MapReduce job mit Java in Eclipse

1. Einleitung

Eclipse kann (unter Nutzung der „Eclipse public license“) frei von www.eclipse.org für alle gängigen Plattformen heruntergeladen werden. Anbei ein paar Screenshots für Installation unter Linux. Es kann die aktuellste Version verwendet werden.

Um gradle als Buildsystem zu verwenden, ist es am einfachsten, das plugin für IntelliJ zu installieren. Für die Verwendung auf der Kommandozeile kann es nötig sein, den Installationspfad des gradle-plugins in den Pfad unter „Umgebungsvariablen“ aufzunehmen oder Gradle direkt von https://gradle.org/releases/ herunterzuladen und zu installieren.

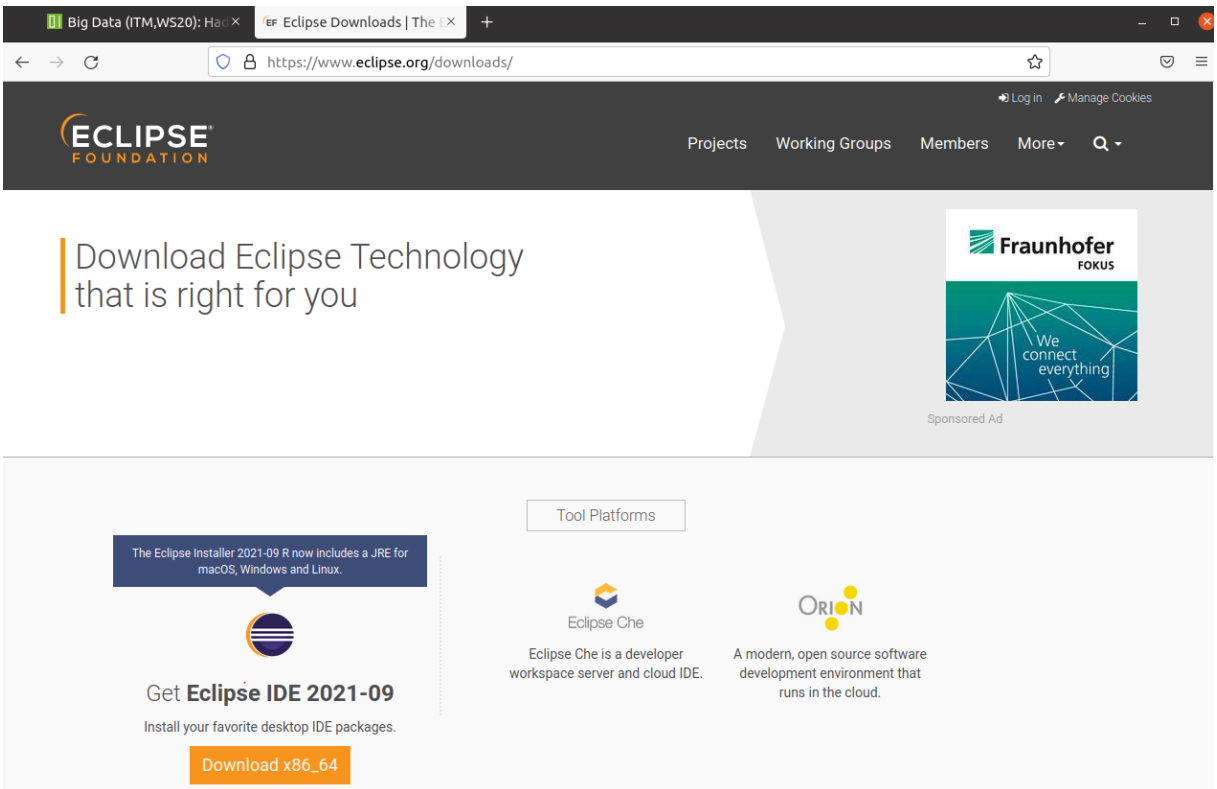
2. Generell

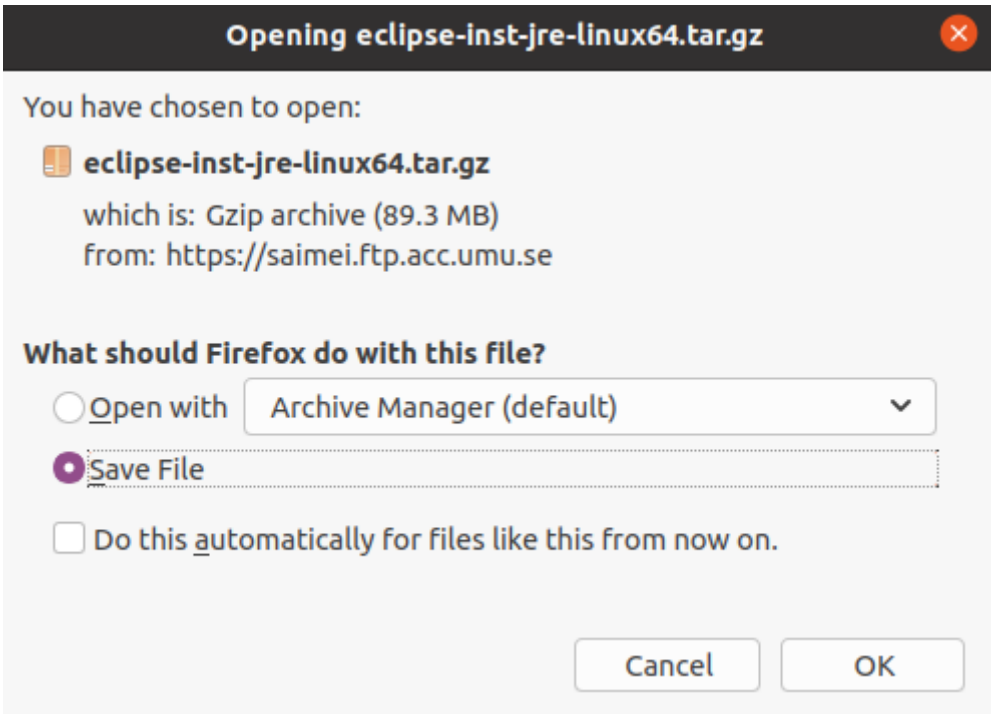
Folgende 2 Varianten bieten sich an, das Projekt zu bauen:

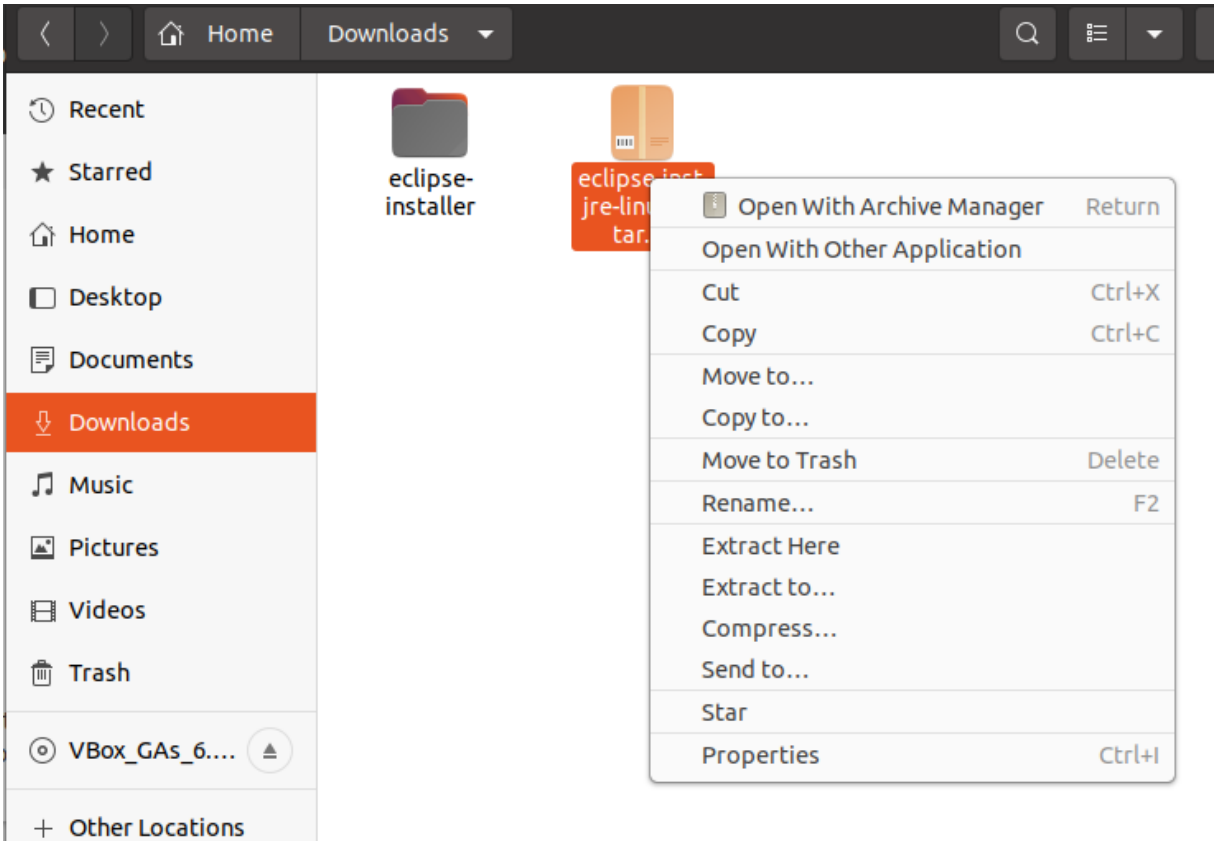
1. Mit Hilfe eines externen build systems (vorbereitetes build.gradle File)
2. Mit Hilfe von Eclipse selbst als Build-System (erfordert etwas mehr Vorbereitung, ist aber vielleicht besser zu verstehen)

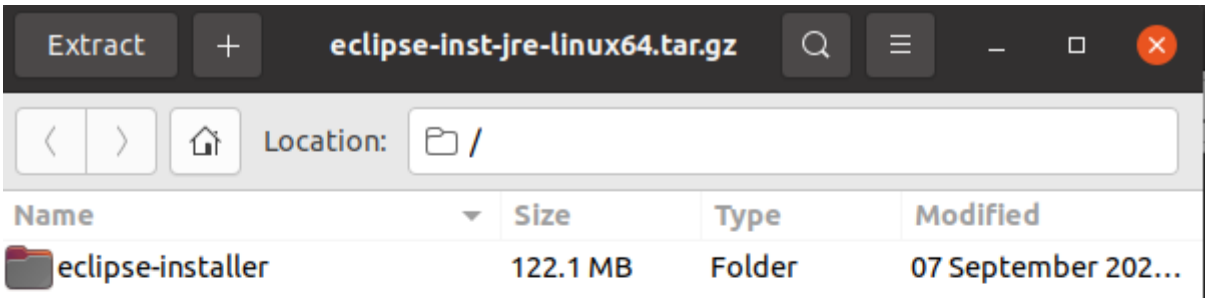
Im Folgenden wird bei Abweichungen zwischen den Varianten stets die Notation @gradle bzw. @eclipse verwendet.

