Tutorium SS17

EINFÜHRUNG + BREAKOUT

LUKAS LEIPOLD L.LEIPOLD@CAMPUS.LMU.DE

EVI BERCHTOLD BERCHTOLD@BIO.IFI.LMU.DE

Überblick

- Homepage: https://www.bio.ifi.lmu.de/studium/ss2017/prk_prop/index.html
- 1. Projekt: Breakout
  - Erlernen von einfachen GUI Komponenten
  - Planung und Umsetzung eines Projekts
  - Java nicht vergessen

- Die anderen Projekte:
  - Vorbereitung auf das ProPra
  - Spaß haben, neues Lernen

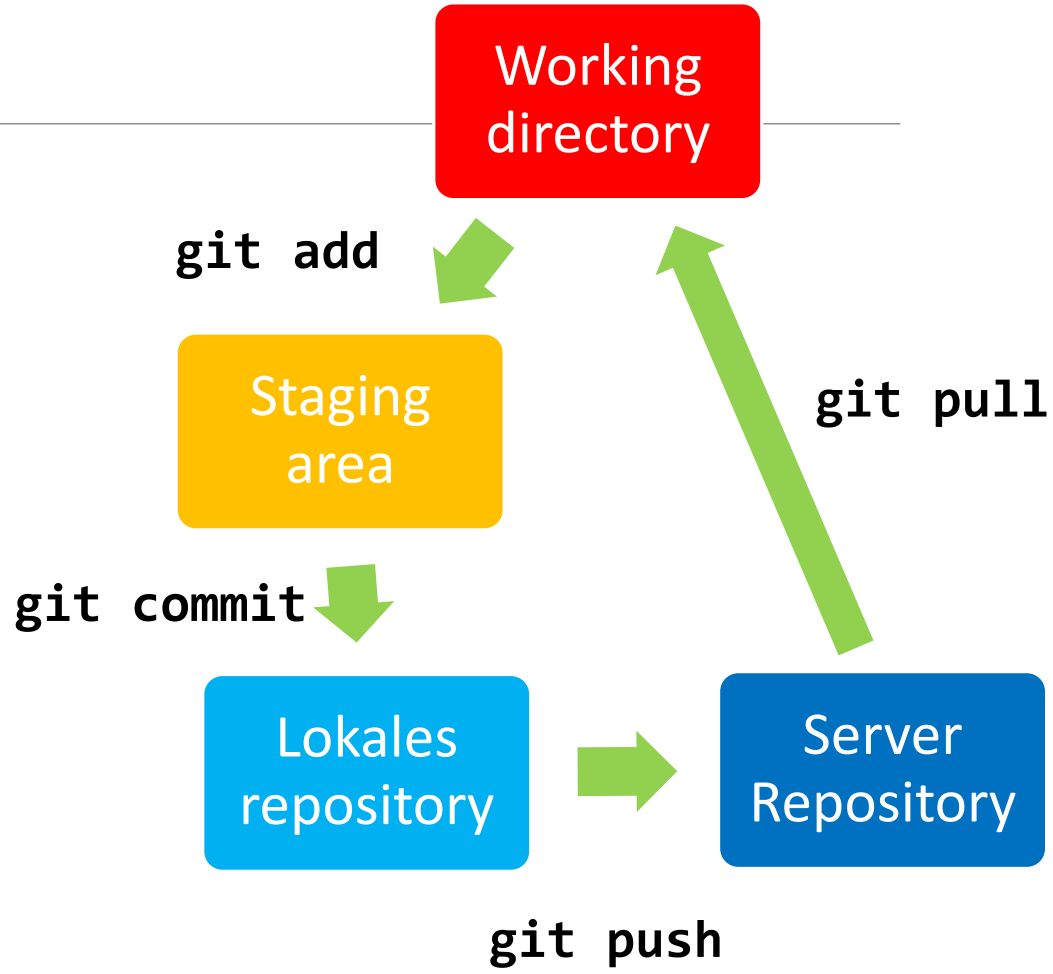
- Unterschiedliche Versionen einer Dateien (z.B. java file) effizient speichern
- Änderungen überwachen bzw. rückgängig machen
- Repository anderen zur Verfügung stellen (z.B. GitHub, GitLab)
- Lokale Kopie des Server Repositoriy‘s
  - Repository Updates einfügen `git pull`
  - Lokale Updates hochladen `git push`
- Arbeiten mehrerer Programmierer an einem Projekt

Git – How To

- Server Repository anlegen
  - Wir: https://gitlab.cip.ifi.lmu.de/users/sign_in
    (Vorher bei RBG freischalten, dann <benutzer>@cip.ifi.lmu.de Email-Adresse abrufen)
  - Alternativ: `git init [--bare]`
    - `--bare` : Wenn man sein Projekt mit anderen teilen möchte, oder auf unterschiedlichen Rechnern arbeitet
    - http://www.saintsjd.com/2011/01/what-is-a-bare-git-repository/
- Nach dem Anlegen (jetzt lokal):
  - Zu `~/git` wechseln
  - `git clone <my_remote_repository>` (von gitLab)
    - Der neu erstellte Ordner ist nun das „working directory“

Git - Dateikontrolle

- Working directory
  - Ordner der mit git clone erstellt wurde
  - Lokale Änderungen an den Dateien
  - `Git add file1 file2 directory1`
- Staging area: (auch Index genannt)
  - Dateien die unter Versionskontrolle stehen
  - Überwachung von geänderten Dateien (modified)
  - `Git commit –m „<commit message>“`
- Repository:
  - Datenbank zum Speichern der unterschiedlichen Versionen
  - Lokal: `.git` Ordner (nicht damit rumspielen!)



