Modul 24

Objektorientierte Programmierung

VITRUVIUS HOCHSCHULE

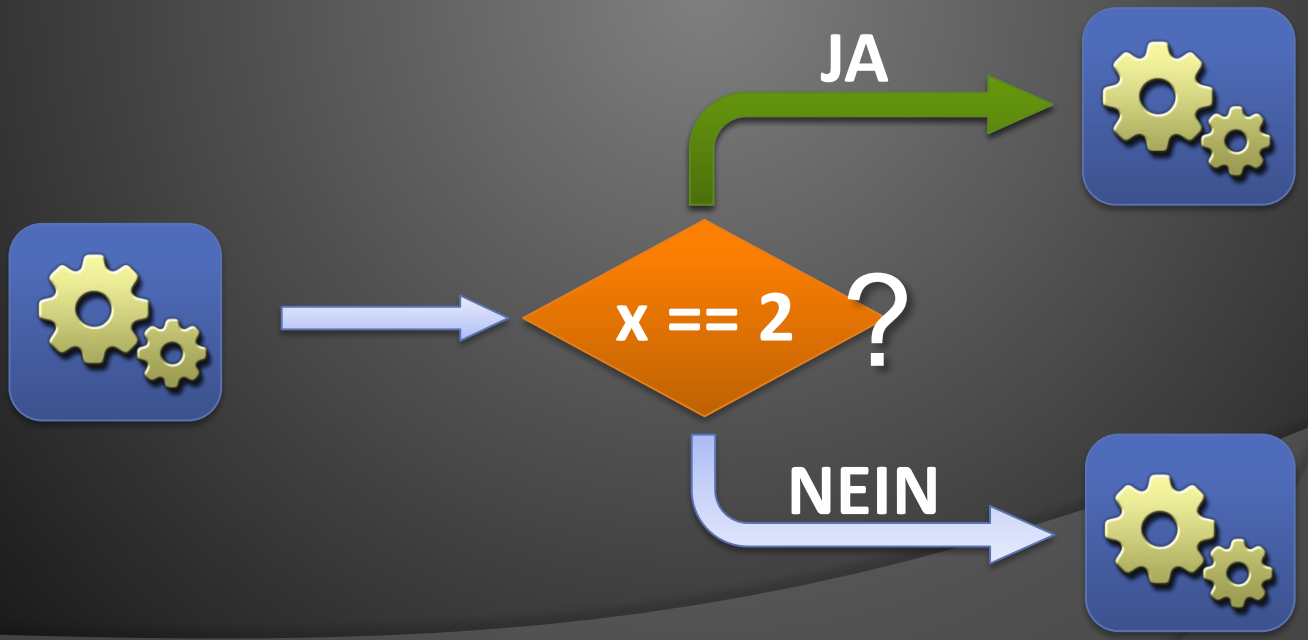
Oliver Ziegler

Bedingungen und Entscheidungen

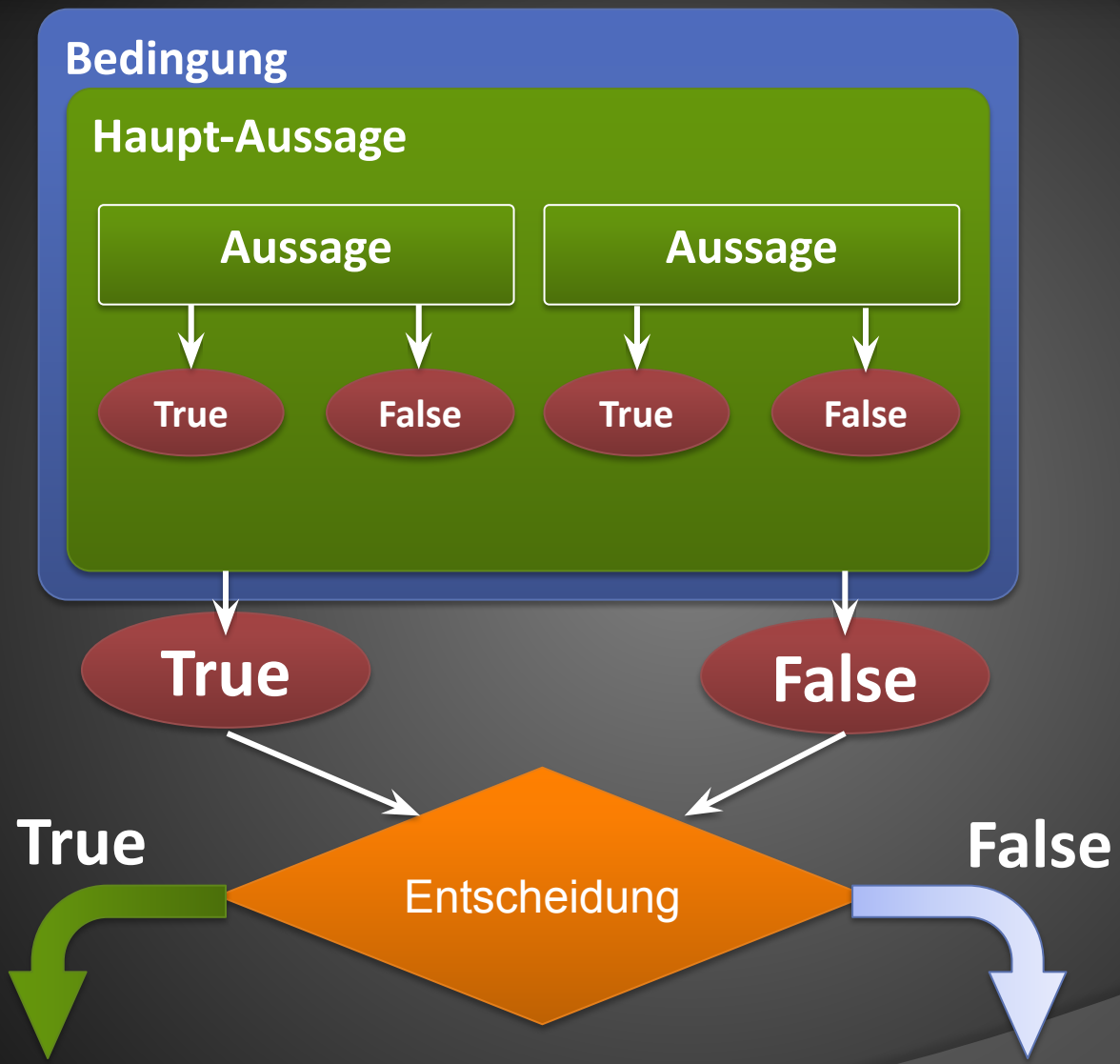
- Herzstück jedes Computers, jedes Programms, jedes Spiels
- Aktionen nicht immer gleich, sondern Situationsabhängig
- Reagieren des Programms/Spiels auf den Nutzer/Spieler

Bedingungen

- Entscheidungen anhand von Zuständen (Aussagen) treffen
- Essentiell, sonst müsste es für jede Funktionalität einen eigenen Button geben



Begriffe


Bedingung
Haupt-Aussage
Aussage
Aussage
True
False
True
False
True
False
True
Entscheidung
False

Aussagen >Wahrheitswerte (bool)>Entscheidung

- Aussagen lassen sich auf einen von zwei Werten herunterbrechen

WAHR **TRUE**

FALSCH **FALSE**

GodMode	Ist GodMode an?
Gegner tot	Ist Gesundheit <= 0 ?
Schwierigkeitsgrad	Ist Schwierigkeitsgrad „Hardcore“ an?
	Ist der Wert des Schwierigkeitsgrads == „Hardcore“?
Aufgabe erledigt?	Sind Gegner A und Gegner B tot?
(Töte A und B)	(Ist Gegner A tot?) UND (Ist Gegner B tot?)
	(Ist Gesundheit von A < 0) UND (Ist Ges. von B < 0)

Boolean Datentyp (bool)

- Kann nur zwei Werte annehmen

```
bool condition;
public void MyFct()
{
        condition =         true;
        condition =         false;
}
```

Vergleiche (Aussage)

- Resultieren zur Laufzeit in einem boolean-Wert

```
bool condition;
public void MyFct()
{
      condition =          12 == 12;
            // condition   ist   true
      condition =          12 > 15;
            // condition   ist   false
      condition =          „Text“ == „Text“;
            // condition   ist   true
}
```

Vergleiche (Aussage)

- Resultieren zur Laufzeit in einem boolean-Wert

```
bool condition;
int x = 30;
int y = 30;
public void MyFct()
{
      condition =       x == y;       // true
      condition =       x  >  y;      // false
      condition =       x >= y;       // true

      //UNGLEICH
      condition =       x !=  y;      //false


}
```

Bedingungen

- Größer , GrößerGleich, Gleich, Ungleich
- Beachte Unterschied Zuweisung (=) und Vergleich- IstGleich (==)

```
int x = 4;
int y = 5;

if( x > y)              // Größer
{   DoStuff();  }

if( x >= y)             // GrößerOderGleich
{   DoStuff();  }

if( x == y)             // Gleich
{   DoStuff();  }

if( x != y )            // Ungleich
    DoStuff();
```