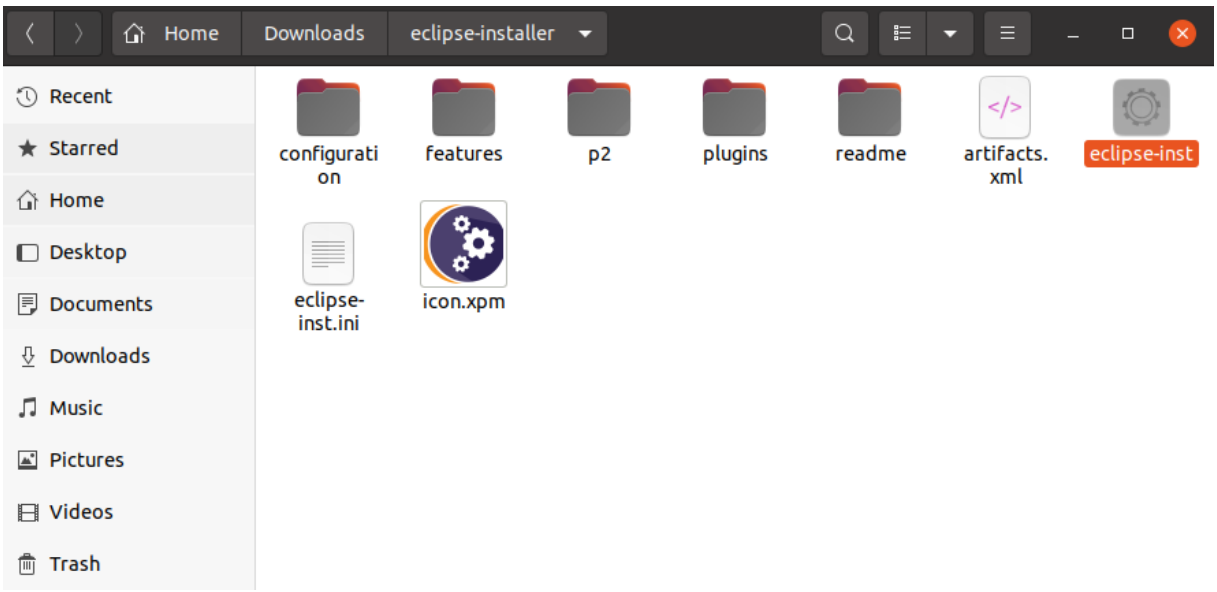
3. Installation

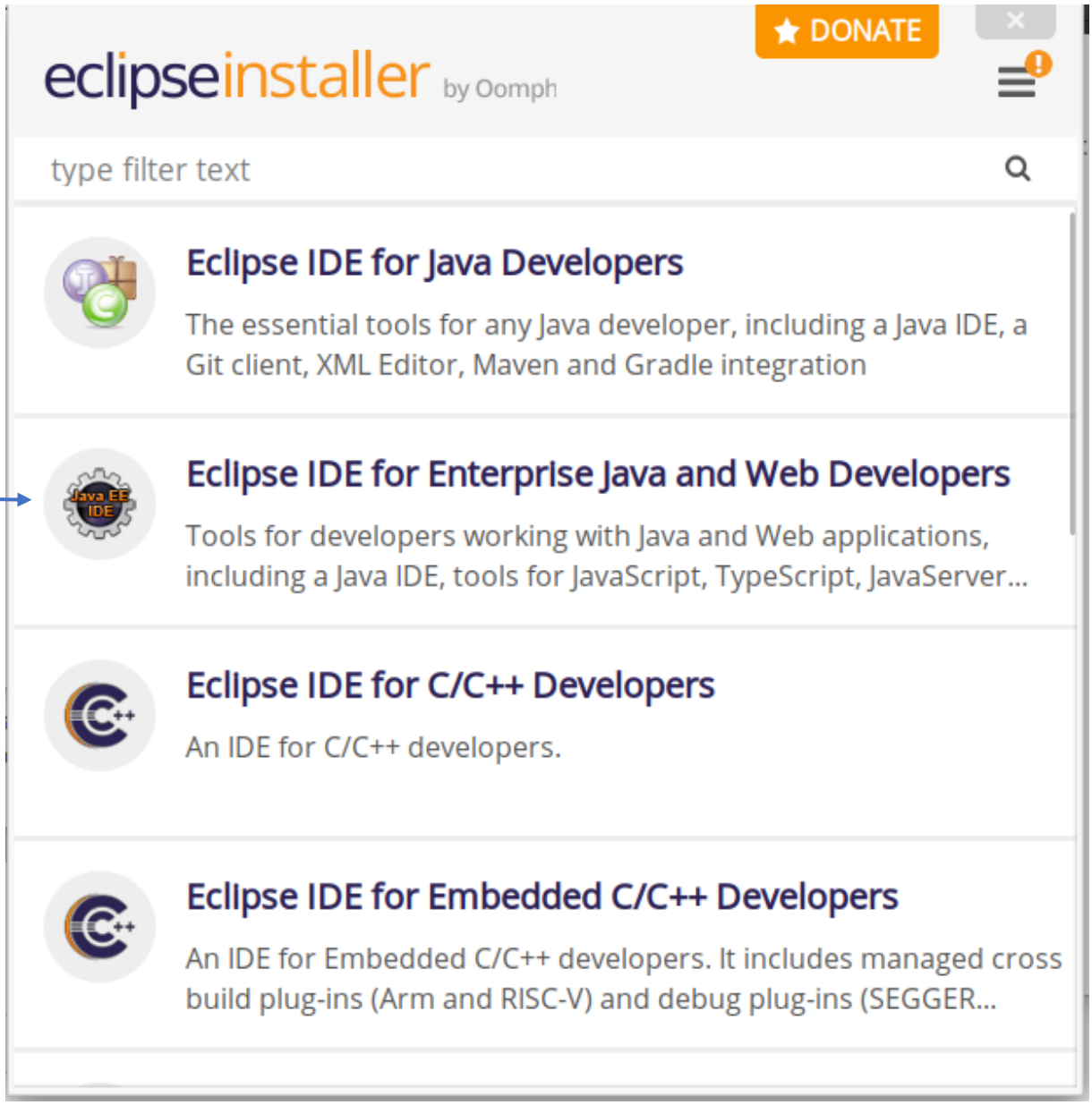



Opening eclipse-inst-jre-linux64.tar.gz
You have chosen to open:
eclipse-inst-jre-linux64.tar.gz
which is: Gzip archive (89.3 MB)
from: https://saimei.ftp.acc.umu.se
What should Firefox do with this file?
Open with
Archive Manager (default)
Save File
Do this automatically for files like this from now on.
Cancel
OK

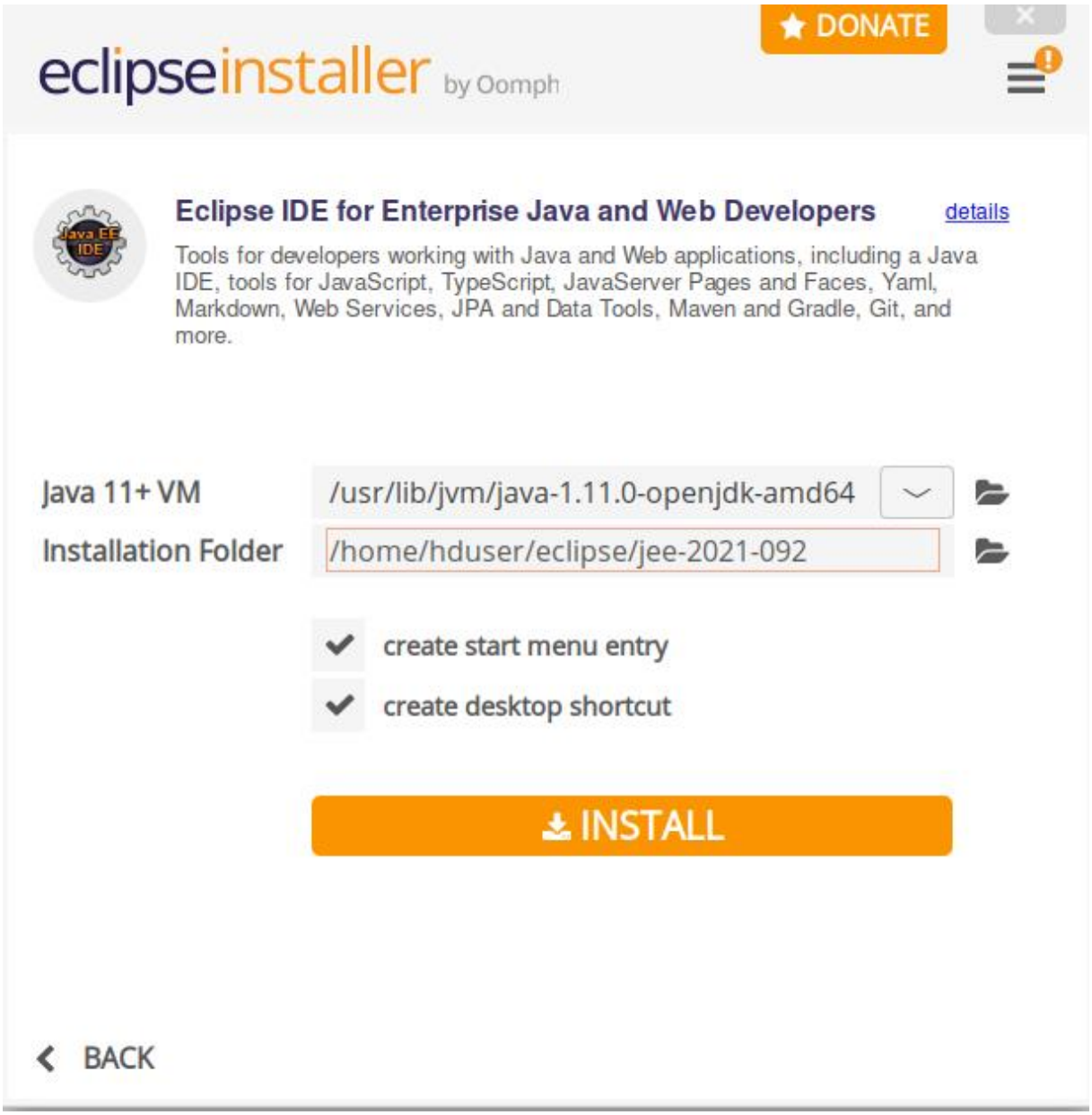

Home
Downloads
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
VBox_GAs_6....
Other Locations
eclipse-installer
Open With Archive Manager
Return
Open With Other Application
Cut
Ctrl+X
Copy
Ctrl+C
Move to...
Copy to...
Move to Trash
Delete
Rename...
F2
Extract Here
Extract to...
Compress...
Send to...
Star
Properties
Ctrl+I