Die Sache mit den Keys

- Zum Kontrollieren wer Zugriff auf ein Server Repository hat
- Meist über ssh keys
  - Habt ihr einen? -> `ls ~/.ssh`
  - Public Key: `id_rsa.pub` (zum authentifizieren auf Servern)
  - Private Key: `id_rsa` (NIEMALS mit anderen teilen)
- Public Keys müssen einem git Server Repository hinzugefügt werden, damit unterschiedliche Benutzer (oder man selbst) Zugriff erhalten.
- Jeder PC kann so einen Public Key haben
- Erstellen mit ssh-keygen (https://git-scm.com/book/en/v1/Git-on-the-Server-Generating-Your-SSH-Public-Key)
- Kopieren der beiden Keys auf seine eigenen Rechner in den ~/.ssh Ordner, kann nützlich sein (Achtung! Überschreiben bzw. Zugriffsverlust möglich)

Das .gitignore file

- Wenn Verzeichnisse bzw. Dateien nicht versioniert werden sollen
- Zum Beispiel : `bin/ ; *.class ; .metadata; .classpath` usw.
- Zu finden:
  - <Git_working_dir>/.gitignore
  - <Git_working_dir>/info/exclude
- Bearbeiten mit Gedit, vi, nano, oder ähnlichem

Merge Konflikte

- Wenn in einer Zeile innerhalb einer Datei unterschiedliche Versionen lokal und serverseitig vorliegen
- Wird durch `git pull` verursacht
  - Deshalb: Immer **erst „pullen“** bevor man lokale Änderungen vornimmt
- Entscheiden, welche Version behalten werden soll
- Mergen mit: Eclipse, git mergetools, vi, emacs oder Ähnlichem

Wichtige Befehle + nützliche Links

- Alles zum Nachlesen unter: https://git-scm.com/book/en/v1/Getting-Started
- Noch mehr Befehle: https://projekte.itmc.tu-dortmund.de/projects/wiki-hsb/wiki/Wichtige_Git_Befehle/1

`git status`	Übersicht über Dateien im working directory
`git add <file1> <dir1>`	Hinzufügen zur Staging Area
`git commit <file1> -m "<commit message>"`	Datei dem Repository (lokal) hinzufügen
`git push`	Änderungen zum Remote-repository (server) hinzufügen
`git pull`	Holen der Änderungen aus dem Remote-Repository
`git rm <file1>`	Löschen einer Datei aus dem Repo, gefolgt von git commit
`git init [--bare]`	Anlegen eines Repositories [auf dem Server]
`git clone <remote_repo>`	Kopieren des Remote-Repositorys

In case of fire

1. git commit -am "untested due to fire"

