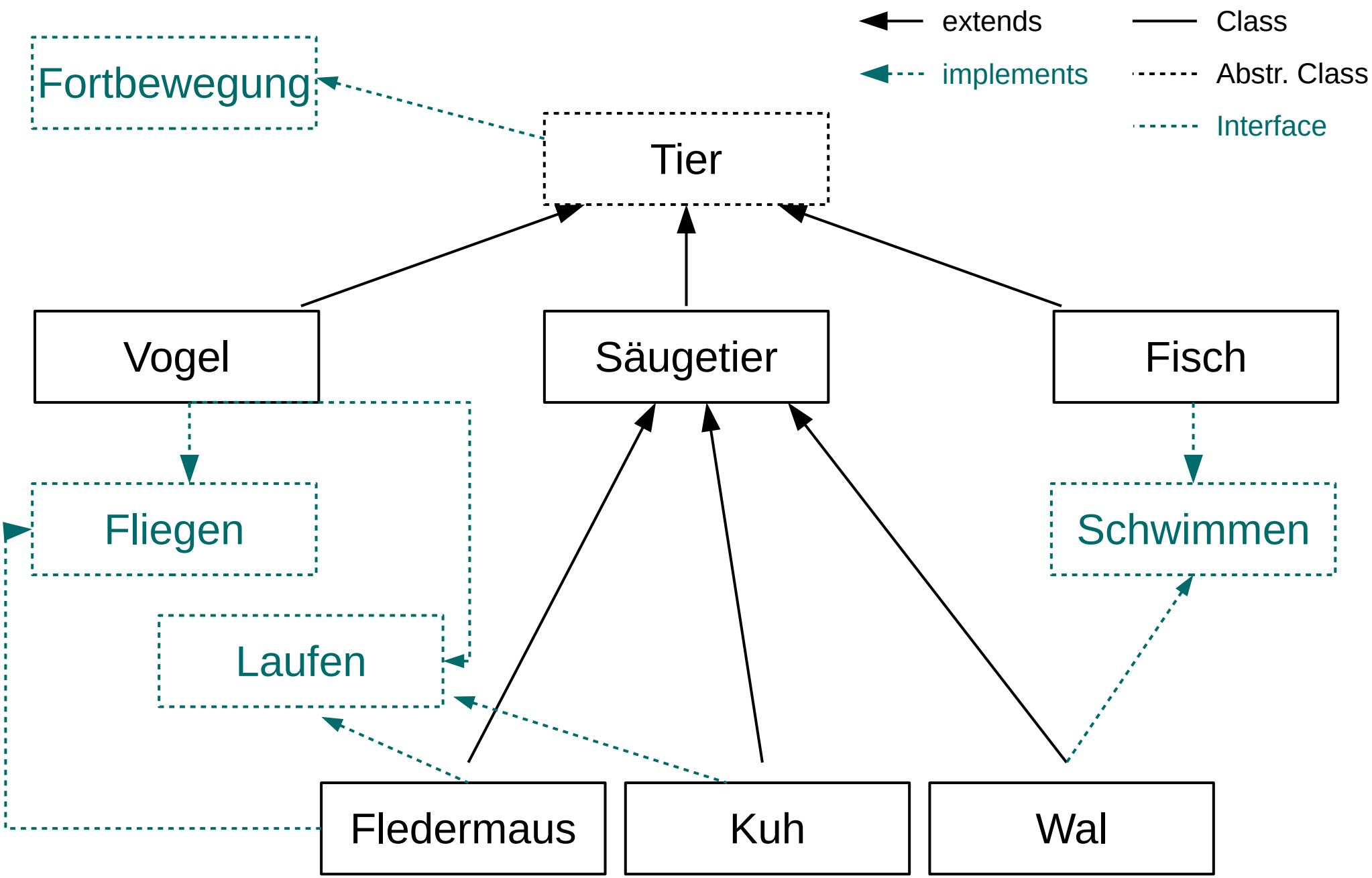
Klassentypen in Java

Klasse	• Instanzierbar • Nur aus-implementiert Methoden • Beliebige Felder und Sichtbarkeiten • Einfachvererbung
Abstrakte Klasse	• **Nicht** Instanzierbar • Abstrakte und implementierte Methoden • Beliebige Felder und Sichtbarkeiten • Einfachvererbung
Interface	• **Nicht** Instanzierbar • Abstrakte Methoden mit default Implementierung • keine Felder, nur public Methoden • Mehrfachvererbung