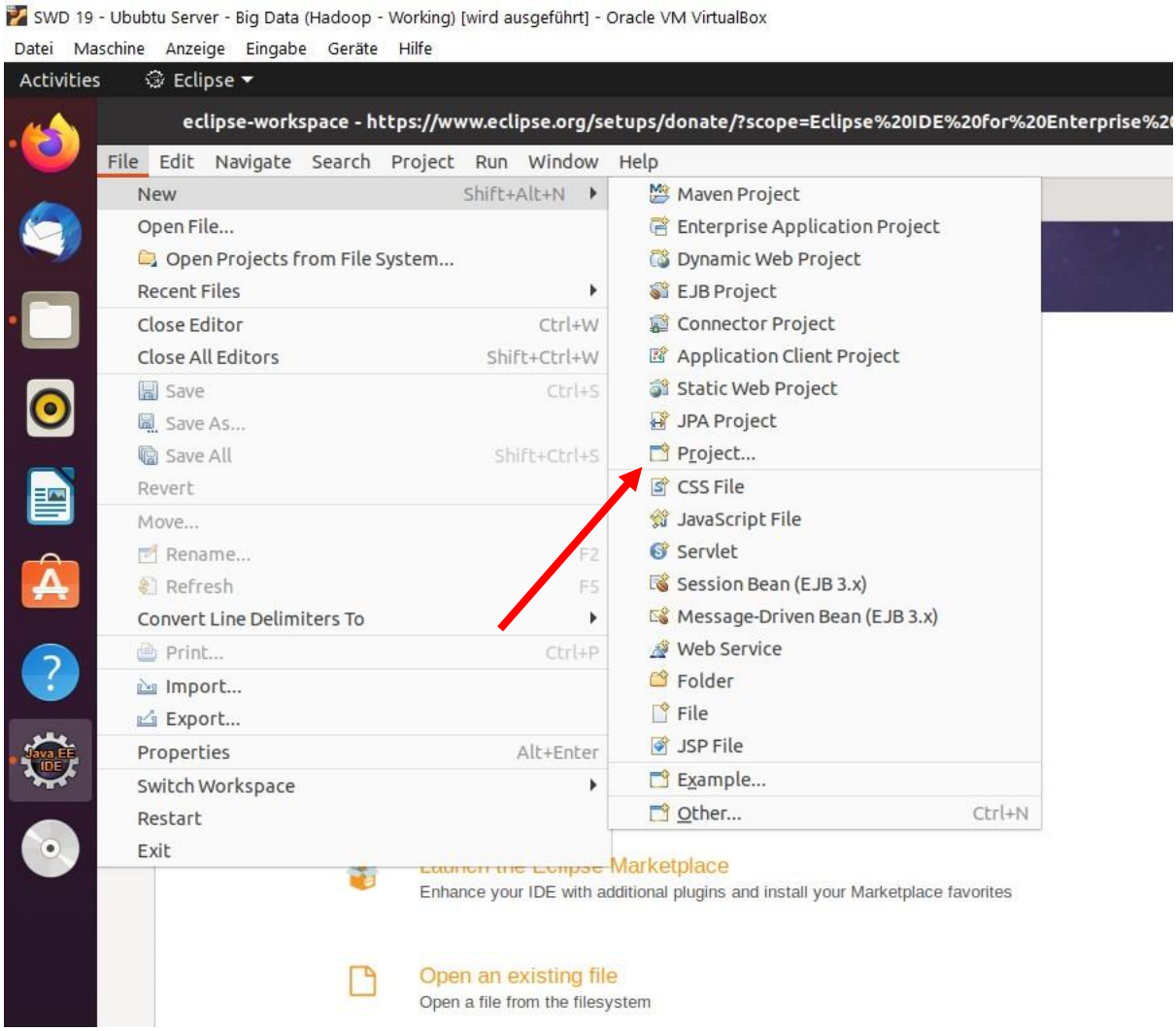

Extract
eclipse-inst-jre-linux64.tar.gz
Location:
/
Name
Size
Type
Modified
eclipse-installer
122.1 MB
Folder
07 September 202...


Home
Downloads
eclipse-installer
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
configuration
features
p2
plugins
readme
artifacts.xml
eclipse-inst
eclipse-inst.ini
icon.xpm



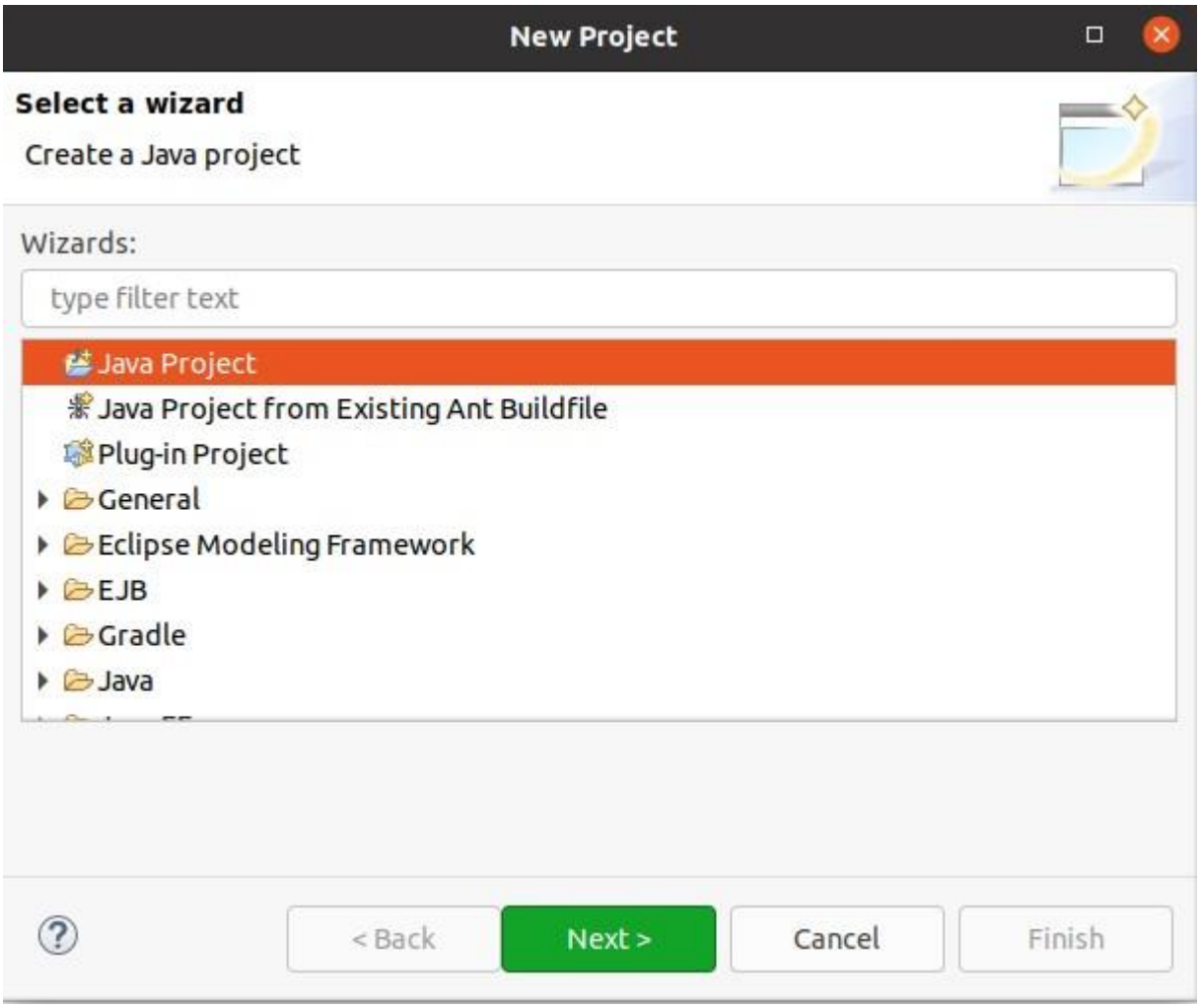
Als Java Version muss eine Version >= 11 installiert werden.


eclipseinstaller by Oomph
DONATE
Eclipse IDE for Enterprise Java and Web Developers
details
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.
Java 11+ VM
/usr/lib/jvm/java-1.11.0-openjdk-amd64
Installation Folder
/home/hduser/eclipse/jee-2021-092
create start menu entry
create desktop shortcut
INSTALL
BACK

