Hochschule für Angewandte
Wissenschaften Hamburg
Hamburg University of Applied Sciences

Programmiertechniken 2

Unit-Testing

Learning Outcomes

Sie ...

- ... entwerfen und verwenden Testklassen (Unit-Tests) während der Entwicklung von Software.

Status Quo

Wie können Sie Fehler in Ihrem Code finden? Wie debuggen Sie?

- Was ändern/ergänzen Sie dazu in Ihrem Code?
- An welchen Stellen?
- Was passiert mit diesen Änderungen nachdem der Fehler gefunden ist?
- Benutzen Sie bestimmte Tools oder Funktionen der IDE dafür?

Unit-Tests

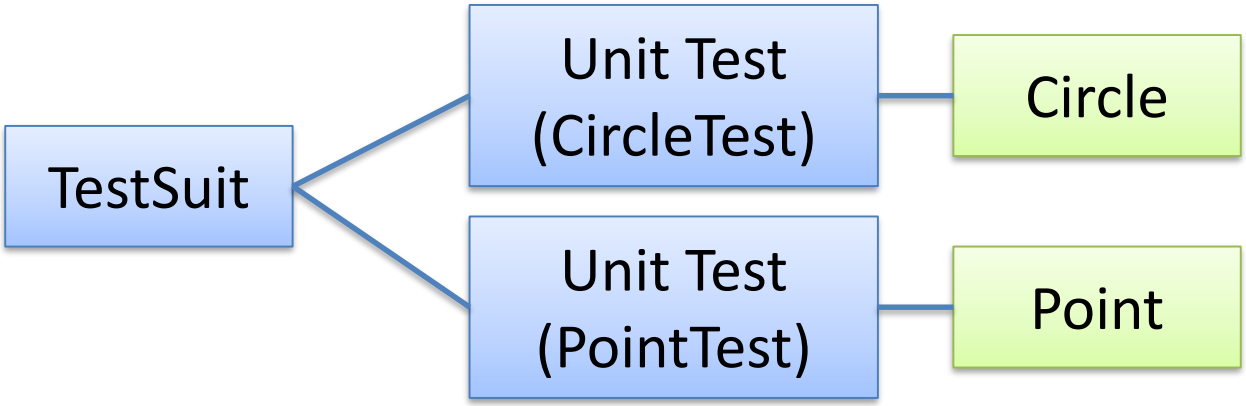
Ein Unit-Test soll unabhängige abgeschlossene Einheiten, eine Unit Under Test (UUT), testen. In der Objektorientierung sind Klassen/ Objekte die kleinsten Einheiten.

Der Unit-Test arbeiten einzelne unabhängige Testfälle automatisiert auf der UUT ab. Die einzelnen Ergebnisse (Abweichungen) werden protokolliert.

Der Unit-Test ruft bei der UUT Methoden auf und prüft die Reaktion gegen eine Erwartung. Die UUT enthält dabei keinen zusätzlichen Code um den Test zu unterstützen!

Unit-Tests

Unit-Tests werden durch Frameworks wie CPPUnit, JUnit, oder PyUnit der entsprechenden Sprachen unterstützt. Die Frameworks stellen Dienste für die Ausführung, den Vergleich und das Reporting zur Verfügung.



Zur vollautomatischen Durchführung in großen Projekten werden die Tests in Testsuits zusammengefasst.

Im Rahmen der Vorlesung wird das Google-Testing-Framework eingesetzt.

Bibliotheken / „Google Test“

- Google Testing Framework „Google Test“
- statische Bibliothek
- ???

- **Bibliotheken** enthalten Funktionen, die häufig benötigt werden, Mittel zur **Wiederverwendung** von Code

- **statisches Einbinden** in die ausführbare Datei (binary)
- **dynamisches Einbinden** zur Laufzeit: Windows *.dll und Linux *.so

- **statisch:** korrekte Version steht fest, keine fehlenden Abhängigkeiten zur Laufzeit
- **dynamisch:** weniger Speicherplatzverbrauch (in RAM, Festplatte/SSD und beim Laden und Installieren

Google Test - Vorbereitung

1. Google Test downloaden:
   https://github.com/google/googletest → z.B. download ZIP
2. Speichern+Auspacken, nicht im NetBeans-Projektordner
3. NetBeans: New C++ Static Library, Name z.B. googletest
4. Rechte Maustaste auf Projekt, Eigenschaften, C++:
   Include-Directory:
   1. `.../googletest`-**Unter**verzeichnis und
   2. `.../googletest/include` Unterverzeichnis
5. Unter „Source Files“ bestehende Dateien hinzufügen:
   `gtest-all.cc`
   `gtest_main.cc`
6. Rechte Maustaste auf Projekt + Build

Google Test – Verwendung (1/2)

1. Bestehendes NetBeans-C++-Projekt auswählen (z.B. NachbarnVL)
2. Rechte Maustaste auf „Test Files“, „New Test Folder“ Name: `testPerson`
3. Rechte Maustaste auf „testPerson“, Eigenschaften, C++: Include-Directory:
   1. `googletest`-**Unter**verzeichnis und
   2. `googletest/include` Unterverzeichnis
4. Und: „Linker“: „Libraries“ unter „Libraries“ dort „Add Project“ und NetBeans-googletest-Verzeichnis auswählen
5. Test: Rechte Maustaste auf „testPerson“ und „Test“

Google Test – Verwendung (2/2)

1. Rechte Maustaste auf „testPerson“, „New“ – „C++ Source File“, Name: `test.cpp`
2. `test.cpp` öffnen, dort hinzufügen:

```
#include <gtest/gtest.h>
#include “Person.h“
```

3. Dann Test-Cases hinzufügen, z.B.:

```
TEST(PersonTest, testConstructor) {
    Person p(“Test Name“, 23, 42);
    EXPECT_EQ(23, p.getAlter());
}
```

4. Test: Rechte Maustaste auf „testPerson“ und „Test“