Vorstellung

- 2005 Maturität
- 2008 – 2016 Polymechaniker
- seit 2010 Selbstständig im Nebenerwerb: Tontechnik, Akustik, Video, Softwareentwicklung
- seit 2014 Velokurier (Velo, Disposition, IT)
- 2015: BSc Elektro- und Kommunikationstechnik, Vertiefung Embedded Systems
- 2015 – 2016 DSP Entwicklung, Akustik
- seit 2016 Unterricht TEKO (Softwareentwicklung, Betriebssysteme, Netzwerktechnik, Microcomputer)

Lernziele I

Die Studierenden kennen die Methoden der objektorientierten Programmierung und können diese anwenden.
Sie sind in der Lage, mittelgrosse, vollständig objektorientierte, grafische Anwendungen zu implementieren, testen und dokumentieren.

Lernziele II

Die Studierenden

- kennen die Konzepte Kapselung, Vererbung, Polymorphie, dynamisches Binden, abstrakte Klassen und generische Programmierung und können diese in einfachten Beispielen anwenden. können den Kontrollfluss eines Programmes mit Ausnahmebehandlung verstehen und die Vorteile erläutern.
- kennen die SOLID - Prinzipien und können sie in eigenen Worten erklären.
- kennen die verschiedenen Testarten und können einfache Unit-Tests selber schreiben.

Lernziele II

Die Studierenden

- kennen das Vorgehen beim Test Driven Development und erkennen, dass Refactoring und Testing ein integraler Teil der Softwareentwicklung ist.
- kennen das Vorgehen sowie Vor- und Nachteile des Pair-Programming.
- können eine GUI-Applikation entwickeln. Sie können dabei gängige Objektorientierte Konzepte anwenden und den Code sinnvoll strukturieren.
- wissen, worauf sie bei der Auswahl eines Frameworks achten müssen.

Zeitplan

Montag 18:30 – 20:00 / 20:15 – 21:45

1. KW 43: Einstieg, Klassen und Objekte
2. KW 44: Testing, TDD
3. KW 45:
4. KW 46: Datenstrukturen
5. KW 47: Vererbung, Polymorphismus
6. KW 48: Repetition MVC
7. KW 49: statische/dynamische Bindung
8. KW 50: Clean Code
9. KW 51: Hardware, binäre Zahlendarstellung, Programmiersprachen und -werkzeuge

10. KW 2:
11. KW 3: Exceptions, Fehlerbehandlung
12. KW 4:
13. KW 5: SOLID: Single-Responsibility
14. KW 7: SOLID: Open Closed
15. KW 8: SOLID: Liskov
16. KW 9: SOLID: Interface-Segregation
17. KW 10: SOLID: Dependency-Inversion
18. KW 11: Frameworks
19. KW 12:

Unterlagen

- github.com/fhirter
- Literatur.pdf

Benotung

Ratschläge

- Sei anwesend im Unterricht
- Mach die Hausaufgaben
- Wenn du etwas nicht verstehst, frage! Dumme Fragen sind nur die, die nicht gestellt werden.

Softwareentwickler bauen Maschinen

- Unsere Maschinen können nicht angefasst werden: Sie sind nicht materiell
- Wir sprechen von Programmen oder Systemen (Software)
- Um eine Softwaremaschine laufen zu lassen brauchen sie eine physische Maschine: den Computer (Hardware)