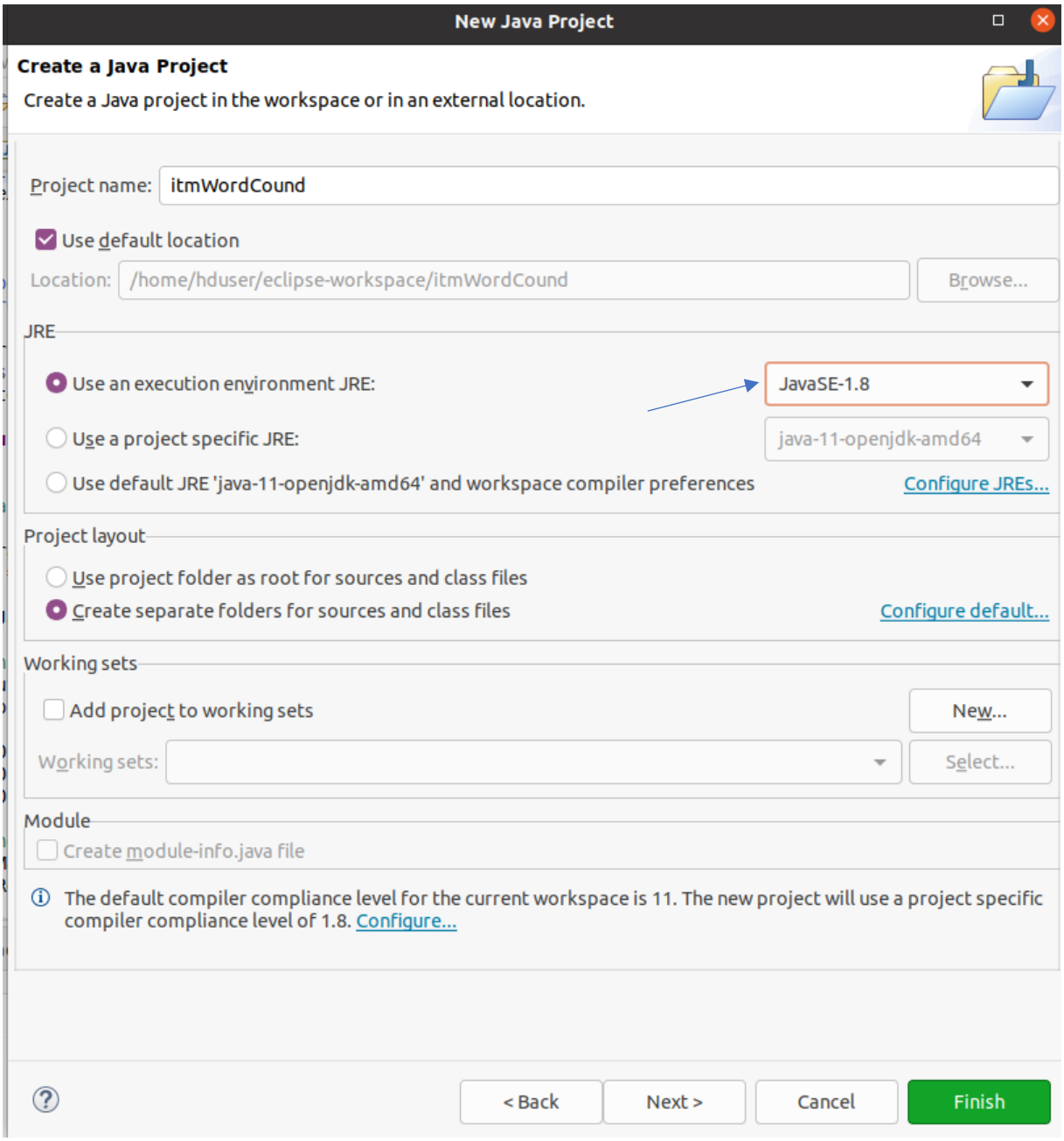


4. Project anlegen

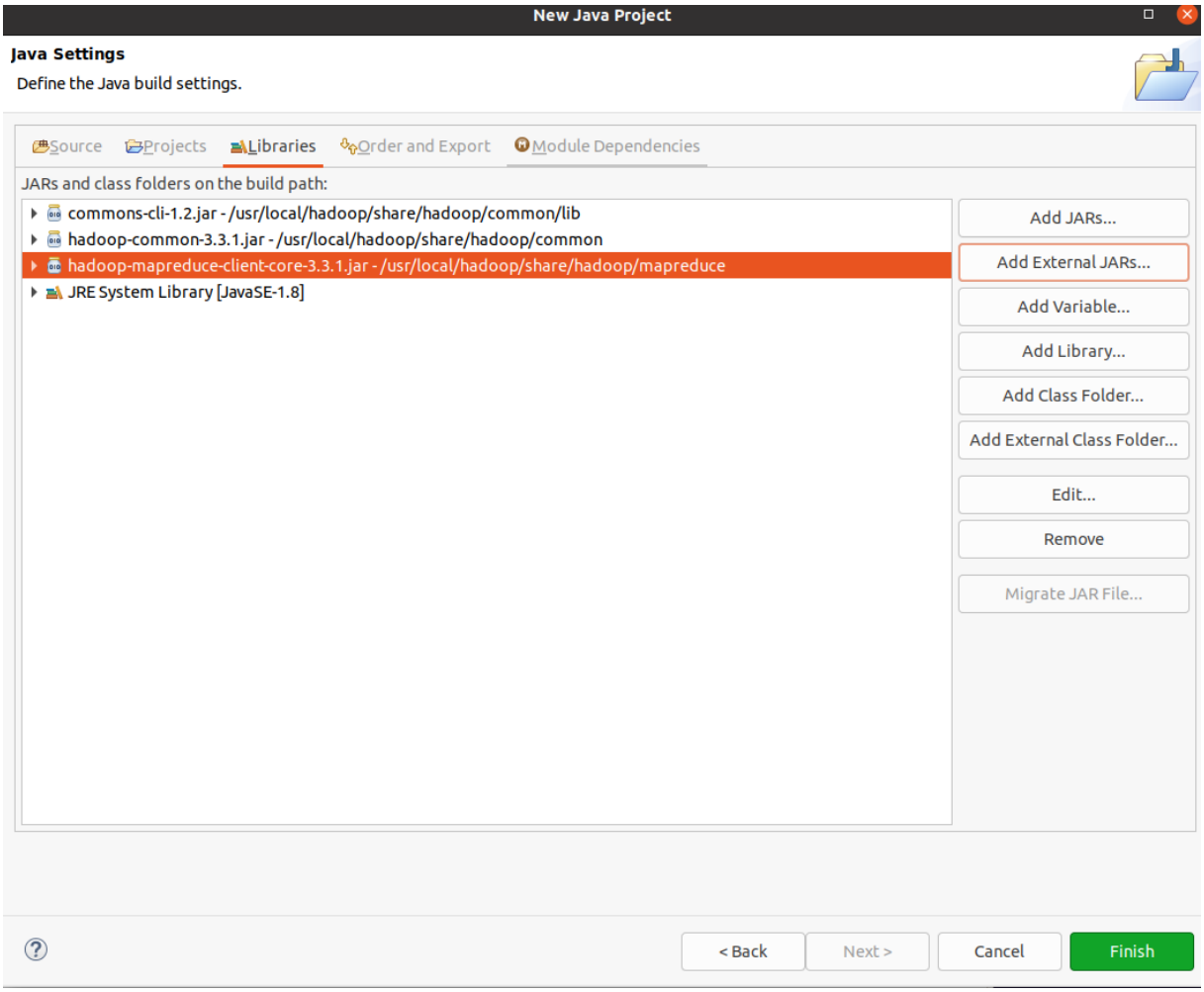
@eclipse:

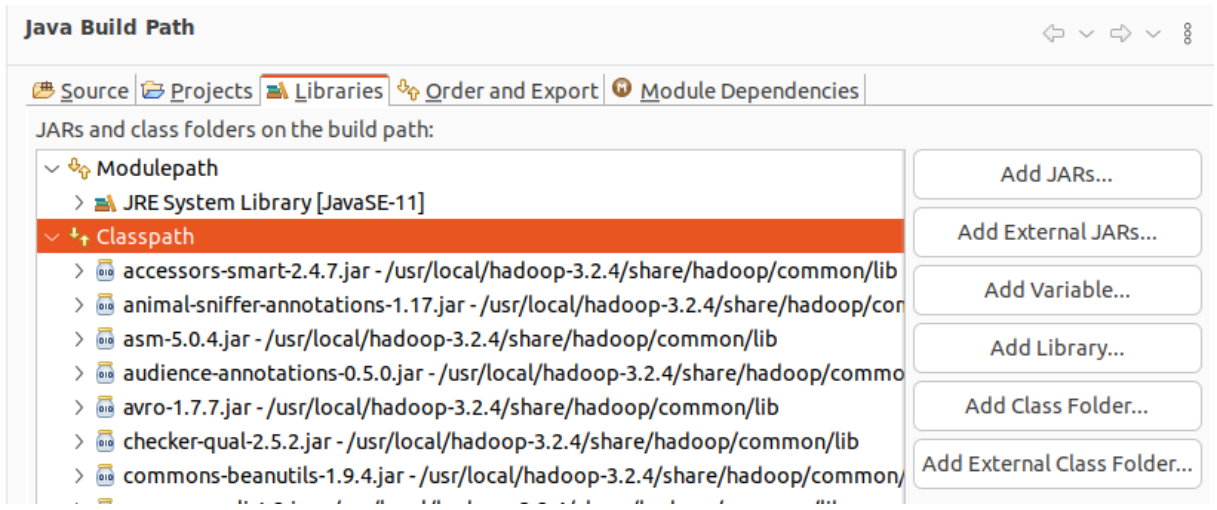


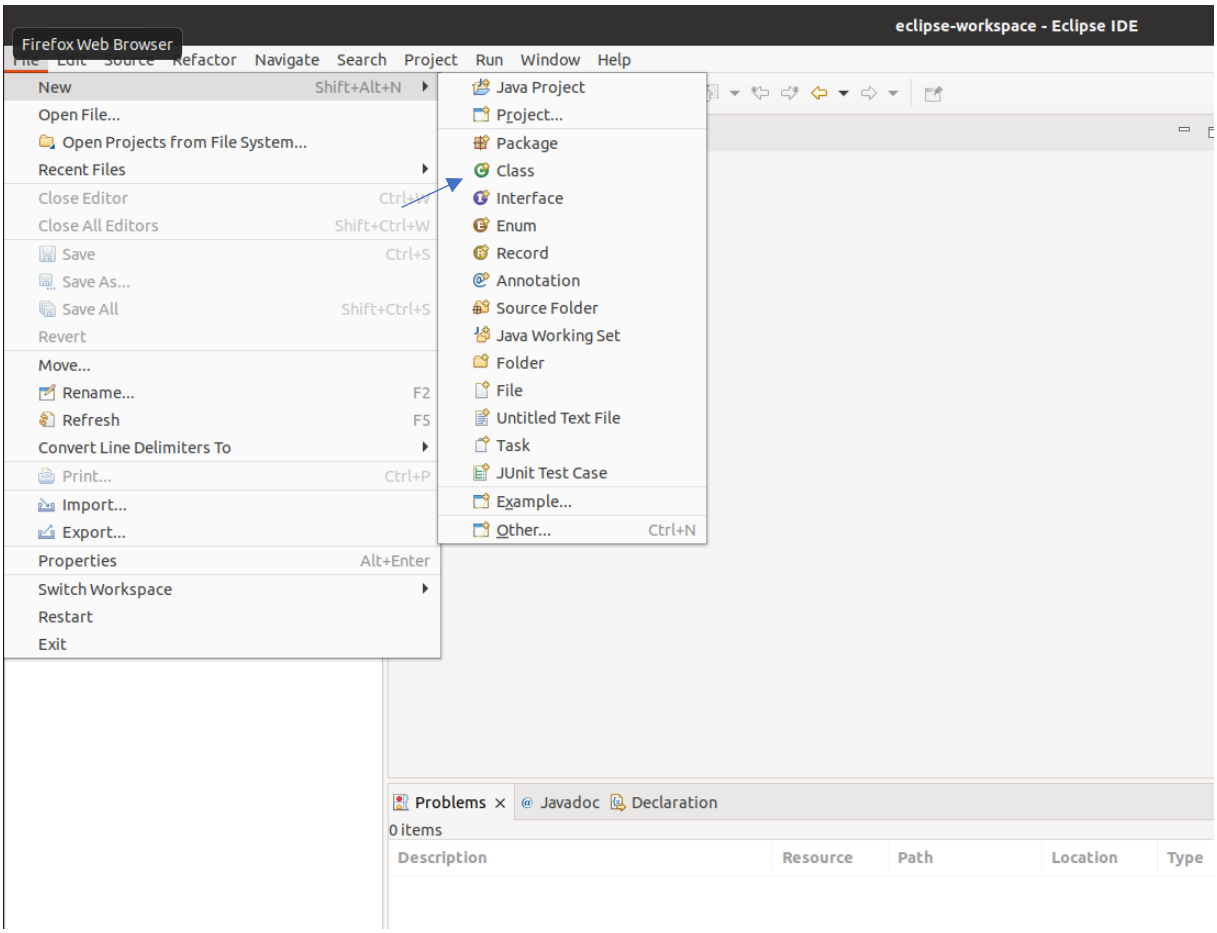
Wenn eine Java-Version > 11 verwendet wird, dann muss in Kompatibilität Version 11 gewählt werden.