Wann welche Klassentypen

Klasse	• Beliebige Objekte
Abstrakte Klasse	• Vermeidung von doppeltem Code • Basis Implementierung bei der essentielle Funktionalität fehlt
Interface	• Definition von Schnittstellen • Gewährleistung von Funktionalitäten ohne die konkrete Implementierung vorzugeben

Vererbung


extends
implements
Class
Abstr. Class
Interface
Fortbewegung
Tier
Vogel
Säugetier
Fisch
Fliegen
Schwimmen
Laufen
Fledermaus
Kuh
Wal

Interfaces?

```
public interface Distance{
    public double distance(Point p1, Point p2);
}

public class ManhattanMetrik() implements Distance{
    public double distance(Point p1, Point p2) {…};
}
public class EuclideanMetrik() implements Distance{
    public double distance(Point p1, Point p2) {…};
}
```

```
public class NearestNeighbor{
    public NNResult cluster(List<Point> points, Distance distance){
        …
    }
}
```

Mehrfachvererbung?

```
public abstract class Tier{
    public abstract void move();
}

public interface CanFly{
    public void fly();
}
```

```
public class Bird extends Tier implements CanFly{
    public void move(){
        //do something
    }
    public void fly(){
        //do something
    }
}
```

Polymorphismus

```
public class Clock{
   public static getFormat(){…}
   public void setTime(long ns){…}
   public void setTime(int h, int m, int s, int ms){...}
}

public class MoreSpecificClock extends Clock{
   public static getFormat(){…}
  @Override
   public void setTime(long ns){…}
}
```

Ueberladen

Ueberdecken

Ueberschreiben

Zugriff auf Attribute durch Getter/Setter

```
public class WithoutEncapsulation{

    public String value;

    public void printLength(){
        System.out.println("Länge: " + value.length());
    }

    public static void main(String[] args){
        WithoutEncapsulation we = new WithoutEncapsulation();
        we.value = null;
        we.printLength();   ⟶   NullPointerException
    }

}
```

Zugriff auf Attribute durch Getter/Setter

```
public class WithEncapsulation{

    private String value;

    public void setValue(String value){
        if(value == null) this.value = “”;
        else this.value = value;

    public void printLength(){
        System.out.println(“Länge: “ + value.length());
    }

    public static void main(String[] args){
        WithoutEncapsulation we = new WithoutEncapsulation();
        we.setValue(null);
        we.printLength();  ⟶  Länge: 0
    }

}
```

Hinweise

Zeit messen

via command line time

```
#$ time sleep 10
real 0m10.116s
user 0m0.001s
sys 0m0.007s
```

via Java

```
System.currentTimeMillis()
System.nanoTime()
long time = System.nanoTime();
//do something
long duration = System.nanoTime() - time;
```

Zeit messen

Vorgehen

- mehrfach messen
- Minimum aller Messungen verwenden

Testumgebung beschreiben

- Betriebssystem
- Hauptspeicher
- CPU
- Java VM version
- Java Heapspace (`-Xmx`)

IO

```
try(BufferedReader reader =
          new BufferedReader(new FileReader(path))){
   String temp = null;
   while((temp = reader.readLine()) != null){
      //werte aus
   }
}catch(IOException e){
   // ...
}
```

Hinweis: Die java.nio Bibliotheken verwenden auch „buffer“.

Strings konkatenieren

```
public static String concatenate(String... s)
{
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < s.length; i++) {
        sb = sb.append(s[i]);
    }

    return sb.toString();
}
```

Hinweis: Wen ihre viele Konkatenierungen, z.B. in einer Schleife, durchführt solltet ihr den ‚+‘ Operator vermeiden und `StringBuilder` verwenden.

Fragen?