Computer

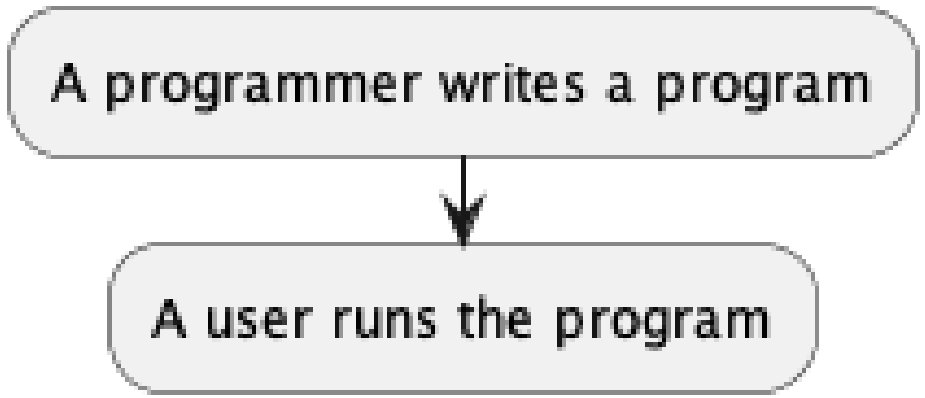
- Computer sind universelle Maschinen. Sie führen die Programme aus, die wir ihnen füttern.
- Die einzigen Grenzen sind unsere Vorstellungskraft
- Gute Nachricht
    - Dein Computer macht genau das was man ihm sagt.
    - Er macht es sehr schnell.
- Schlechte Nachricht
    - Dein Computer macht genau das was man ihm sagt.
    - Er macht es sehr schnell