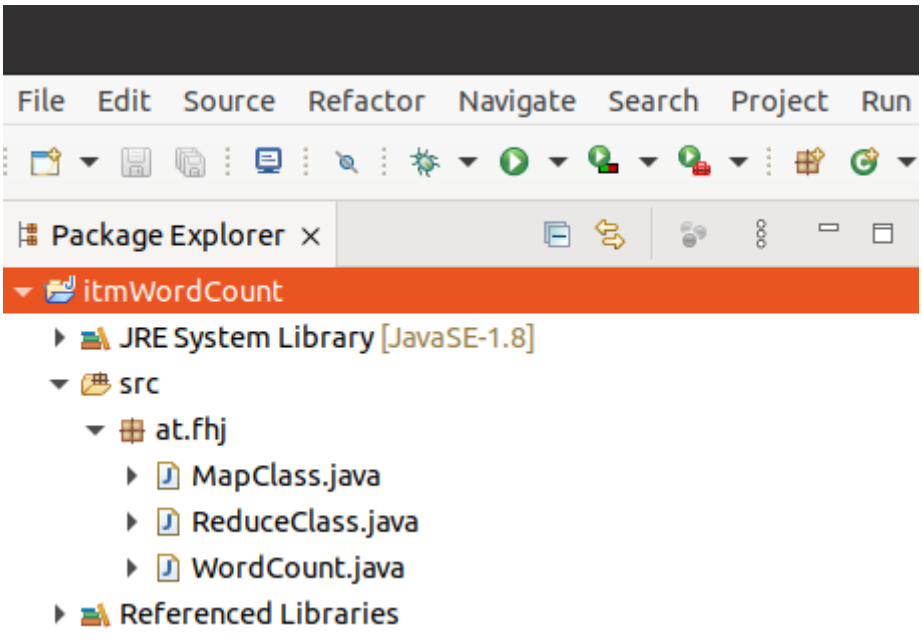


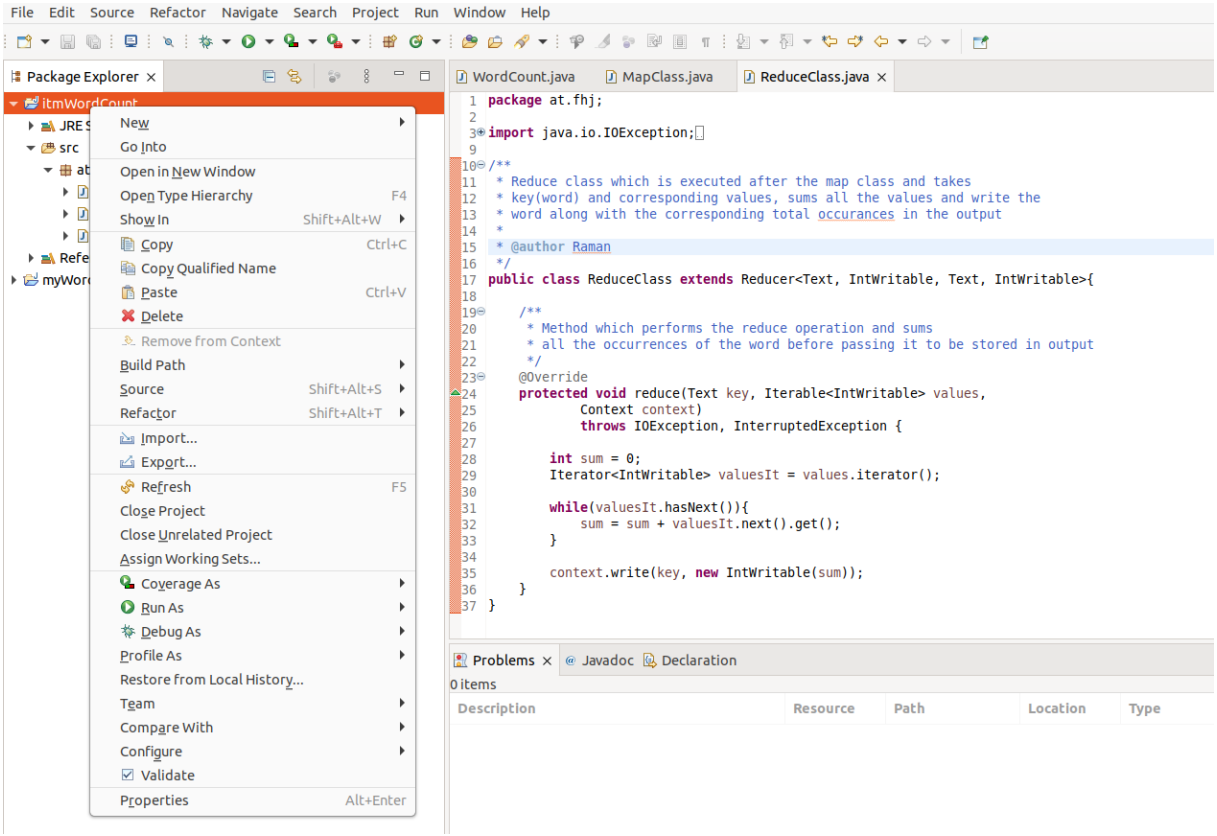
Achtung: in neueren Eclipse-Versionen muss man die einzelnen Libraries unter "Classpath" dazuhängen, davor sind die Buttons ausgegraut.

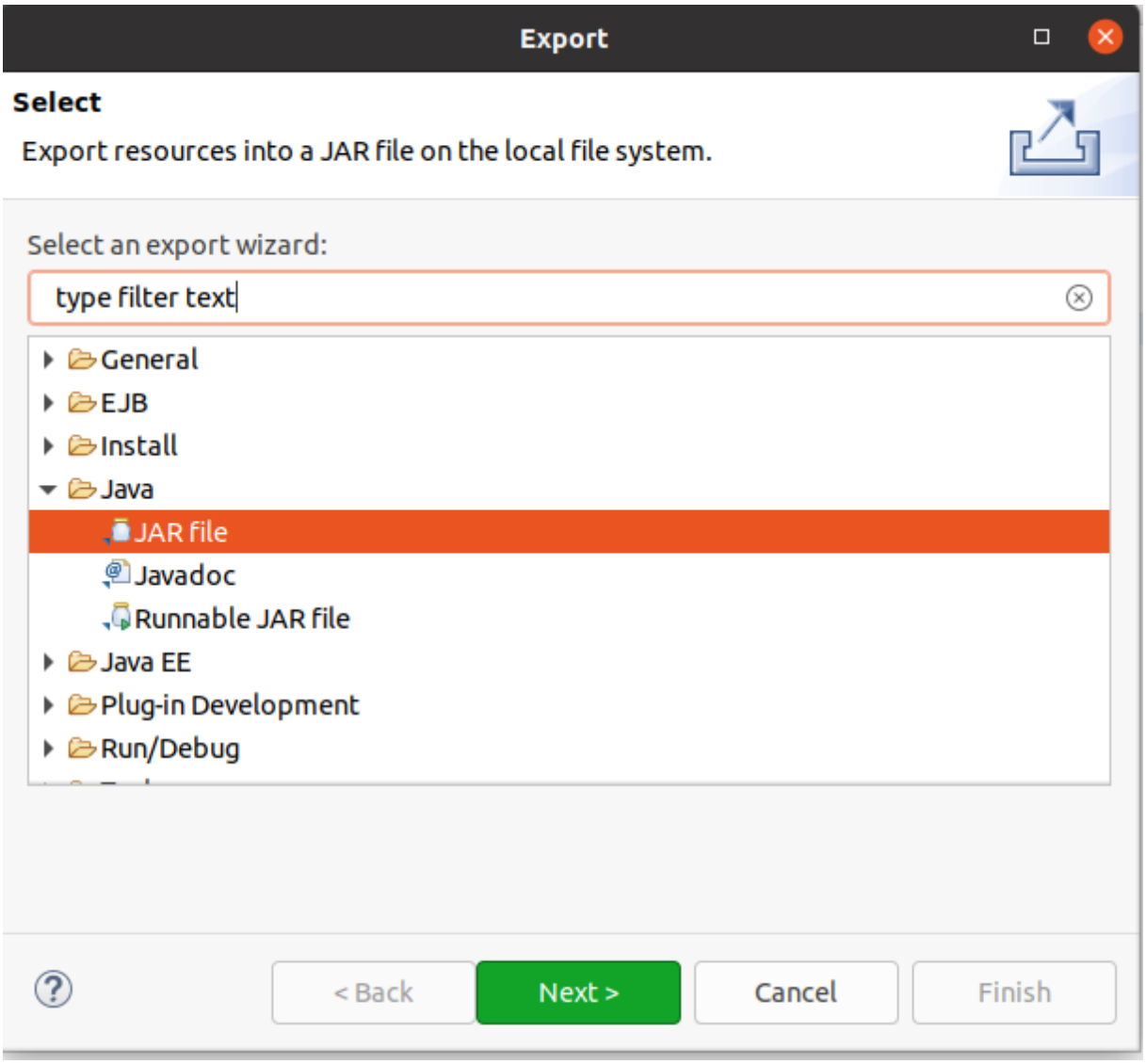

New Java Project
Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
Module Dependencies
JARs and class folders on the build path:
commons-cli-1.2.jar - /usr/local/hadoop/share/hadoop/common/lib
hadoop-common-3.3.1.jar - /usr/local/hadoop/share/hadoop/common
hadoop-mapreduce-client-core-3.3.1.jar - /usr/local/hadoop/share/hadoop/mapreduce
JRE System Library [JavaSE-1.8]
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
< Back
Next >
Cancel
Finish


Java Build Path
Source
Projects
Libraries
Order and Export
Module Dependencies
JARs and class folders on the build path:
Modulepath
JRE System Library [JavaSE-11]
Classpath
accessors-smart-2.4.7.jar - /usr/local/hadoop-3.2.4/share/hadoop/common/lib
animal-sniffer-annotations-1.17.jar - /usr/local/hadoop-3.2.4/share/hadoop/cor
asm-5.0.4.jar - /usr/local/hadoop-3.2.4/share/hadoop/common/lib
audience-annotations-0.5.0.jar - /usr/local/hadoop-3.2.4/share/hadoop/commo
avro-1.7.7.jar - /usr/local/hadoop-3.2.4/share/hadoop/common/lib
checker-qual-2.5.2.jar - /usr/local/hadoop-3.2.4/share/hadoop/common/lib
commons-beanutils-1.9.4.jar - /usr/local/hadoop-3.2.4/share/hadoop/common/
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...