Bedingungen

- Verneinung

```
bool condition = true;

if( !condition )          //enspricht  !true  also  false
{   DoStuff();  }

if( ! (myNumber > 734) )
{   DoStuff();  }

if( myNumber != 734)
    DoStuff();
DoThisStuffAnyway();
```

Bedingungen

- Verknüpfung

```
bool condition = true;
bool status = true;

if( condition && status )
{   DoStuff();   }

if( condition || status )
{   DoStuff();   }

if(  condition && !status)
    DoStuff();
DoThisStuffAnyway();
```

Bedingungen

- Verschachtelung

```
if( condition && ( x > y  ||  condition2) )
{   DoStuff();  }

if( condition || status )
{   DoStuff();  }

if(  condition && !status)
    DoStuff();
DoThisStuffAnyway();
```

Bedingungen

- **If – Abfrage**
- „condition“ kann ein vorher berechneter bool-Wert sein, aber auch eine komplexe Verkettung von logischen Operationen (Boolsche Operationen)

```
public void MyFct()
{
    if(condition)   FALSE
    {
        Function1();
    }

    Function2();        // Wird immer ausgeführt, weil er
                        außerhalb der Klammern liegt
}
```

Bedingungen

- Abfragen

```
bool condition = true;

if( condition == true)
{
DoStuff();
}

if( condition )
{
DoStuff();
}

// Bedeutet beides dasselbe    x == true? => x
```

Bedingungen

- Abfragen

```
bool condition = true;

if( condition )
{
DoStuff();
}

if( 552 > 734 )
{   DoStuff();   }

if( myNumber > 552 )
{
    DoStuff();
}
```

Bedingungen

- Schreibweisen

```
public void MyFct()
{
    if( condition )
    {
        DoStuff1();
        DoStuff2();
    }

    if( condition )
        DoStuff1();
    DoStuff2();

}
```

In Klammern zusammengefasst
Werden beide ausgeführt, wenn condition == true
Sonst beide nicht!

Ohne Klammern
Nur der erste Befehl ist von der if-Abfrage abhängig

Wird immer ausgeführt!
Nicht in Klammern und nicht erste Zeile nach der if-Abfrage

Bedingungen

- If - Abfrage

```
public void MyFct()
{
    if(condition)   FALSE
    {
        Function1();
    }

    if(condition2)  FALSE
        Function2();

    Function3();  // Wird immer ausgeführt, weil eine if-Abfrage ohne Klammern nur den nächsten Befehl verhindert/erlaubt
}
```

AUFGABE

Evade – Folie 9

Bedingungen

- If / Else- Abfrage

```
public void MyFct()
{
    if(condition)   FALSE
    {
        Function1();
    }
    else
    {
        Function2();
    }
    DoStuff(); //immer, danach
}
```