2. git push -f

3. leave building

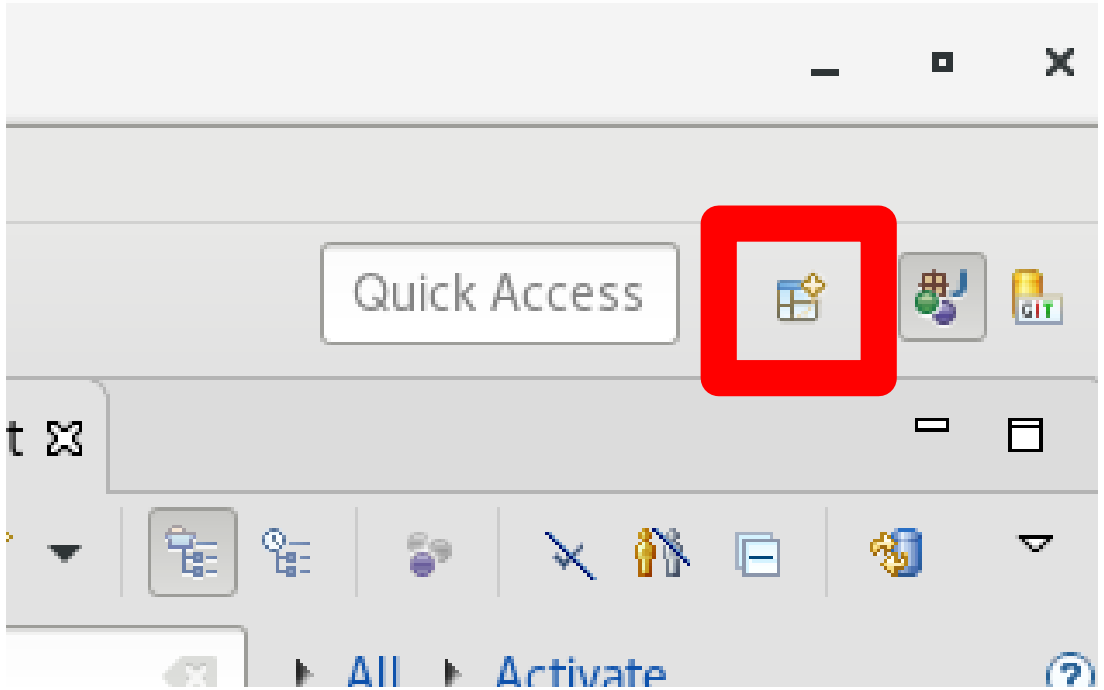
Neuste Eclipse-Version downloaden

- http://www.eclipse.org/downloads/eclipse-packages/
- Dann Eclipse IDE for Java Developers (64-bit)
- In Downloads speichern
- Konsole öffnen
- `mkdir ~/Programme`
- `mv ~/Downloads/eclipse-java-neon-3-linux-gtk-x86.tar.gz ~/Programme`
- `cd Programme`
- `tar –xf eclipse`
- `rm eclipse.tar.gz`

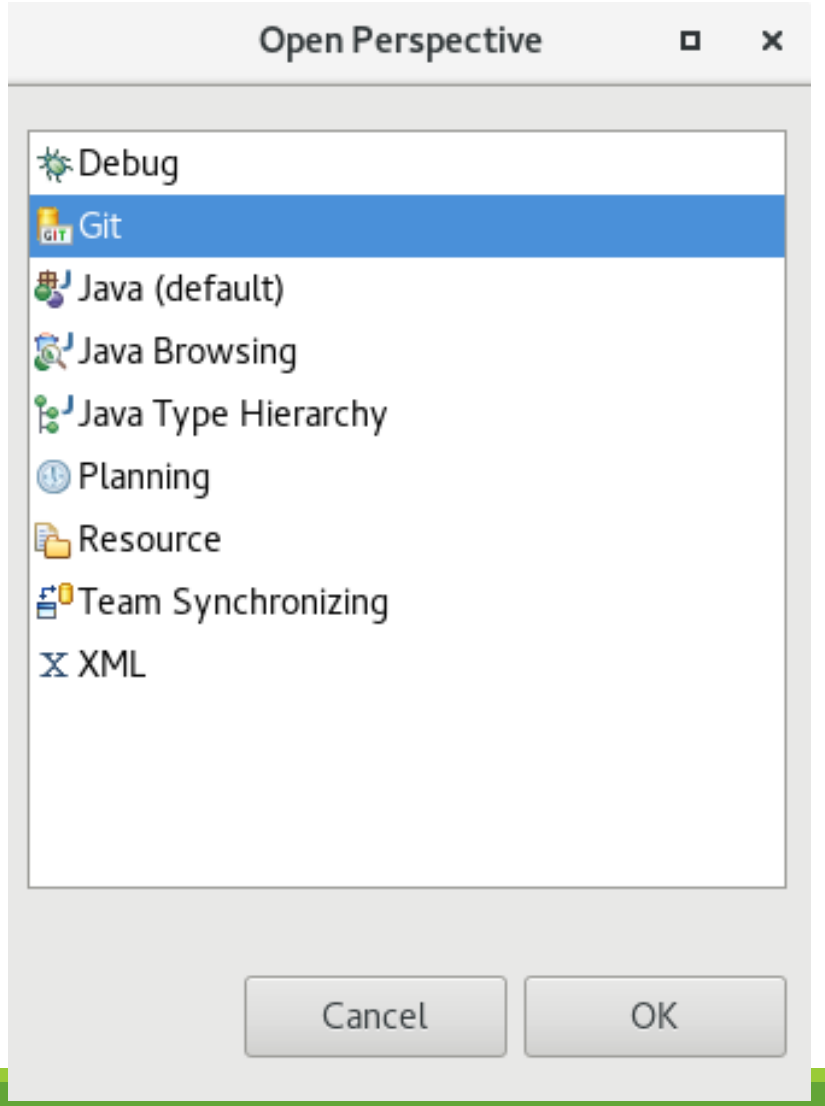
Git einrichten Eclipse

- 1. Repository, aus dem ihr den Code für das Breakout-Spiel holen könnt, hinzufügen
  - Pushen nicht erlaubt!
  - Falls ihr nicht mitschreiben wollt, oder die Musterlösung braucht
- 2. Eigenes Repository hinzufügen (vorher in GitLab angelegt)
  - Eigene Versionskontrolle (auch für spätere Projekte, ProPra)
  - Vertraut machen mit git in Eclipse