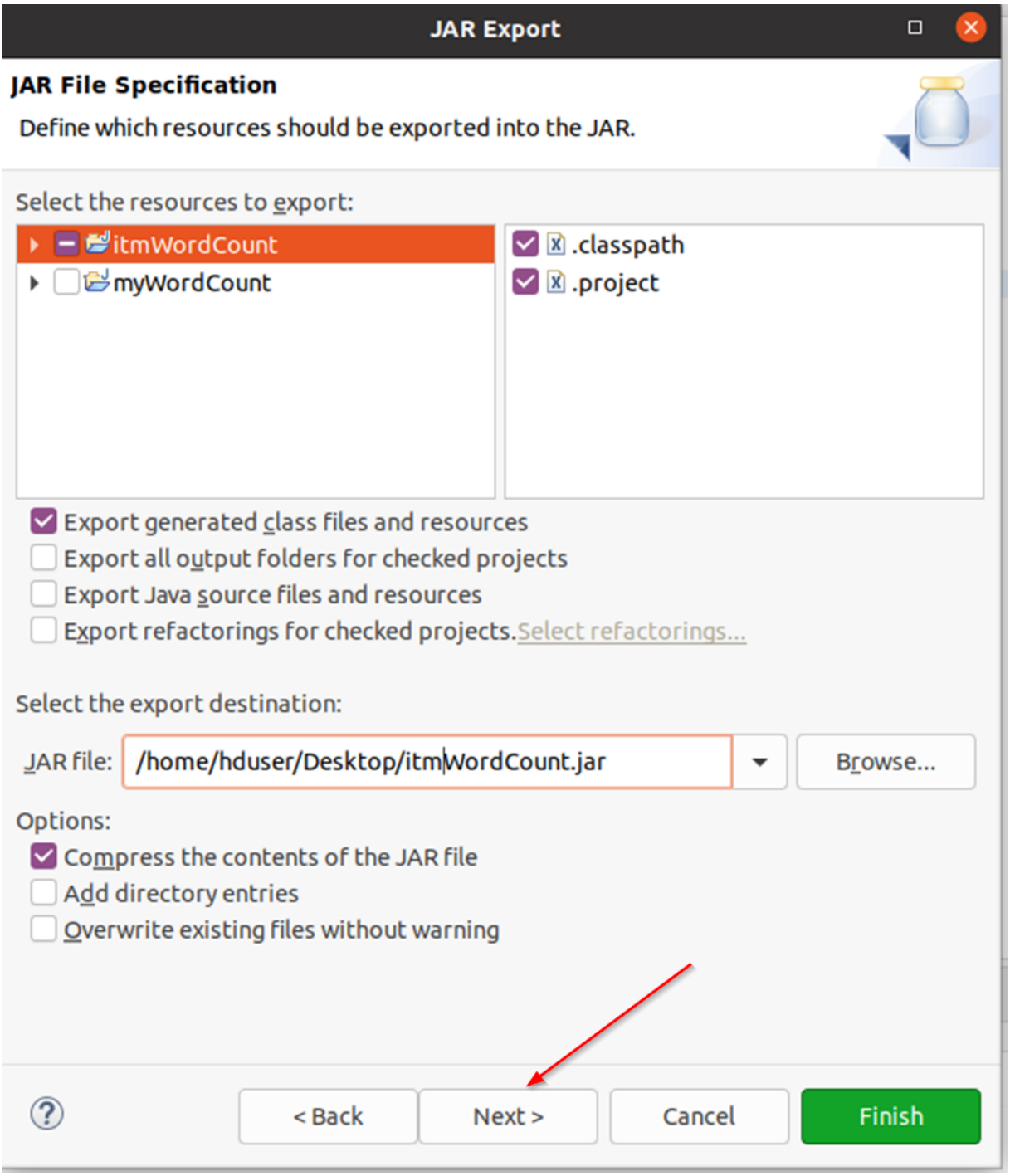

Firefox Web Browser
eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
New Shift+Alt+N
Open File...
Open Projects from File System...
Recent Files
Close Editor Ctrl+W
Close All Editors Shift+Ctrl+W
Save Ctrl+S
Save As...
Save All Shift+Ctrl+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Import...
Export...
Properties Alt+Enter
Switch Workspace
Restart
Exit
Java Project
Project...
Package
Class
Interface
Enum
Record
Annotation
Source Folder
Java Working Set
Folder
File
Untitled Text File
Task
JUnit Test Case
Example...
Other... Ctrl+N
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type

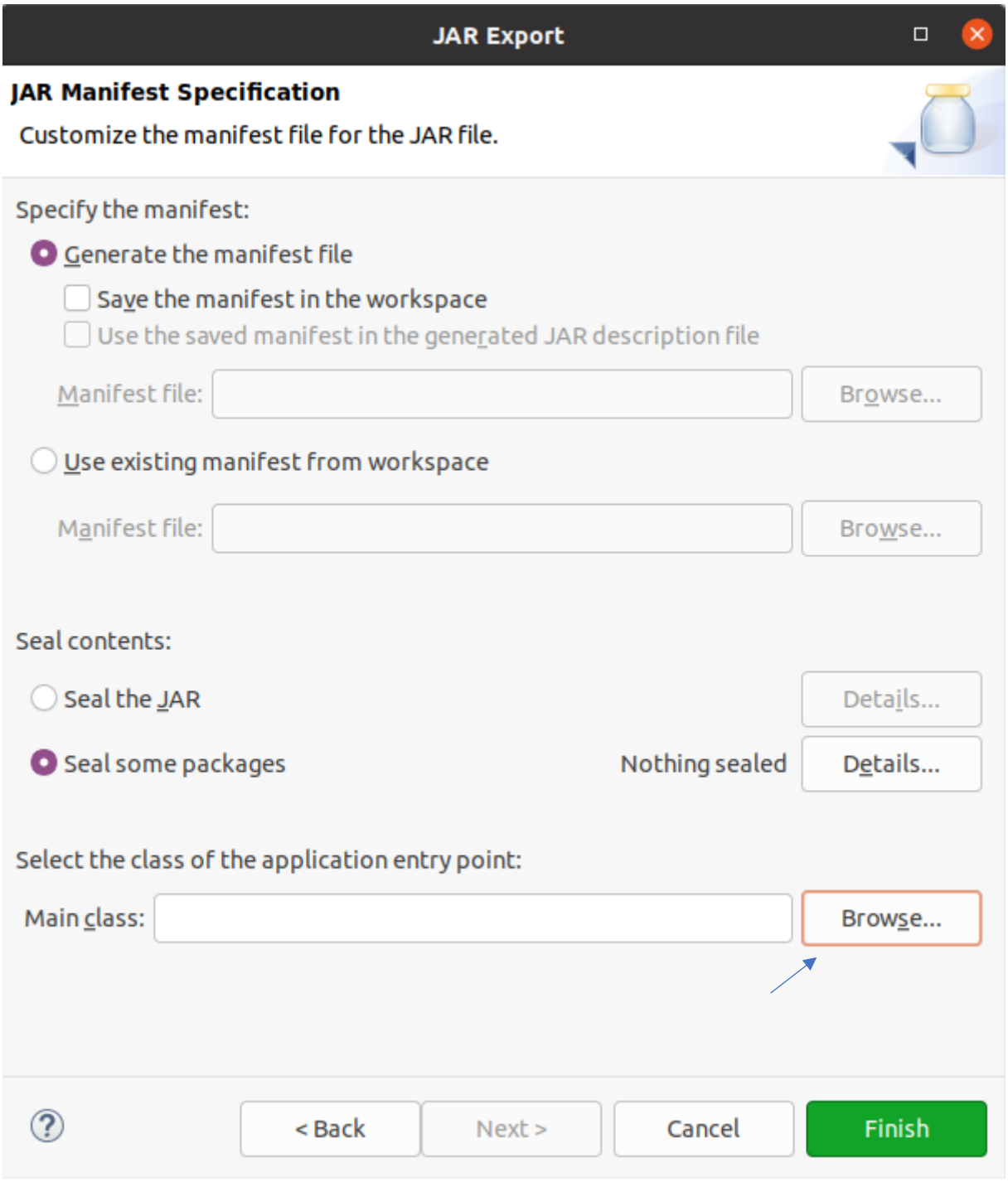

File Edit Source Refactor Navigate Search Project Run
Package Explorer
itmWordCount
JRE System Library [JavaSE-1.8]
src
at.fhj
MapClass.java
ReduceClass.java
WordCount.java
Referenced Libraries

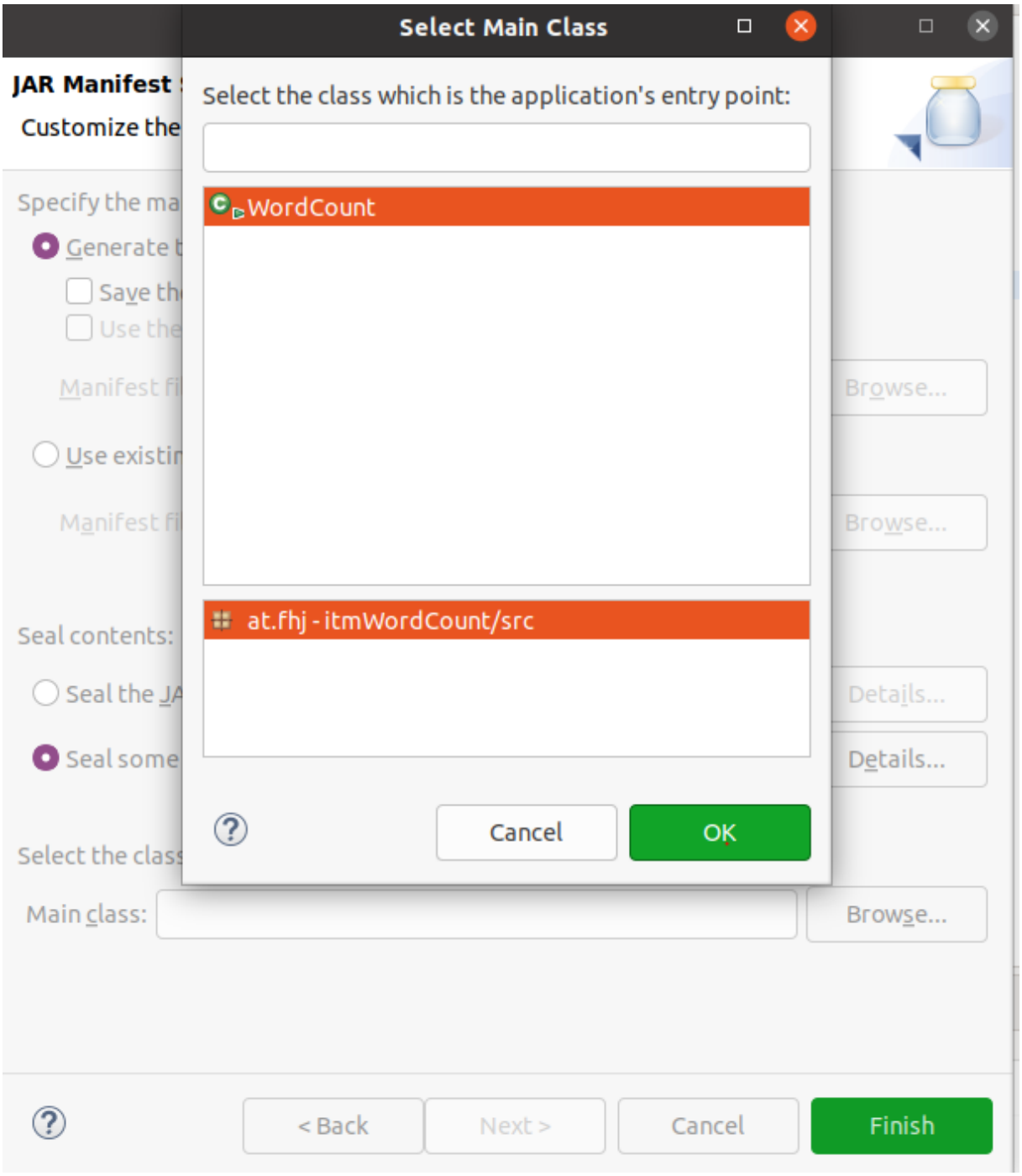

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
itmWordCount
JRE S
src
at
Refe
myWor
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Shift+Alt+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete
Remove from Context
Build Path
Source Shift+Alt+S
Refactor Shift+Alt+T
Import...
Export...
Refresh F5
Close Project
Close Unrelated Project
Assign Working Sets...
Coverage As
Run As
Debug As
Profile As
Restore from Local History...
Team
Compare With
Configure
Validate
Properties Alt+Enter
WordCount.java
MapClass.java
ReduceClass.java
1 package at.fhj;
2
3 import java.io.IOException;
9
10 /**
11 * Reduce class which is executed after the map class and takes
12 * key(word) and corresponding values, sums all the values and write the
13 * word along with the corresponding total occurances in the output
14 *
15 * @author Raman
16 */
17 public class ReduceClass extends Reducer<Text, IntWritable, Text, IntWritable>{
18
19 /**
20 * Method which performs the reduce operation and sums
21 * all the occurrences of the word before passing it to be stored in output
22 */
23 @Override
24 protected void reduce(Text key, Iterable<IntWritable> values,
25 Context context)
26 throws IOException, InterruptedException {
27
28 int sum = 0;
29 Iterator<IntWritable> valuesIt = values.iterator();
30
31 while(valuesIt.hasNext()){
32 sum = sum + valuesIt.next().get();
33 }
34
35 context.write(key, new IntWritable(sum));
36 }
37 }
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type