Jetzt bzw. Freitag gemeinsam in der Gruppe:

1. Donwload und Installieren eines JDK11

   https://de.azul.com/downloads/zulu-community/?version=java-11-lts&architecture=x86-64-bit&package=jdk

2. Download und installieren von Gradle

   https://gradle.org/next-steps/?version=6.7&format=bin

3. Download und installieren von Git

   https://git-scm.com/downloads

   Mac: Xcode aus AppStore, **Win:** Installer verwenden

   Linux: Paketquellen

4. Download und installieren von IntelliJ (Community)

   https://www.jetbrains.com/de-de/idea/download/#section=windows

Jetzt bzw. Freitag gemeinsam in der Gruppe:

- ~~Projekt „exact-search“ in euer Gruppe erstellen~~
  - ~~Readme anlegen, Master unprotecten~~
- JDK, Gradle und IntelliJ zum laufen bekommen
- Projekt in ein Verzeichnis auf eurem System clonen:

```
git clone https://git.bio.informatik.uni-jena.de/datamining/ws[year]/group[num]/exact-search.git
```

- .gitignore File in Verzeichnis packen (online)

```
git clone add .gitignore
```

```
git commit -m "add .gitignore"
```

```
git push
```

`https://git.bio.informatik.uni-jena.de`

Jetzt bzw. Freitag gemeinsam in der Gruppe:

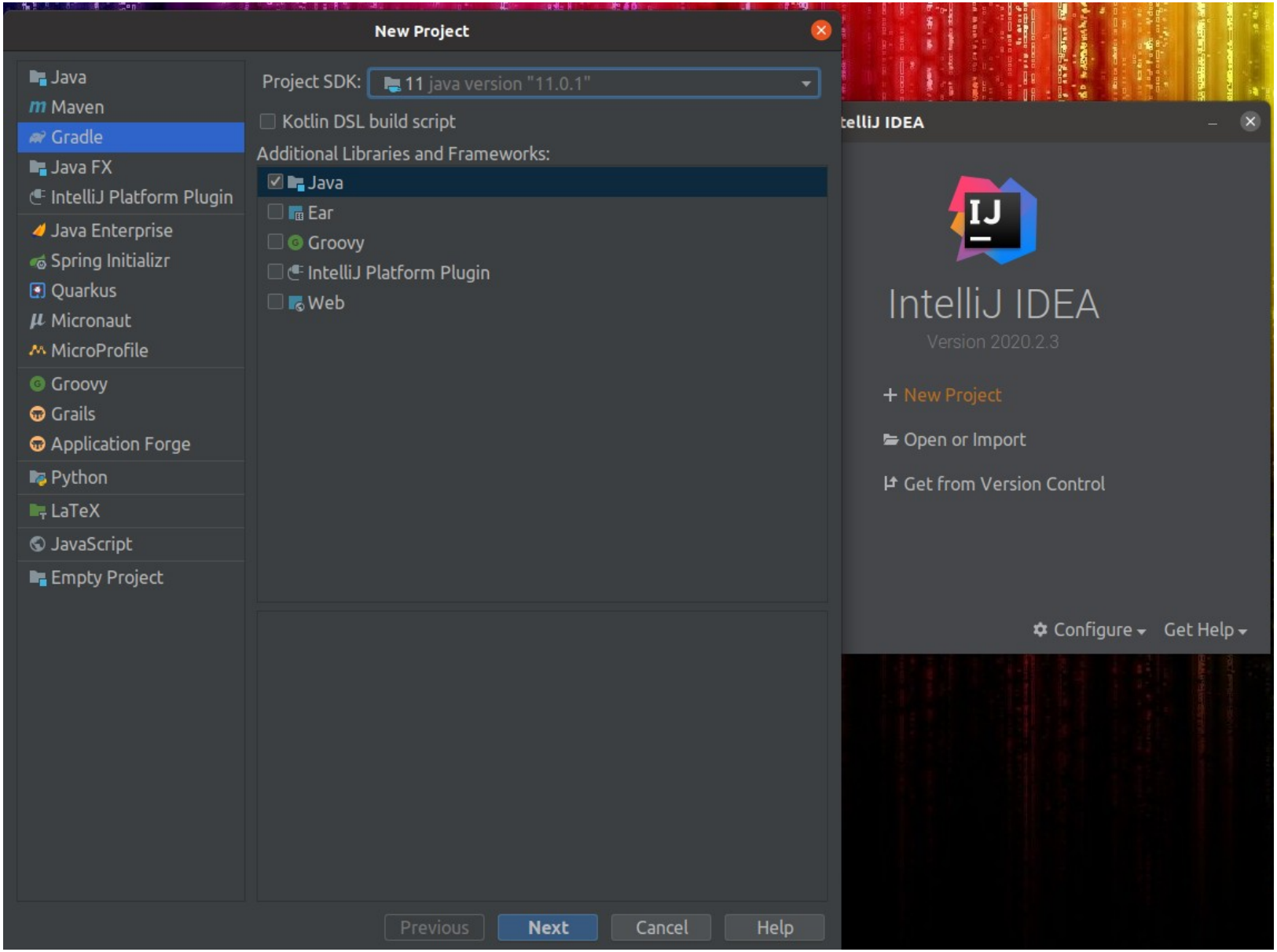
- Mit IntelliJ ein neues Gradle project anlegen
  - Files → new → Project → Gradle → Select Java

```
ArtifactId 'exact-search'
GroupId 'de.unijena.bioinf.dm.20[year].grp[num]'
Version '1.0-SNAPSHOT'
```