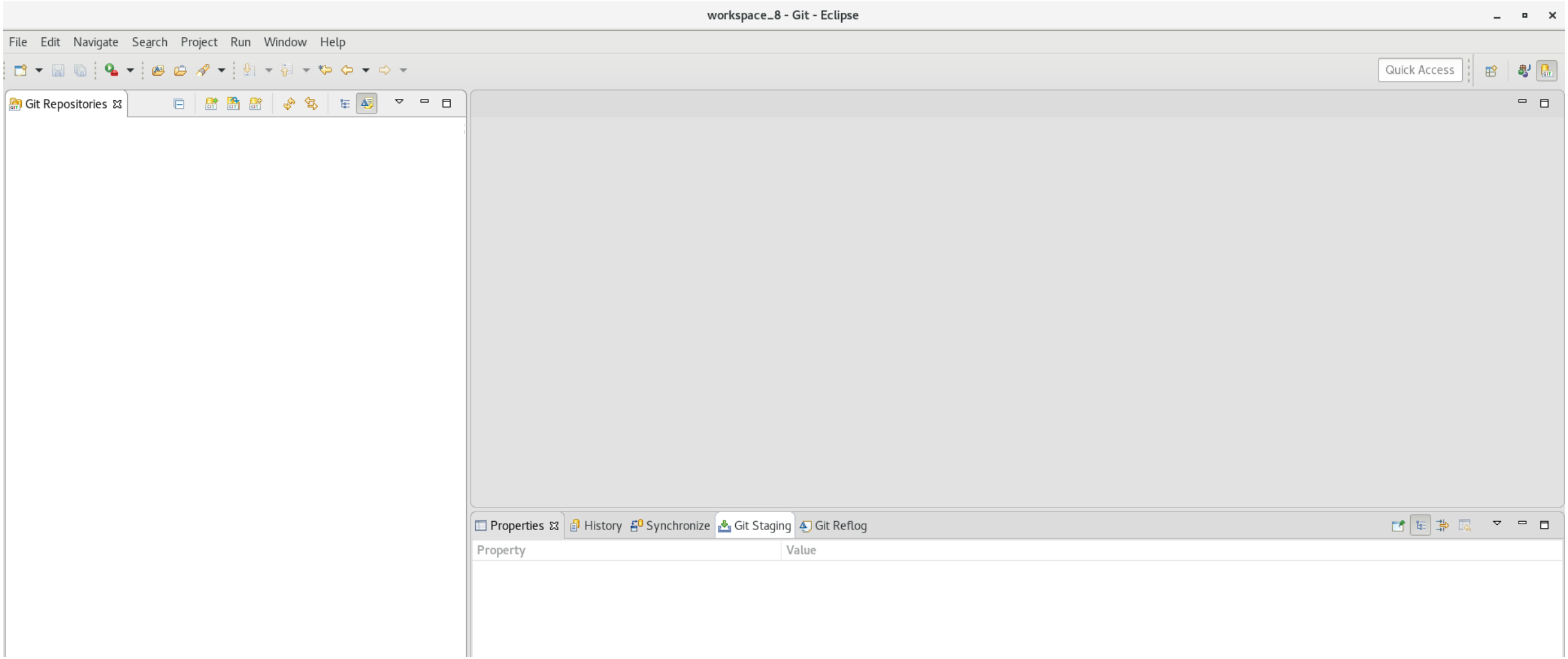
1.


Quick Access
All
Activate

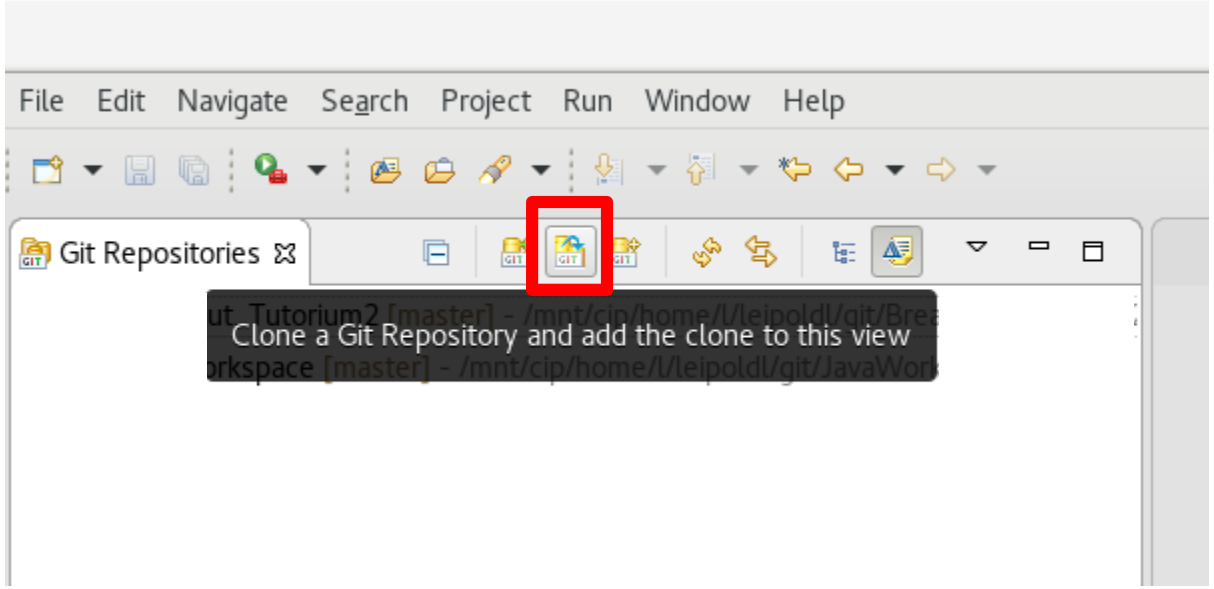
2.