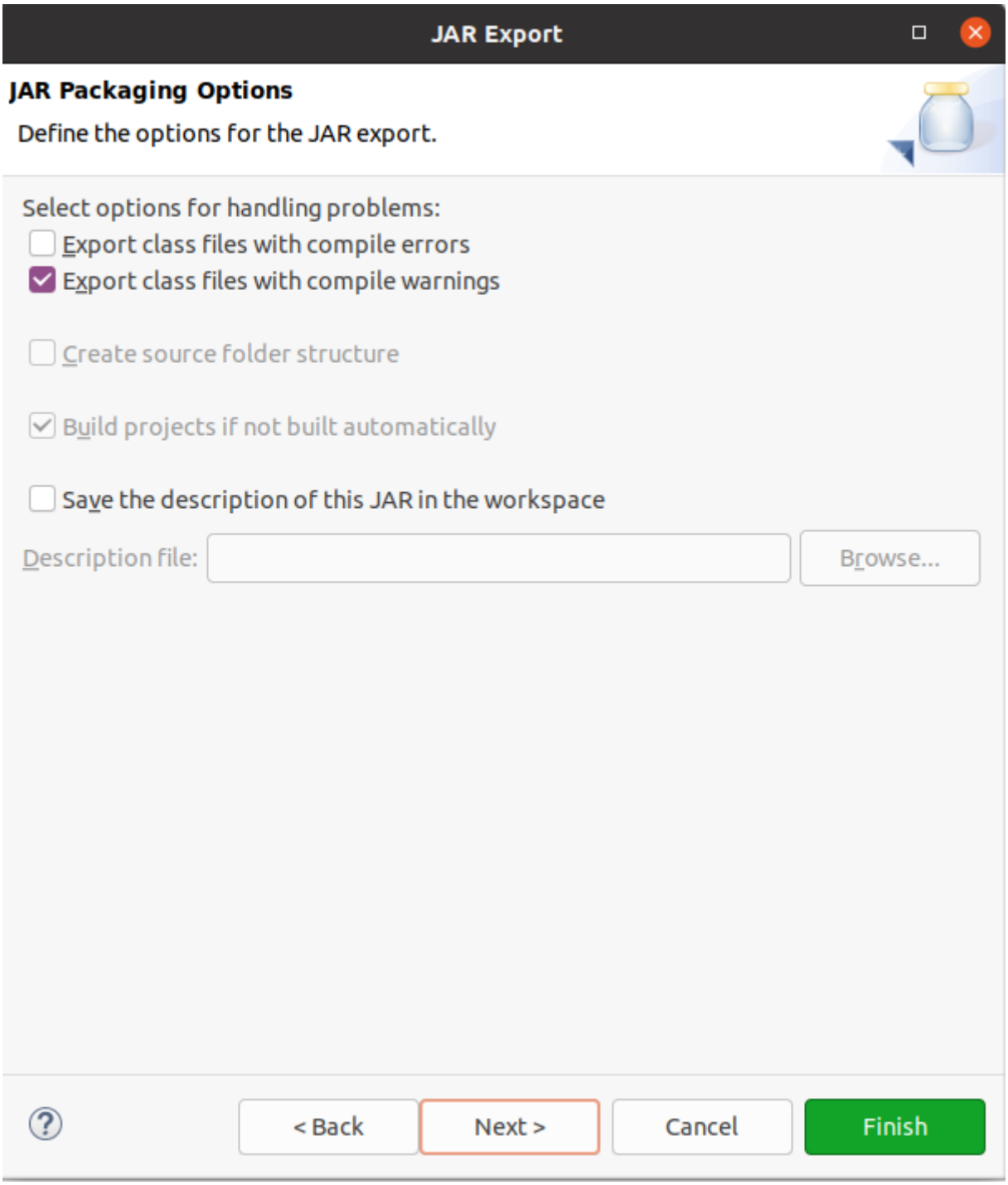
Wichtig: Im folgenden Fenster nicht "Finish", sondern "Next" klicken, damit die weiteren Detaileingaben zu Manifest und v.a. Einsprungspunkt gesetzt werden können.


JAR Export
JAR File Specification
Define which resources should be exported into the JAR.
Select the resources to export:
itmWordCount
myWordCount
.classpath
.project
Export generated class files and resources
Export all output folders for checked projects
Export Java source files and resources
Export refactorings for checked projects. Select refactorings...
Select the export destination:
JAR file:
/home/hduser/Desktop/itmWordCount.jar
Browse...
Options:
Compress the contents of the JAR file
Add directory entries
Overwrite existing files without warning
< Back
Next >
Cancel
Finish


JAR Export
JAR Manifest Specification
Customize the manifest file for the JAR file.
Specify the manifest:
Generate the manifest file
Save the manifest in the workspace
Use the saved manifest in the generated JAR description file
Manifest file:
Browse...
Use existing manifest from workspace
Manifest file:
Browse...
Seal contents:
Seal the JAR
Details...
Seal some packages
Nothing sealed
Details...
Select the class of the application entry point:
Main class:
Browse...
< Back
Next >
Cancel
Finish



Das Hakerl bei "Export class file with compile errors" sollte man entfernen, sonst gibt's erst Fehler zur Laufzeit.



@gradle:

1. Projekt als „Existing Gradle Project“ importieren

1. Menü File → Import….
2. Im Dialog:
   - Kategorie Gradle aufklappen.
   - Eintrag Existing Gradle Project wählen.
   - Next.

3. Bei „Project root directory“ den Ordner auswählen, in dem die build.gradle für das WordCount-Projekt liegt.
4. Next → Eclipse liest jetzt das Gradle-Projekt ein.
5. Finish.

Ergebnis:

- Im „Package Explorer“: Das Projekt sollte bereits die richtige Ordnerstruktur haben (`.java`Dateien unter `src/main/java/...`).
  - Wenn dies nicht der Fall, ist, dann einfach einen Source Folder anlegen und die Dateien dort hin verschieben: Rechtsklick auf Projekt → `New` → `Source Folder` → Name: `src/main/java`.
  - Damit die Einstellungen übernommen werden: Rechtsklick auf das Projekt → `Gradle` → `Refresh Gradle Project`.
- Die Hadoop-Bibliotheken werden automatisch über die `dependencies` in der `build.gradle` übernommen – man muss sie nicht manuell als External JARs hinzufügen.

2. Projekt über den Gradle Tasks View bauen und Jar erstellen

1. Menü `Window` → `Show View` → `Other…`.
2. Kategorie `Gradle` → `Gradle Tasks`.
3. In der „Gradle Tasks“-Ansicht das Projekt auswählen.
4. Unter der Gruppe `build` entweder `build` oder `jar` wählen.

Standardmäßig erzeugt Gradle das Jar unter dem Pfad `build/libs/<projektname>-<version>.jar`. In diesem Fall unter `build/libs/Hadoopwordcount-1.0.jar`

6. Prüfung des erstellten Artifacts und Kopieren auf Zielplattform

Je nach verwendetem Buildsystem und deren Settings findet man das generierte Jar-File unter z.B. out/artifacts oder build/libs und beinhaltet der Dateiname eine Versionsnr. oder nicht.

Das build.gradle beinhaltet einen „deploy“ Task, dieser funktioniert jedoch nur bei Passwort-based login bzw. müsste man die Credentials und Hostnamen ändern.

Daher am besten händisch die Datei manuell kopieren als user „student“ und dann die Datei dem hduser „schenken“:

```
scp -I <PrivateKeyFile> Hadoopwordcount-1.0.jar
student@<hostname>:/tmp/Hadoopwordcount-1.0.jar
ssh student@<hostname>
sudo chown hduser:hadoop /tmp/Hadoopwordcount-1.0.jar
su – hduser
mv /tmp/Hadoopwordcount-1.0.jar ~/Hadoopwordcount.jar
```

7. Test des generierten jar-Files

Hadoop starten und Test aufrufen mit dem zuvor generierten und auf die Zielplattform kopierten Jar-File.