- Eine von beiden Befehlsketten wird immer ausgeführt
- Aber nie beide

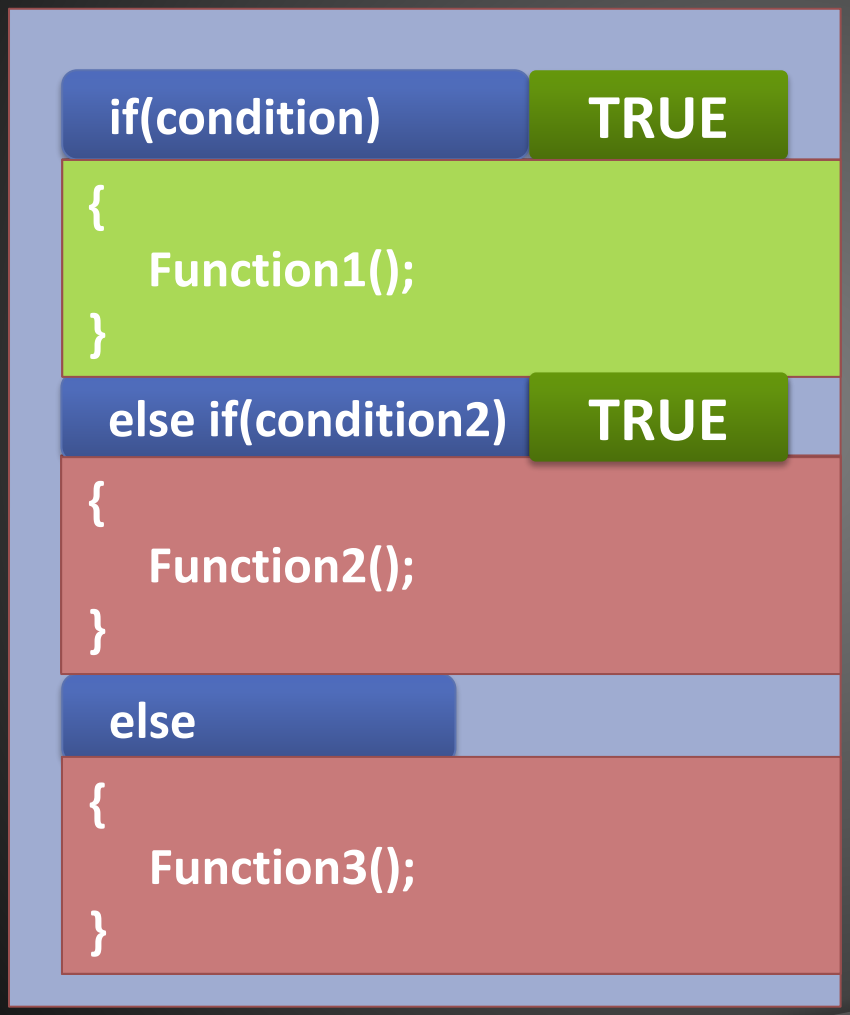
Bedingungen

- **Verneinung**
  (Kehrt den Wahrheitswert einer Aussage um)

```
bool condition = true;

if( !condition )          //enspricht  !true  also  false
{   DoStuff();  }

if(condition == false)
{   DoStuff();  }

if( ! (myNumber > 734) )
{   DoStuff();  }

if( myNumber != 734)
    DoStuff();
DoThisStuffAnyway();
```

Bedingungen

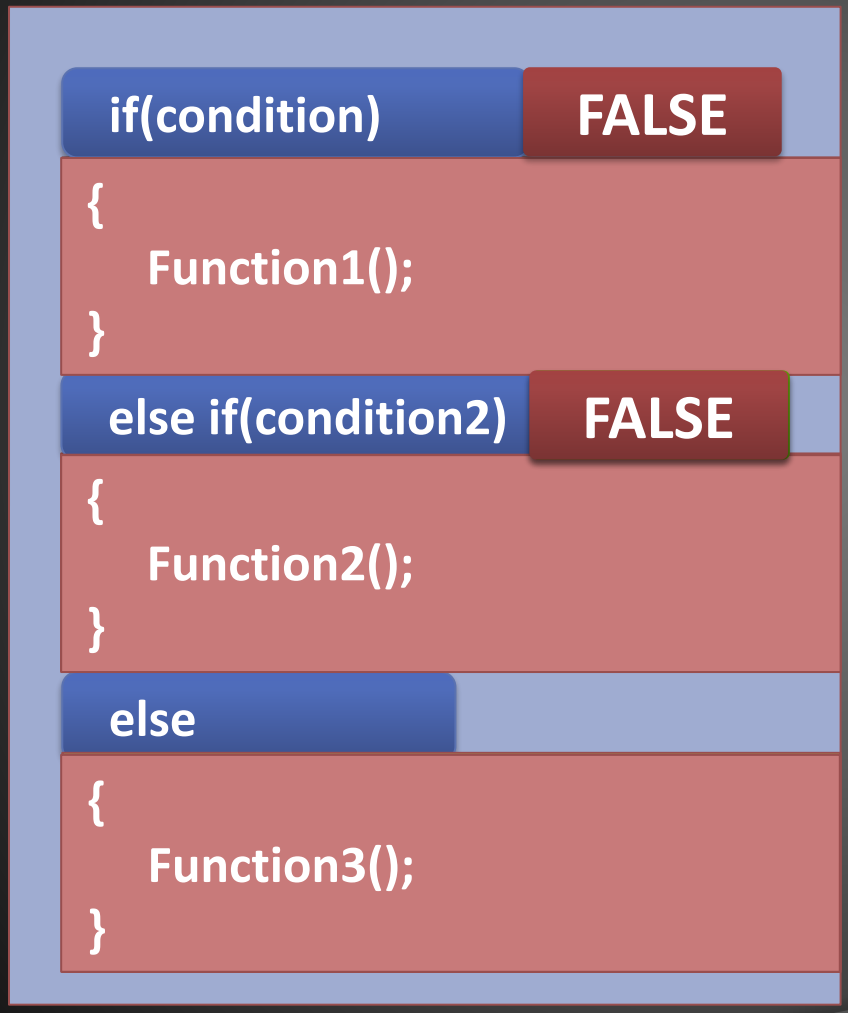
- If / Else if / else- Abfrage



- Es kann mehrere else if – Bedingungen geben
- Es wird immer nur eine Befehlskette ausgeführt
- Die erste Bedingung die wahr ist, bekommt den Zuschlag
- Auch wenn die späteren auch wahr sind
- Ist KEINE Bedingung wahr, wird „else“ ausgeführt