Open Perspective
Debug
Git
Java (default)
Java Browsing
Java Type Hierarchy
Planning
Resource
Team Synchronizing
XML
Cancel
OK

3. Git Perspektive

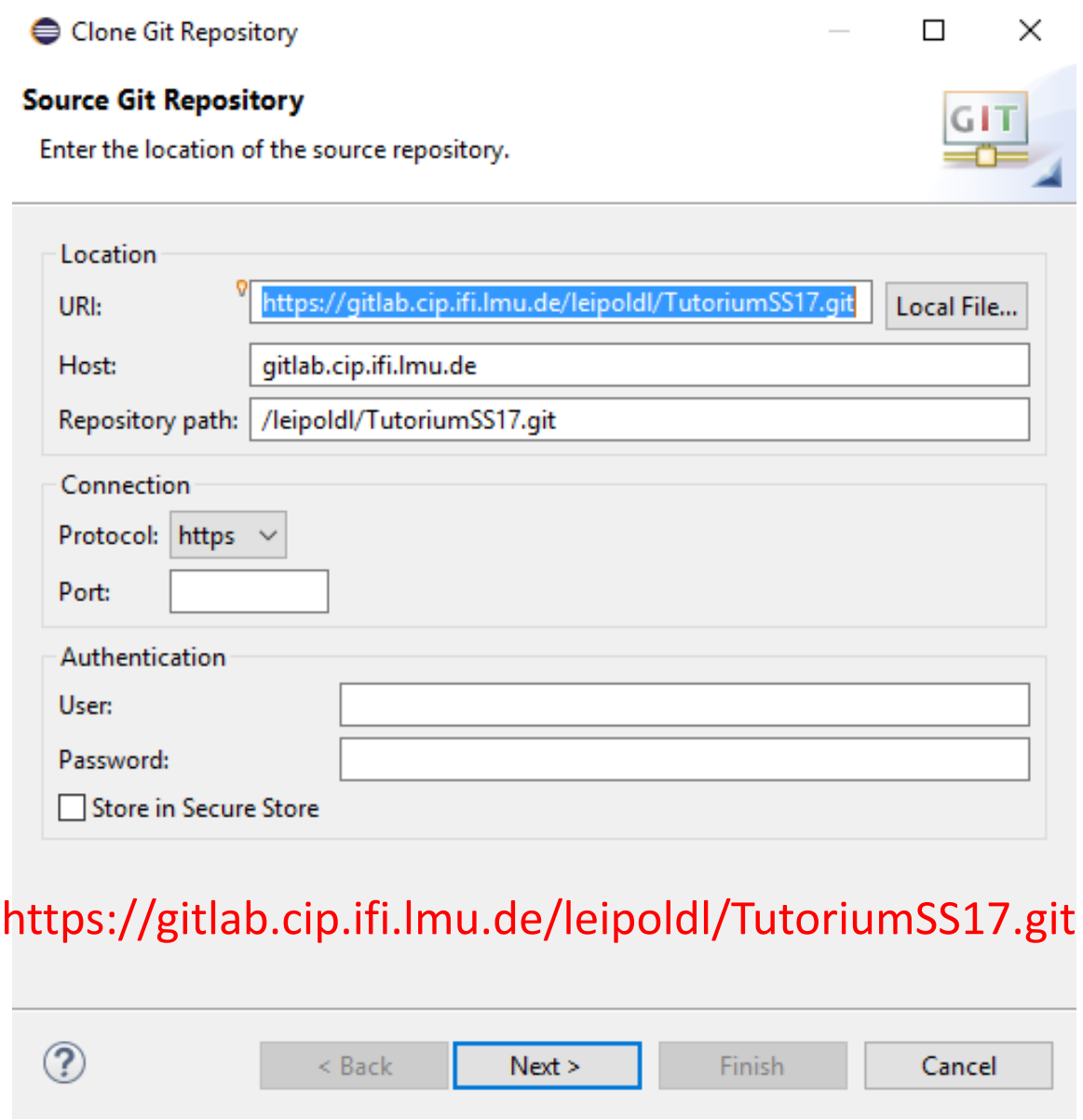


4.