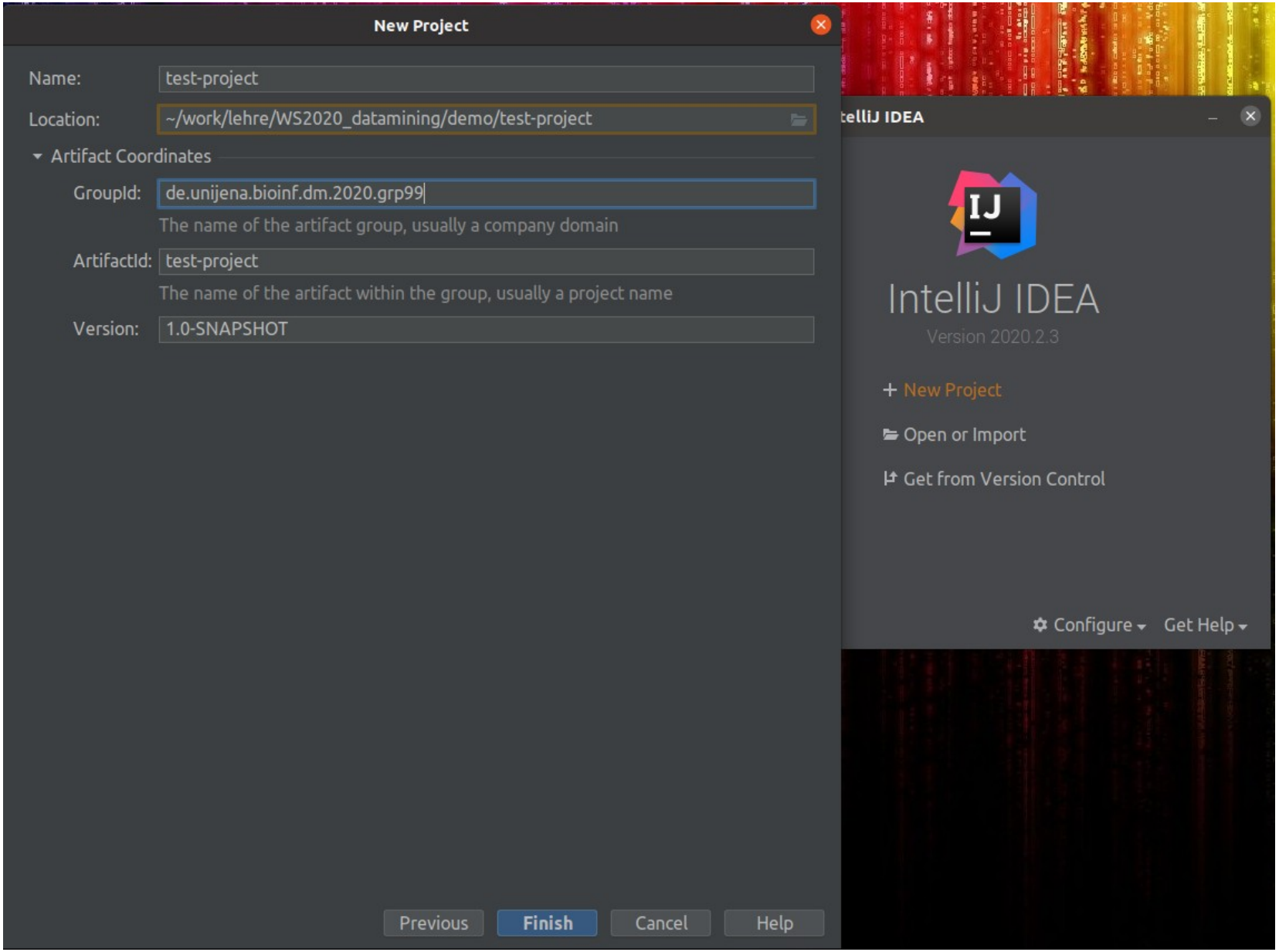
- Als Speicherort geclonetes git repo auswählen
- .gitignore, build.gradle, settings.gradle zu git hinzufügen
- Änderungen commiten und pushen

