Aufgabe: Bedeutung der Entscheidbarkeit

Nach Meinung vieler Informatiker ist die Tatsache, dass manche Probleme unentscheidbar sind, aus praktischer Sicht nur von untergeordneter Bedeutung.

Suchen Sie in Gruppen zu zwei bis drei Personen Antworten auf folgende Fragen:

1. Woran erkennt man unentscheidbare Probleme in der Praxis?
2. Woran könnte es liegen, dass die Unentscheidbarkeit praktisch nur von untergeordneter Bedeutung ist?

Zeit: 3 Minuten

PK

Statisches Denken

Zusicherungen als Kommentare

```
zahl = (new Random()).nextInt() % grenze;
// -grenze < zahl < grenze     (wegen ... % grenze)

if (zahl < 0) {
    // -grenze < zahl < 0       (wegen Bedingung zahl < 0)
    zahl = zahl + grenze;
    // 0 < zahl < grenze        (wegen Addition von grenze)
}

// 0 < zahl < grenze           (wenn Bedingung wahr war)
// oder 0 <= zahl < grenze     (wenn Bedingung falsch war)
// ergibt: 0 <= zahl < grenze
```

assert-Anweisung in Java

```
zahl = (new Random()).nextInt() % grenze;
assert((-grenze < zahl) && (zahl < grenze));

if (zahl < 0) {
    assert((-grenze < zahl) && (zahl < 0));
    zahl = zahl + grenze;
    assert((0 < zahl) && (zahl < grenze));
}

assert((0 <= zahl) && (zahl < grenze));
```

Zusicherungen auf Schnittstellen

```
// Initialisierung mit Zufallszahl x; 0 <= x < grenze
// Voraussetzung: grenze > 0
public UnbekannteZahl(int grenze) { ... }
```

Vorbedingung: vor Methodenausführung erfüllt (Parameter)

Nachbedingung: nach Methodenausführung erfüllt

Aufgabe: Warum statisch?

Beim Programmieren kommt es hauptsächlich darauf an, Programme und Algorithmen statisch zu verstehen, obwohl die Ausführung dynamisch ist. Warum ist das so?

A: Weil das Programm selbst statisch ist.

B: Weil das Nachvollziehen der Ausführung viel zu aufwendig wäre.

C: Weil das Konzept der Zusicherungen nur statisch funktioniert.

D: Das stimmt nicht. Zusicherungen werden dynamisch überprüft.

PK

Softwareentwicklung

Softwarelebenszyklus

Idee

Entwicklung

Anwendung und Wartung (gleichzeitig)

letzte Anwendung

Softwareentwicklungsschritte

zyklisch wiederholt (und oft überlappend):

Analyse	
Entwurf	Programmieren
Implementiertung	
Verifikation	
Testen	
Validierung	

Schritte beim Programmieren

zyklisch wiederholt:

- Planen
- Editieren
- Übersetzen
- Testen
- Debuggen

Programmierwerkzeuge

Editor

Compiler und Interpreter

Debugger

Integrierte Entwicklungsumgebung