Git Repository klonen

5.



https://gitlab.cip.ifi.lmu.de/leipoldl/TutoriumSS17.git

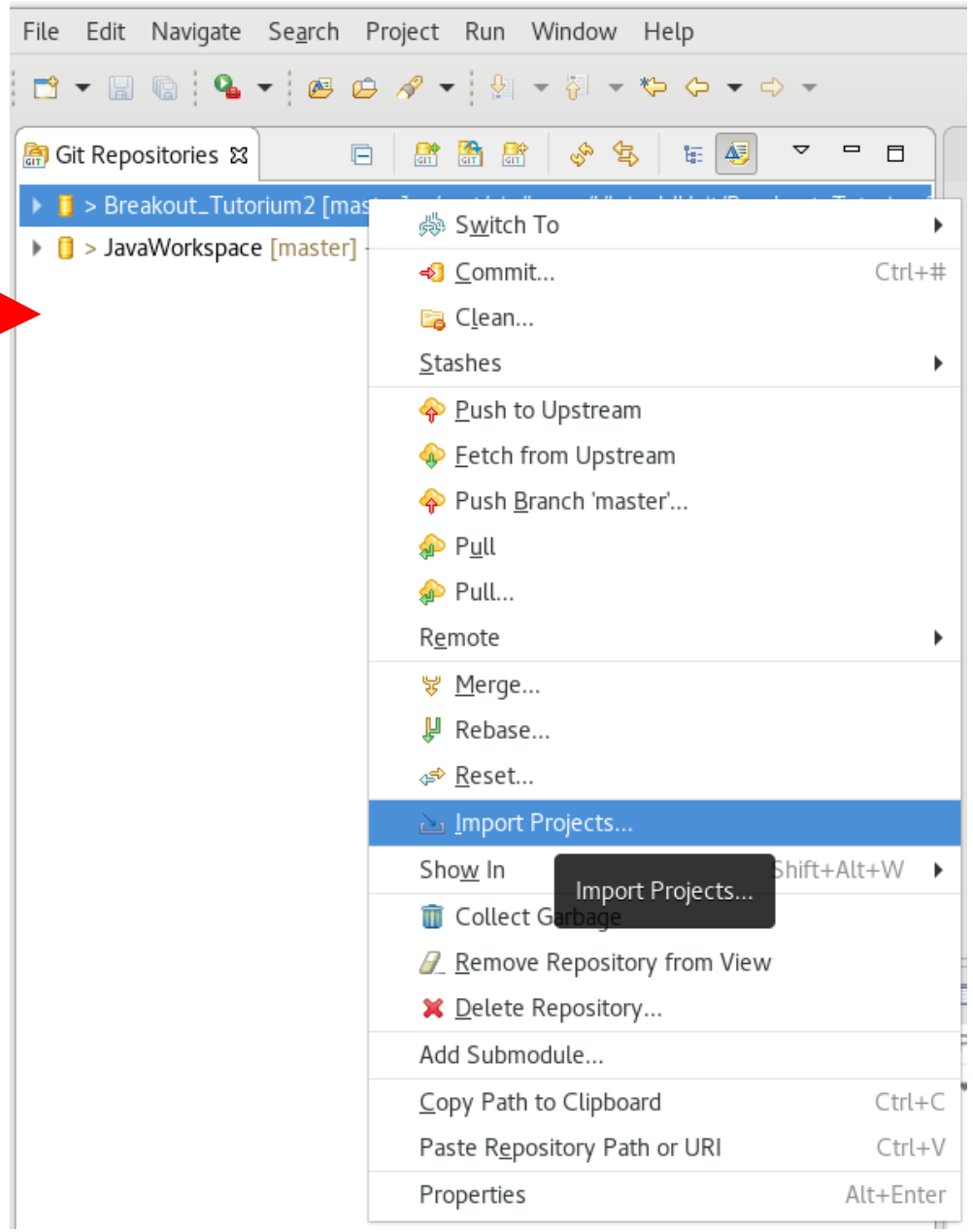
6. Projekt importieren (Rechtsklick auf Git Repository)