Bedingungen

- If / Else if / else- Abfrage



- Es kann mehrere else if – Bedingungen geben
- Es wird immer nur eine Befehlskette ausgeführt
- Die erste Bedingung die wahr ist, bekommt den Zuschlag
- Auch wenn die späteren auch wahr sind
- Ist KEINE Bedingung wahr, wird „else“ ausgeführt

Bedingungen

- else / else if

```
bool condition = true;

if( condition )
{   DoStuff();   }
else
{   DoSomeOtherStuff();  }

if( frust >= 100 )
{   DoStuff();   }
else if ( condition )
{   BeendeSpiel();  }
else
{   DoSomeOtherStuff();  }
```

Bedingungen

- Größer , GrößerGleich, Gleich, Ungleich
- Beachte Unterschied Zuweisung (=) und Vergleich- IstGleich (==)

```
int x = 4;
int y = 5;

if( x > y)              // Größer
{   DoStuff();  }

if( x >= y)             // GrößerOderGleich
{   DoStuff();  }

if( x == y)             // Gleich
{   DoStuff();  }

if( x != y )            // Ungleich
    DoStuff();
```

Bedingungen

- Selbststudium zum Thema

Verknüpfung von Aussagen
Boolsche Operationen

Verknüpfung von Aussagen

- Resultieren zur Laufzeit in einem boolean-Wert

```
bool condition;
public void MyFct()
{
      condition =    ( zustand == 2 && livePoints < 0 );

      // && = „UND“
      // Ergibt „true“, wenn beide Aussagen „true“ sind
}
```

Verknüpfung von Aussagen

- Resultieren zur Laufzeit in einem boolean-Wert

zustand = 2
livePoints = 20

- zustand == 2 UND livePoints <= 0
- true UND livePoints <= 0
- true UND false
- false

Verknüpfung von Aussagen

- Resultieren zur Laufzeit in einem boolean-Wert

```
bool condition;
public void MyFct()
{
      condition =    ( zustand == 2 || livePoints < 0 );

      //|| = „ODER“
      // Ergibt „true“, wenn eine oder beide Aussagen
      „true“ sind
}
```

Verknüpfung von Aussagen

- Resultieren zur Laufzeit in einem boolean-Wert

zustand = 2
livePoints = 20

- zustand == 2 ODER livePoints <= 0
- true ODER livePoints <= 0
- true ODER false
- true

Bedingungen

- Verknüpfung

```
bool condition = true;
bool status = true;

if( condition && status )
{   DoStuff();   }

if( condition || status )
{   DoStuff();   }

if(  condition && !status)
    DoStuff();
DoThisStuffAnyway();
```

Verknüpfung von Aussagen (Klammern)

- Resultieren zur Laufzeit in einem boolean-Wert

zustand = 3
livePoints = 0
test = true

- (zustand == 2 UND livePoints <= 0) ODER test
- (false UND livePoints <= 0) ODER test
- (false UND true) ODER test
- (false) ODER test
- false ODER true
- true

Verknüpfung von Aussagen (Klammern)

- Resultieren zur Laufzeit in einem boolean-Wert

zustand = 3
livePoints = 0
test = true

- zustand == 2 UND (livePoints <= 0 ODER test)
- false UND (livePoints <= 0 ODER test)
- false UND (true ODER test)
- false UND (true ODER true)
- false UND true
- false

Bedingungen

- Verschachtelung

```
if( condition && ( x > y  ||  condition2) )
{   DoStuff();  }

if( condition || status )
{   DoStuff();  }

if(  condition && !status)
    DoStuff();
DoThisStuffAnyway();
```

Bedingungen

- **HÄUFIGSTER FEHLER:**

```
if( condition && ( x > y  ||  condition2) )
{   DoStuff();  }

if( condition = status )
{   DoStuff();  }

if(  condition && !status)
    DoStuff();
DoThisStuffAnyway();
```

Ein „=„ statt 2x

„=„ Variablen Zuweisung

„==„ Vergleich

Switch - Abfrage

- Abfragen einer Variable mit mehreren möglichen Zuständen

```
public void ReactToStars( int starAmount)
{
    switch (starAmount)
    {
      case 1:   Say( „Das war ganz gut.“);
            break;
      case 2:   Say( „Hey, nicht übel!“);
            break;
      case 3:   Say( „Wow! Beeindruckend!“);
            break;
      default: Say( „Na das war wohl nichts.“ );
            break;
    }
}
```