`https://git.bio.informatik.uni-jena.de`

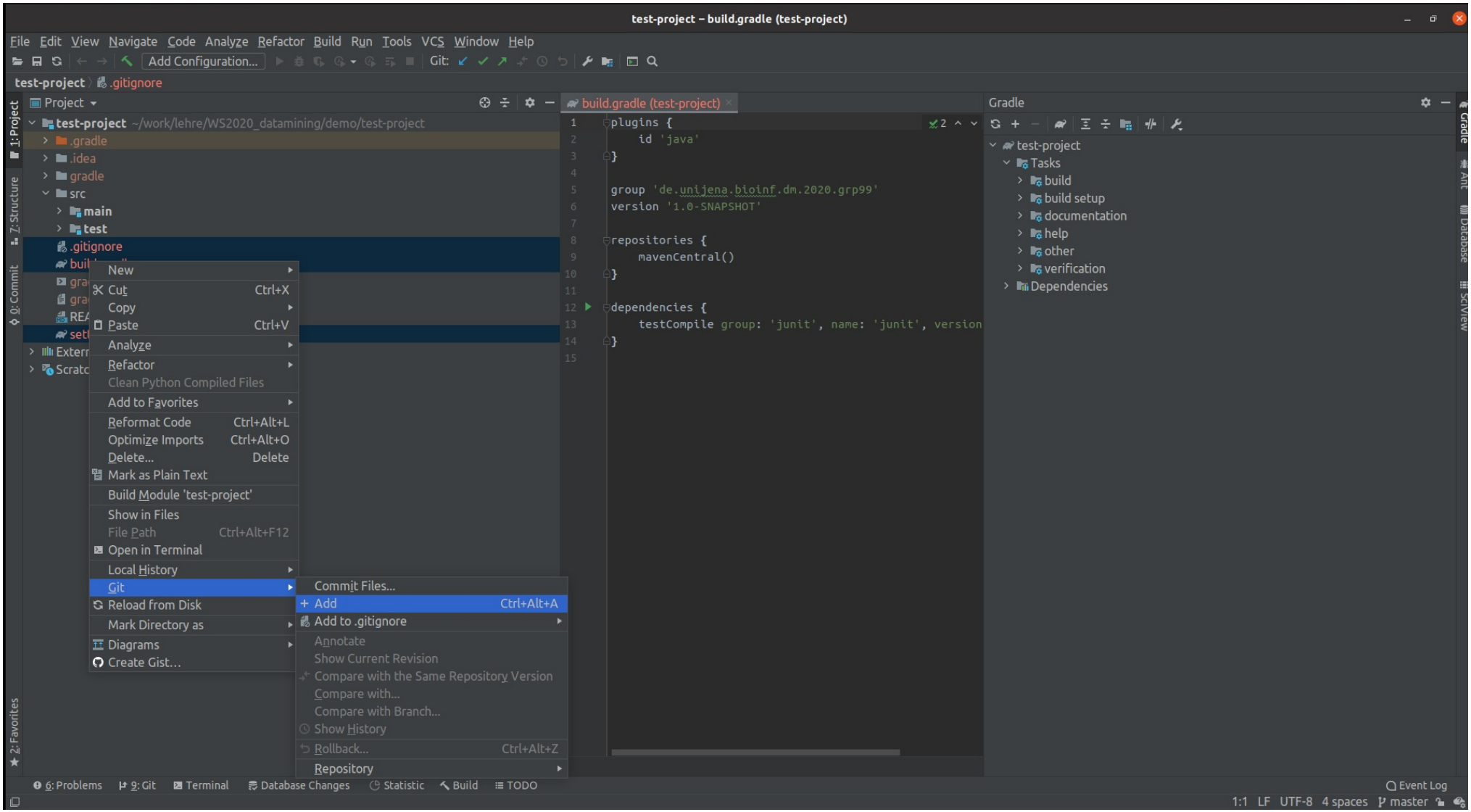
Neues Gradle-Java-Projekt