7. Das selbe für euer **eigenes** Repository wiederholen

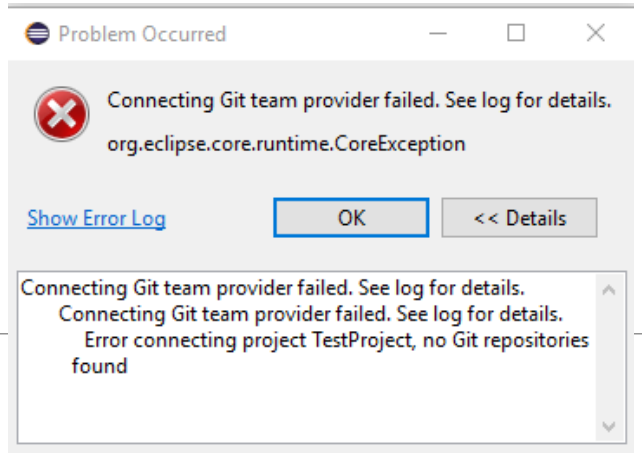
-> Import: New Project Wizard

-> Error: siehe nächste Folie

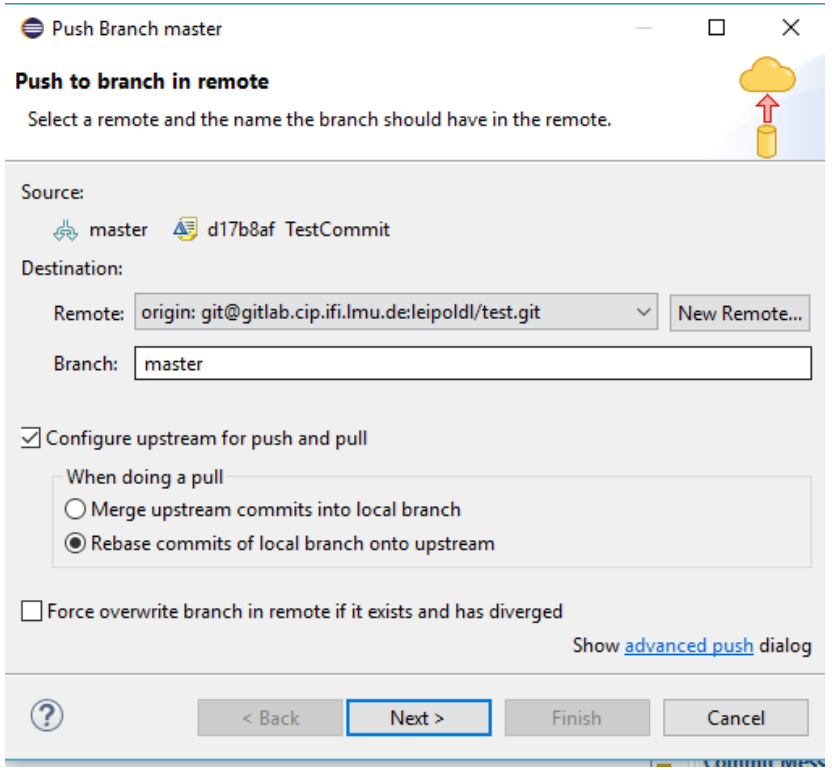
8. Perspektive wieder wechseln (rechts oben)



Error nachdem man das Java Projekt erstellt hat



- Bug in Eclipse.
- Lösung:
  - Nach dem Fehler > OK
  - Dann im Import Fenster > Abbrechen (Java Projekt wurde angelegt!)
  - Projekt mit Git-repository verknüpfen:
    - Rechtsklick auf das erstellte Projekt
    - Team > Share > Geklontes Git Repo auswählen > Ok > Finish
    - Bereit packages Klassen usw. anzulegen!
  - Rechtsklick auf das package > Team > Add to Index
  - Dialog beim Pushen (siehe rechts) > Next > Finish

