

Einführung in die Fehleranalyse

Natja Scharz
Siegen, Juni 2003

Ort: Gymnasium „Auf der Morgenröte“, Siegen
Lerngruppe: Informatik Grundkurs 11.2
Zeit: 02.05.2003, Einzelstunde 9:35-10:20 Uhr

1 Thematischer Zusammenhang

1.1 Reihenthema

Erschließung theoretischer Probleme der Informatik und falls möglich, deren Verknüpfung mit der Praxis.

1.2 Stundenthema

Einführung in die Fehleranalyse anhand ausgewählter Programmbeispielen.

1.3 Anschluss an die Vorstunde

In der vorherigen Stunde wurde das Thema UML abgeschlossen und in dieser Stunde soll der neue Block Fehleranalyse beginnen. Mit der Hausaufgabe der letzten Stunde wurde eine Brücke zum neuen Thema geschaffen. Die Schüler[innen] erhielten Disketten mit unterschiedlichen fehlerhaften Programmen und sollten versuchen syntaktische und semantische Fehler zu entdecken. In dieser Stunde wird die Hausaufgabe aufgegriffen und die Ergebnisse besprochen. Dann soll ein fließender Übergang zur strukturierten Analyse des Programmcodes stattfinden.

2 Ziele

2.1 Stundenlernziel

Die Schüler[innen] sollen nach dieser Stunde die Vorzüge der strukturierten Fehleranalyse kennen gelernt haben. Sie sollen häufig auftretende typische Syntaxfehler und Semantikfehler kennen gelernt haben

2.2 Teillernziele

kognitive Zieldimension:

Die Schülerinnen und Schüler

1. lernen die Bedeutung der Fehleranalyse kennen
2. erkennen die Vorteile der strukturierten Analyse
3. lernen das gezielte Suchen nach Syntax- und Semantikfehler kennen
4. lernen typische häufig auftretende Fehler kennen

affektive Zieldimension:

Die Schülerinnen und Schüler

1. Zufriedenheit an der strukturierten Fehleranalyse
2. empfinden die strukturierte Fehleranalyse als neue Möglichkeit schnell und effektiv Programmfehler aufzuspüren

3 Hausaufgaben

3.1 Zur Stunde

Als Hausaufgabe zu dieser Stunde wurden Disketten mit fehlerhaftem Programmcode ausgeteilt. (Siehe Aufgabenstellung zur vorherigen Stunde) Die Schüler[innen] sollten versuchen diese Fehler zu finden und ihre Ursache erkennen(Syntax) erkennen.

3.2 Für die nächste Stunde

Die Schüler[in] erhalten die ersten vier Mini-Referate mit den Themen:

- Geschichte der UML
- Aktivitätsdiagramm
- Zustandsdiagramm
- Beziehungen und Kardinalitäten

5 Begründung zur Gestaltung des Unterrichts

Da der vorgesehene Lernstoff von UML abgearbeitet ist und die Schüler[innen] die notwendigen Grundkenntnisse erlangt haben, startet nun die nächste Unterrichtsreihe. Da die Programmplanung, mit Hilfe der grafischen Sprache UML und in unserem ersten Projekt die konkrete Programmerstellung als Unterrichtsreihe schon abgedeckt wurde, gehe ich nun auf die Analyse des fertigen Programms ein. Nach dieser Unterrichtsreihe sind dann alle Phasen der Programmerstellung von der Planung, Programmierung bis zur Fehleranalyse und Testen durchlaufen worden und die Schüler[innen] haben in alle diese Bereiche Einblick erhalten.

6 Fehlerhafter Code

Es sind drei Syntaxfehler eingebaut!

```
unit mKugel9;
```

```
uses mSuM, mTisch2;
```

```
type kugel = class
```

```
    hStift : Buntstift;
```

```
    kTisch : Tisch;
```

```
    zGroesse, zGeschwindigkeit : Zahl; // Zustandsvariable ("Zettel") für die Größe und die Geschw.
```

```
    constructor init(phPos, pvPos, pRichtung, pGroesse, pGeschwindigkeit: Zahl); virtual; // Darf später mal überschrieben werden.
```

```
    procedure merkeTisch(pTisch: Tisch);
```

```
    procedure zeigeDich;
```

```
    procedure versteckeDich;
```

```
    procedure rolle(pTempo: Zahl); virtual; // Darf später mal überschrieben werden.
```

```
    procedure setzeRichtung(pRichtung: Zahl);
```

```
    function gibhPos: Zahl;
```

```
    function gibvPos: Zahl;
```

```
    function gibRichtung: Zahl;
```

```
    destructor gibFrei;
```

```
end;
```

```
constructor Kugel.init(phPos, pvPos, pRichtung, pGroesse, pGeschwindigkeit: Zahl);
```

```
begin
```

```
    hStift := Buntstift.init;
```

```
    hStift.bewegeBis(phPos, pvPos);
```

```
    hStift.dreheBis(pRichtung);
```

```
    hStift.setzeFuellmuster(GEFUELLT);
```

```
    zGroesse := pGroesse;
```

```
    zGeschwindigkeit := pGeschwindigkeit;
```

```
end;
```

```
function Kugel.merkeTisch(pTisch: Tisch);
```

```
begin
```

```
    kTisch := pTisch;
```

```
end;
```

```

procedure Kugel.zeigeDich;
begin
  hStift.zeichneKreis(zgroesse);
end;

procedure Kugel.versteckeDich;
begin
  hStift.radiere;
  self.zeigeDich;
  hStift.normal;
end;

procedure Kugel.rolle(pTempo: Zahl);
begin
  self.versteckeDich;
  hStift.bewegeUm(pTempo);
  if self.gibhPos >= kTisch.gibhPos+kTisch.gibBreite-1-zGroesse then self.setzeRichtung(180-
self.gibRichtung);
  if self.gibhPos <= kTisch.gibhPos+2+zGroesse then self.setzeRichtung(180-self.gibRichtung);
  if self.gibvPos >= kTisch.gibvPos+kTisch.gibHoehe-1-zGroesse then self.setzeRichtung(-
self.gibRichtung);
  if self.gibvPos <= kTisch.gibvPos+2+zGroesse then self.setzeRichtung(-self.gibRichtung);
  self.zeigeDich;
end;

procedure Kugel.setzeRichtung(pRichtung: Zahl);
begin
  hStift.dreheBis(pRichtung);
end;

procedure Kugel.gibhPos: Zahl;
begin
  result := hStift.hPosition;
end;

function Kugel.gibvPos: Zahl;
begin
  result := hStift.vPosition;
end;

function Kugel.gibRichtung: Zahl;
begin
  result := hStift.winkel;
end;

destructor Kugel.gibFrei;
begin
  self.versteckeDich;
  hStift.gibFrei;
end;

```