in IntelliJ



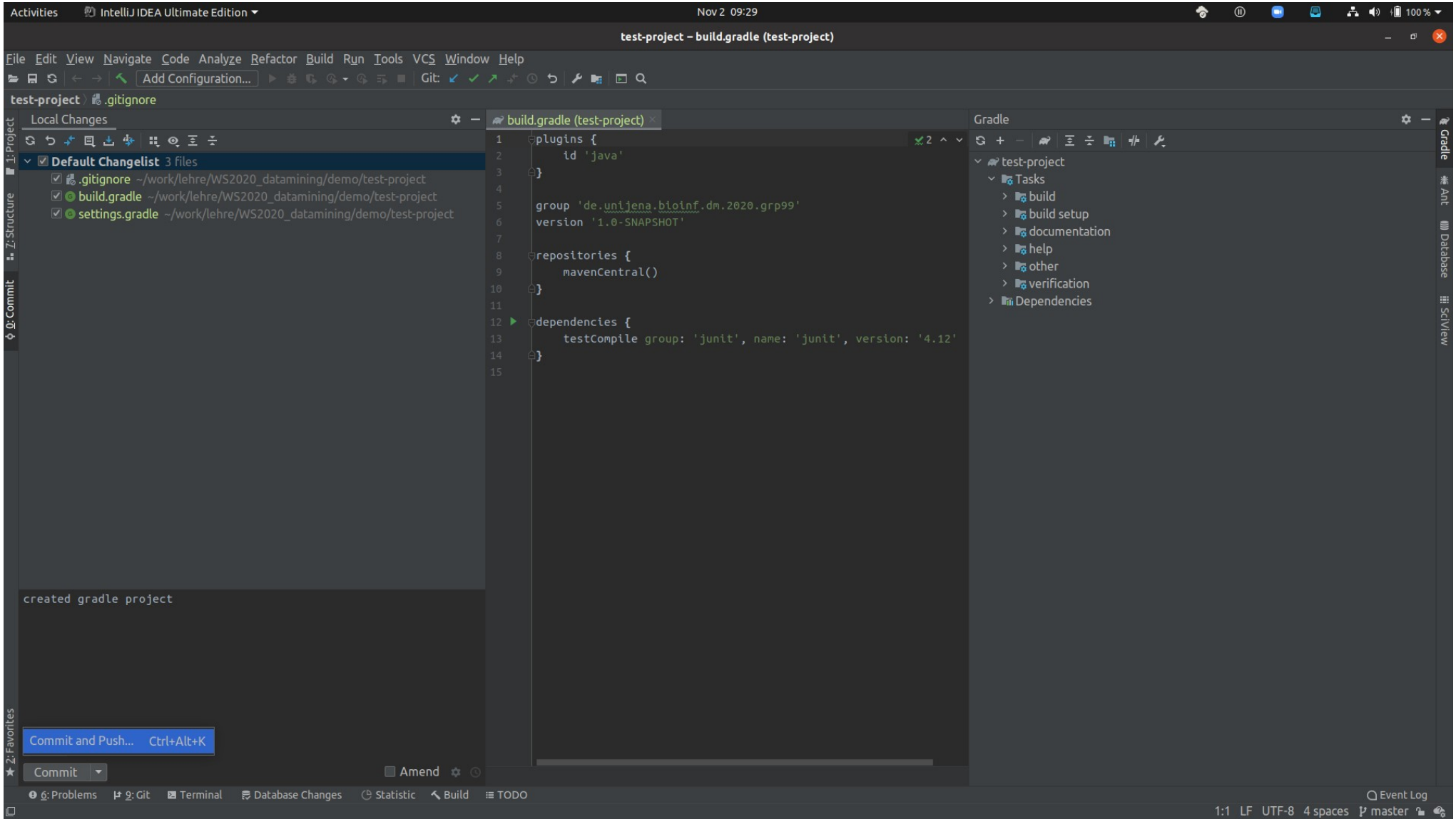
Pfad auswählen und Artifakt-Infos eintragen