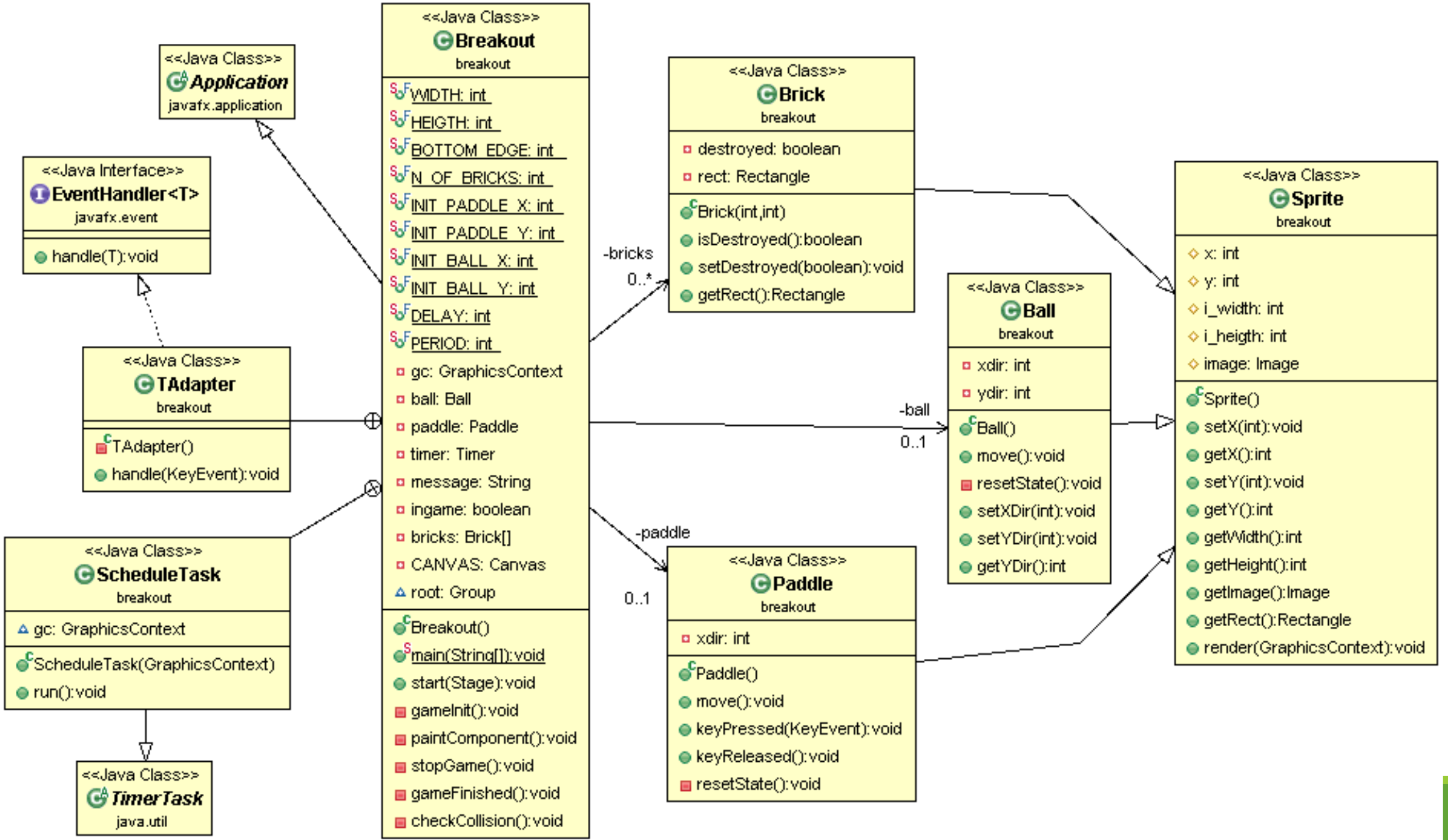


Git in Eclipse

- Rechtsklick auf Projekt > Team > Pull
- Rechtsklick auf Projekt, Package oder Datei, um sie zum Index hinzuzufügen
  - Team > Add to Index
- Rechtsklick auf Projekt > Team > Commit… (Öffnen der Staging Area, Commit message verfassen)
  - Dann „Commit“ oder „Commit and Push“ auswählen
  - Evtl Passphrase eingeben (Passwort des ssh keys)
- Außerdem:
  - Ignore — Eine Datei zur Gitignore Liste hinzufügen
  - Advanced > Untrack — Datei vom Index löschen
  - Synchronize Workspace — Mergekonflikt per Hand beheben

Breakout – Was brauchen wir?

1. Welche möglichen Objekte kann man sehen?
2. Welche kann man nicht sehen?


<<Java Class>>
Application
javafx.application
<<Java Interface>>
EventHandler<T>
javafx.event
handle(T):void
<<Java Class>>
TAdapter
breakout
TAdapter()
handle(KeyEvent):void
<<Java Class>>
ScheduleTask
breakout
gc: GraphicsContext
ScheduleTask(GraphicsContext)
run():void
<<Java Class>>
TimerTask
java.util
<<Java Class>>
Breakout
breakout
WIDTH: int
HEIGTH: int
BOTTOM_EDGE: int
N_OF_BRICKS: int
INIT_PADDLE_X: int
INIT_PADDLE_Y: int
INIT_BALL_X: int
INIT_BALL_Y: int
DELAY: int
PERIOD: int
gc: GraphicsContext
ball: Ball
paddle: Paddle
timer: Timer
message: String
ingame: boolean
bricks: Brick[]
CANVAS: Canvas
root: Group
Breakout()
main(String[]):void
start(Stage):void
gameInit():void
paintComponent():void
stopGame():void
gameFinished():void
checkCollision():void
-bricks
0..*
<<Java Class>>
Brick
breakout
destroyed: boolean
rect: Rectangle
Brick(int,int)
isDestroyed():boolean
setDestroyed(boolean):void
getRect():Rectangle
-ball
0..1
<<Java Class>>
Ball
breakout
xdir: int
ydir: int
Ball()
move():void
resetState():void
setXDir(int):void
setYDir(int):void
getYDir():int
-paddle
0..1
<<Java Class>>
Paddle
breakout
xdir: int
Paddle()
move():void
keyPressed(KeyEvent):void
keyReleased():void
resetState():void
<<Java Class>>
Sprite
breakout
x: int
y: int
i_width: int
i_heigth: int
image: Image
Sprite()
setX(int):void
getX():int
setY(int):void
getY():int
getWidth():int
getHeight():int
getImage():Image
getRect():Rectangle
render(GraphicsContext):void

JavaFX

- Soll AWT und Swing für grafische Benutzeroberflächen (GUI) ersetzen
- Java 8!
- Stage:
  - Das Fenster
  - Top-Level Container
- Scene:
  - Inhalte
  - “Scene Graph”
- Node:
  - Spezielle Klasse
  - Allgemeinster Node: Group()
  - Scene benötigt einen Node als Einstiegspunkt (root)

- http://docs.oracle.com/javafx/2/scenegraph/jfxpub-scenegraph.htm