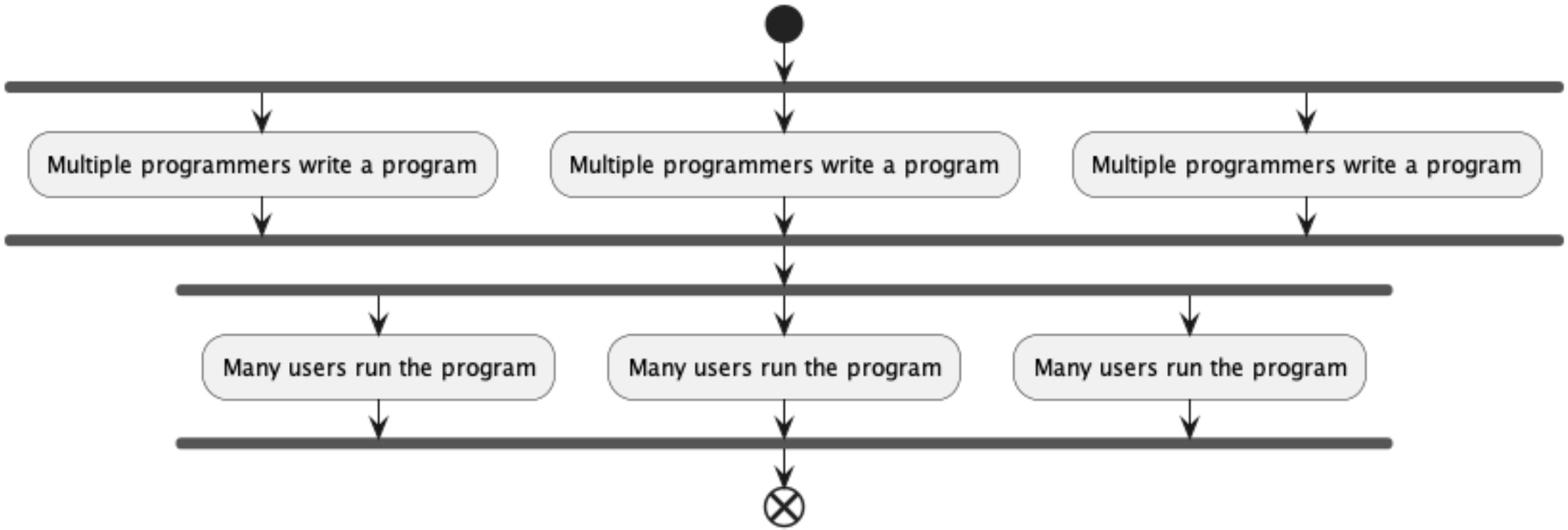
Programme erstellen und laufen lassen

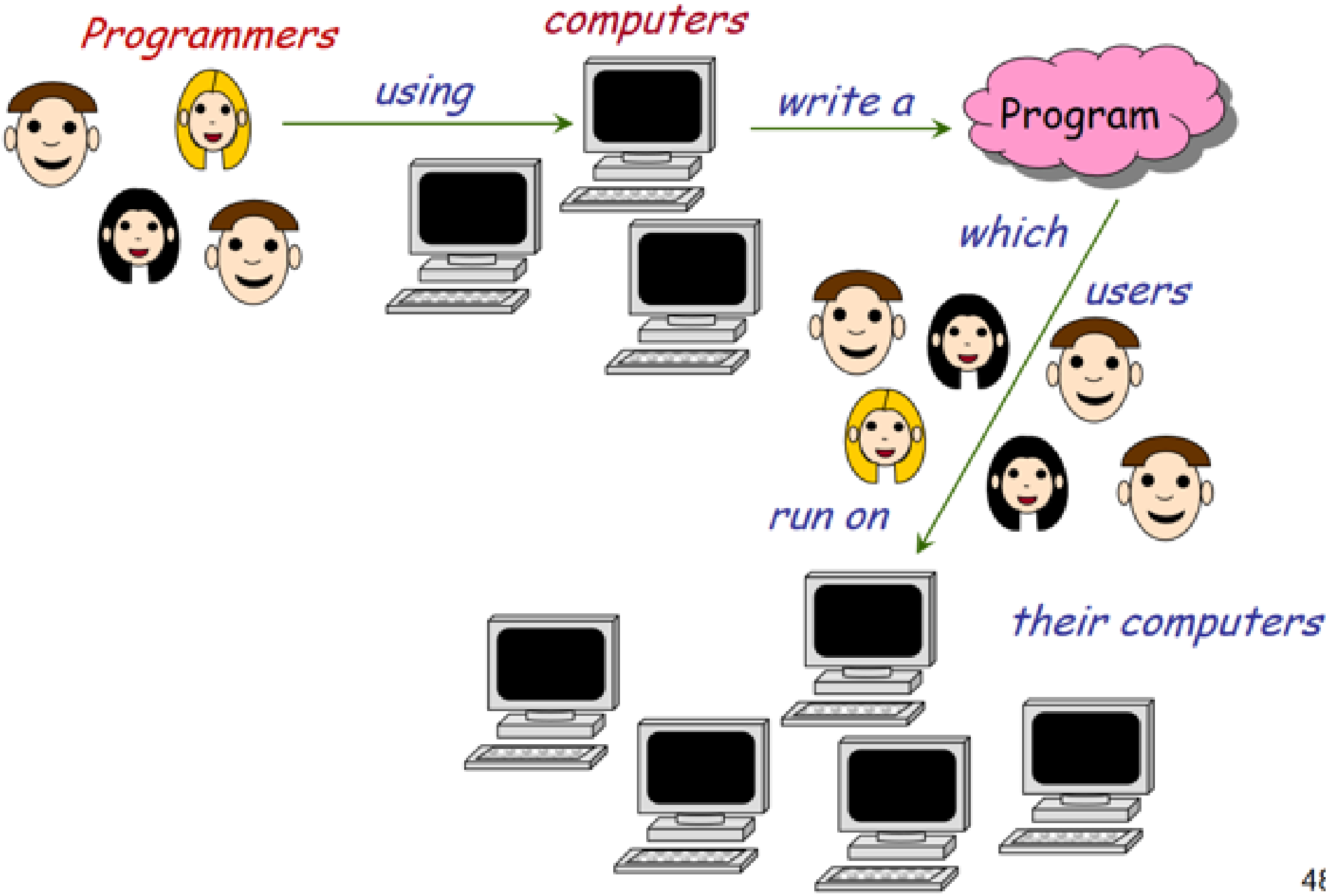
Programme erstellen und laufen lassen

Verbreitete Mythen und Entschuldigungen

- «Computer sind intelligent»
  - Fakt: Computer sind weder intelligent noch dumm. Sie führen Programme aus, die von Menschen entwickelt wurden.
  - Diese Programme bilden die Intelligenz ihrer Autoren ab.
  - Die grundlegenden Computeroperationen sind elementar (Speichere diesen Wert, Addiere diese beiden Zahlen…)
- «Der Computer ist abgestürzt»
- «Der Computer erlaubt das nicht»
- «Der Computer hat ihren Datensatz verloren»
- Fireship.io


A programmer writes a program
A user runs the program


Multiple programmers write a program
Multiple programmers write a program
Multiple programmers write a program
Many users run the program
Many users run the program
Many users run the program


Programmers
using
computers
write a
Program
which
users
run on
their computers