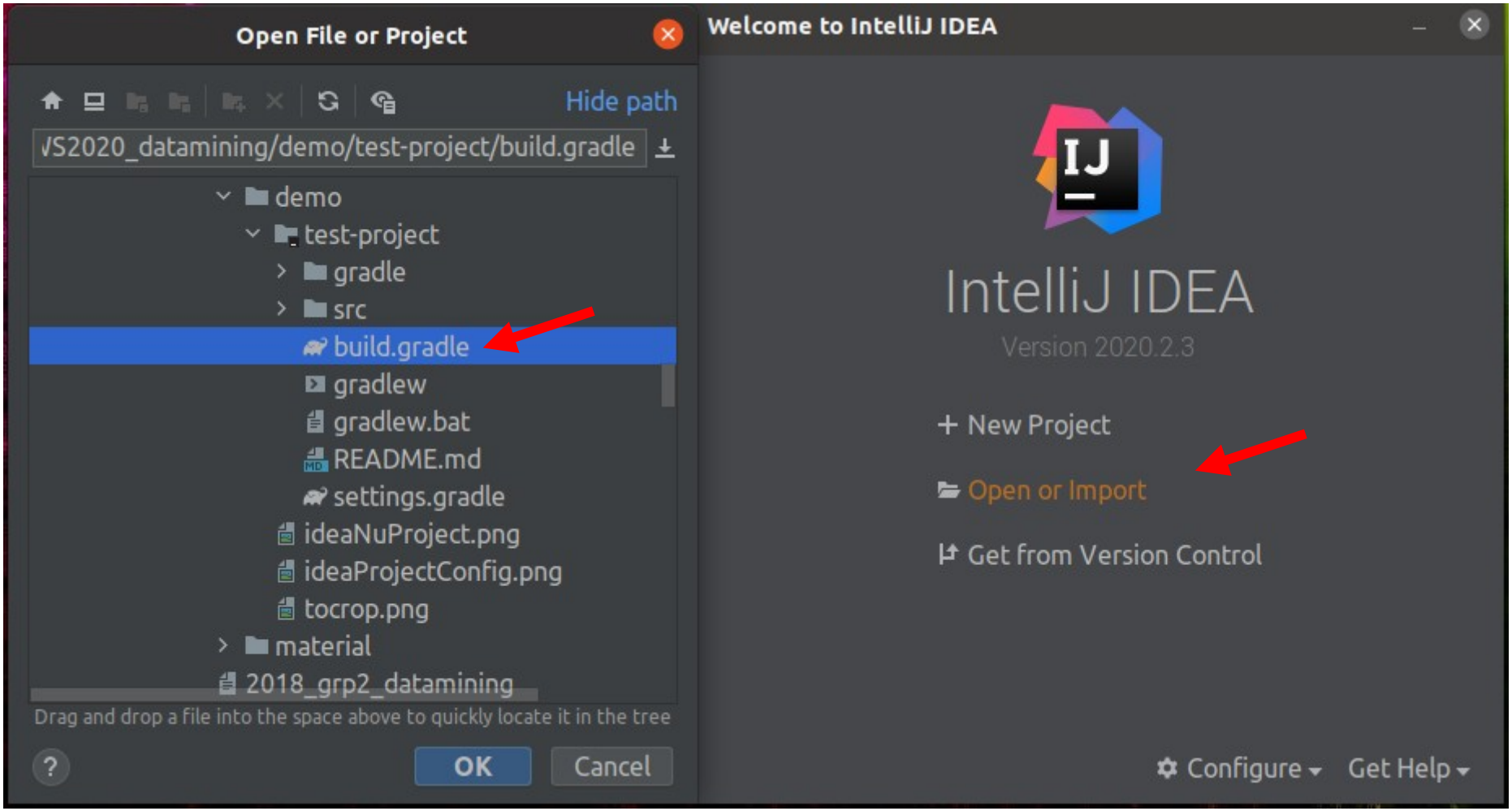
Dateien zu git hinzufügen



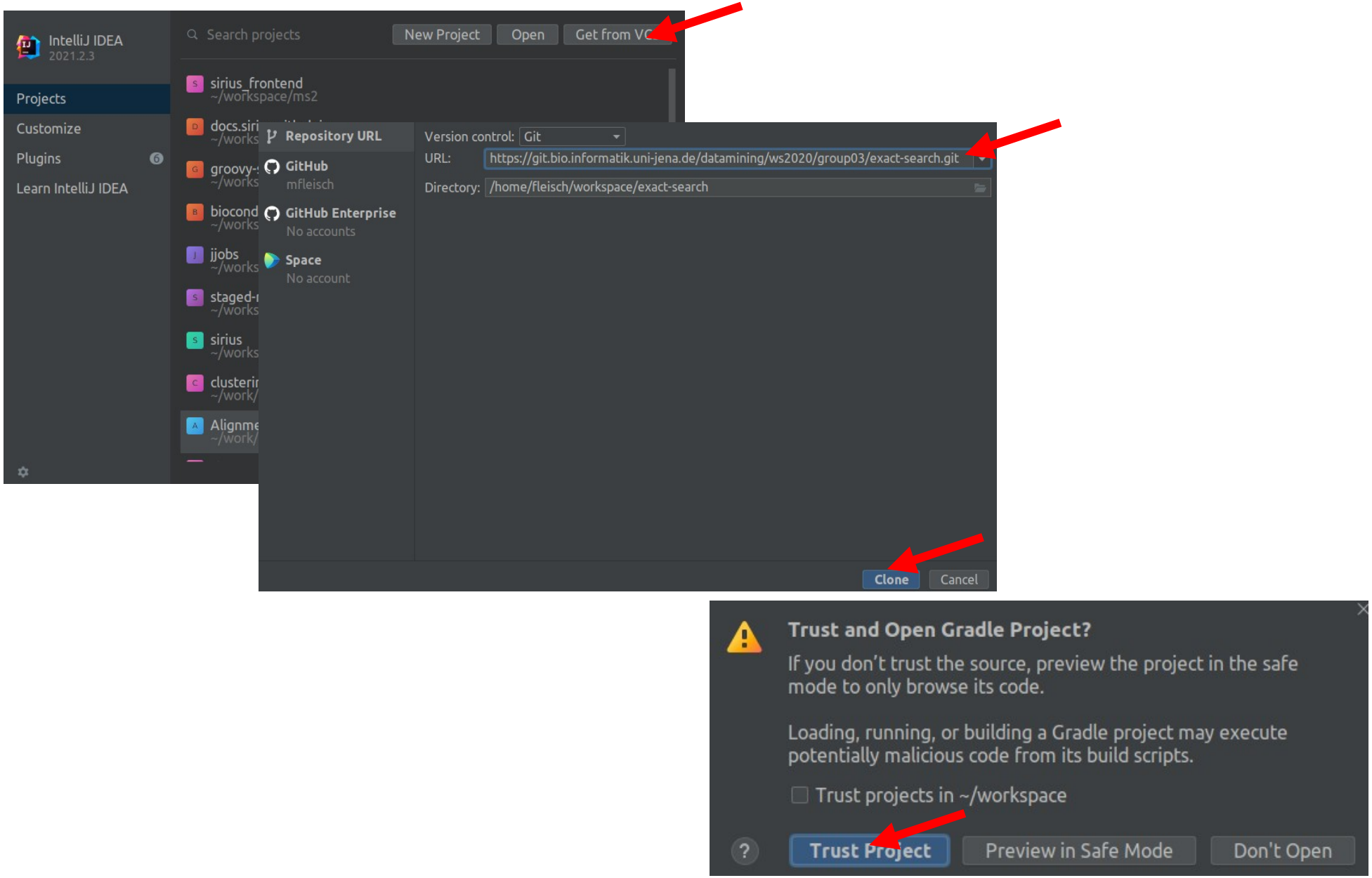
Commit & Push der neuen Dateien



Import eines existierendes Projektes



Import aus Versionskontrolle (GitLab)



System gradle verwenden