Job ausführen

1.) Input Ordner und Beispiel Textdatei im hdfs anlegen
2.) Job ausführen (Vorsicht Pfade sind ggf. anzupassen!)

```
su - hduser
start-dfs.sh
start-yarn.sh
hdfs dfs -mkdir /input
hdfs dfs -mkdir /output
hdfs dfs -put ~/BigData/data/Bibel.txt /input/
OutputDir=/output/Bibel
# folgendes für jeden Versuch neu ausführen
hdfs dfs -rm -R $OutputDir
hadoop jar Hadoopwordcount.jar /input/Bibel.txt $OutputDir
```

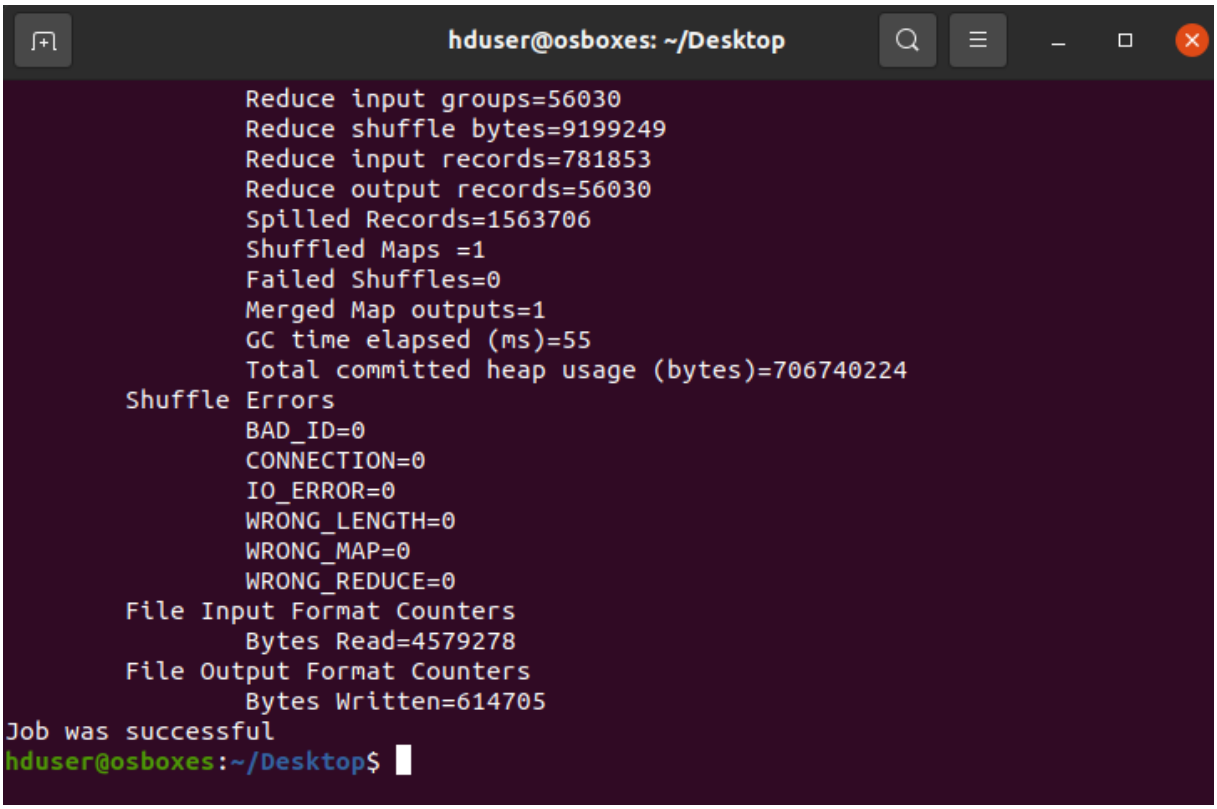
Bei Problemen (vor allem wenn Meldung „main class not found“) die Jar-Datei überprüfen ("jar -tvf" funktioniert meines Wissens nur unter Linux), sie muss zumindest folgende 4 Dateien beinhalten:

```
jar -tvf Hadoopwordcount.jar | awk '{ print $8 }'
META-INF/MANIFEST.MF
at/fhj/MapClass.class
at/fhj/ReduceClass.class
at/fhj/WordCount.class
```

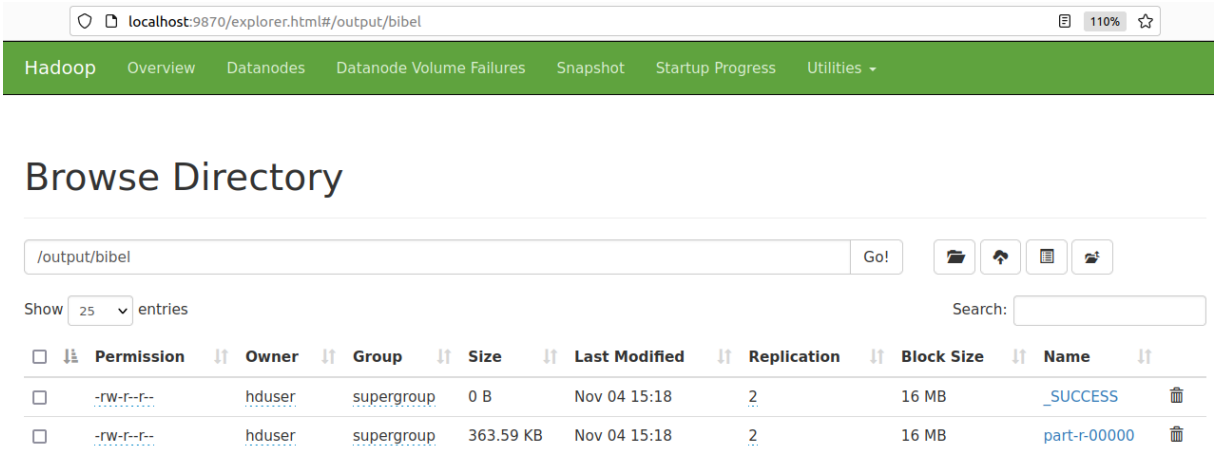
Der Inhalt der Datei MANIFEST.MF sollte wie folgt sein (optional stehen dahinter noch ClassPaths):

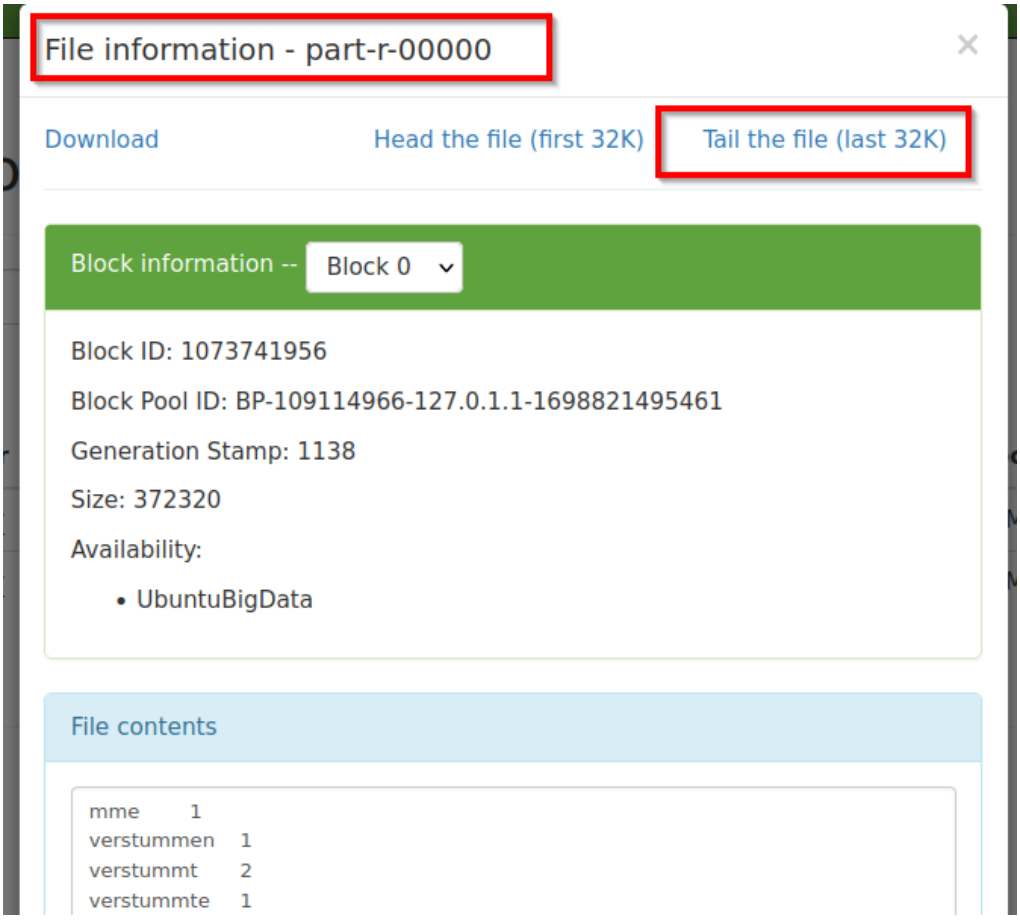
```
jar -xf Hadoopwordcount.jar META-INF/MANIFEST.MF
cat META-INF/MANIFEST.MF
Manifest-Version: 1.0
Main-Class: at.fhj.WordCount
```

Erwarteter Output nach erfolgreichem MapReduce Job wie folgt:



Im angegebenen Output-Verzeichnis befinden sich dann 2 Dateien, eine leere Datei "_SUCCESS" und eine Datei mit dem Ergebnis des Jobs (wäre der Output größer als konfigurierte Blockgröße würde es weitere Dateien part-r-00001 usw. geben).

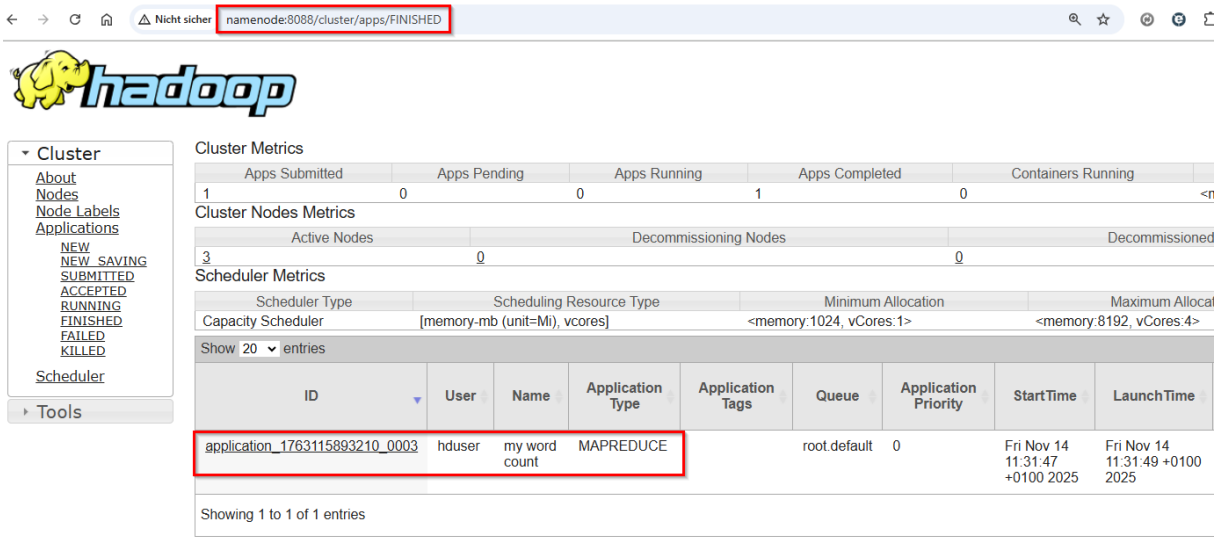




Das “Tail the file” funktioniert üblicherweise nicht über die GUI, daher besser über Kommandozeile:

```
hdfs dfs -tail /output/Bibel/part-r-00000
hdfs dfs -head /output/Bibel/part-r-00000
```

Auf der Webseite von Yarn sieht man die erledigten Jobs ebenfalls (zu sehen nur im Fall, wenn mapreduce.framework.name auf „yarn“ und nicht „local“ eingestellt wurde):



Wichtig: wenn der Job erneut gestartet werden soll, muss zuvor das Output-Verzeichnis gelöscht werden (sowohl wenn man Output ins lokale Dateisystem als auch wenn man dies in hdfs schreibt)!