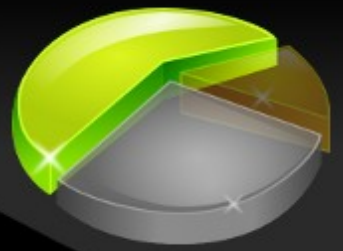
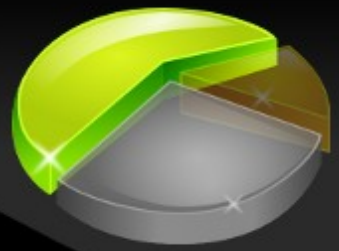
Softwaremetriken verstehen und nutzen



Kore Nordmann <kore@php.net>,
Manuel Pichler <mapi@pdepend.org>

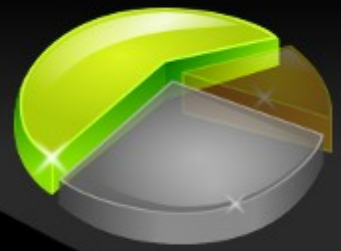
17. November 2009

Über uns



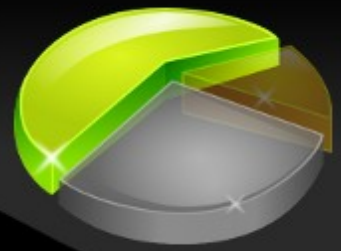
- Kore Nordmann <kore@php.net>
  - Langzeit PHP-Entwickler
  - Speaker auf diversen Konferenzen, Autor etc.
  - Entwickler diverser Open-Source-Projekte:
    - eZ Components, Arbit, PHPUnit, PHPillow
- Manuel Pichler <mapi@pdepend.org>
  - Jahrgang 1978
  - Diplom Informatiker, Softwarearchitekt
  - Entwickler diverser Projekte:
    - PHP_Depend, phpUnderControl, PHPMD

Agenda



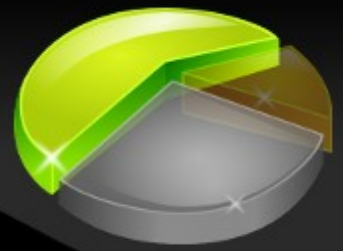
- Was sind Metriken?
- Welche Arten an Metriken gibt es?
- Klassische Softwaremetriken
  - Umfang
  - Komplexität
- Objektorientierte Softwaremetriken
  - Kopplung
  - Abstraktion

Softwaremetriken



- Eine Softwaremetrik ist eine Maßzahl für ein Qualitätsmerkmal von Software
  - „Funktionen zur Ermittlung von Kennzahlen eines Softwareartefakts“ (Wikipedia)
  - „You cannot control what you cannot measure.“ (Tom DeMarco)
    - Auch wenn er diese Aussage bereits relativierte
  - „Ohne Messung können Auswirkungen durch Änderungen nicht bewertet werden“ (A. Fleischer)

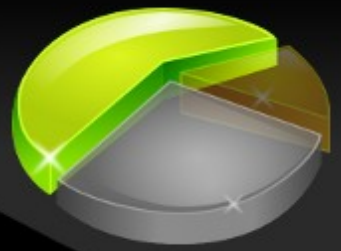
Arten an Metriken



- Prozessmetrik
- Aufwandsmetrik
- Projektlaufzeitmetrik
- Anwendungsmetrik

Quelle: Wikipedia

Vergesst das aber alles

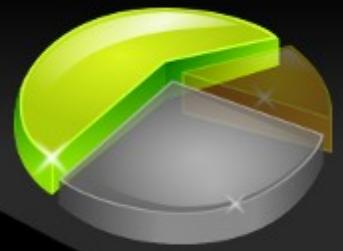


wir beschäftigen uns ausschließlich

mit

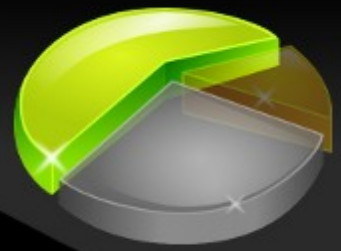
Produktmetriken

Agenda



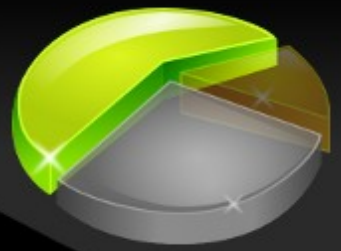
- Umfang
- Komplexität
- Kopplung, Kopplung & Abstraktion
- Lesbarkeit
- Code Rank

Umfangsmetriken



- Summen über Softwareartefakte
  - Lines Of *
    - LOC - Lines Of Code
    - ELOC - Executable Lines Of Code
    - CLOC - Comment Lines Of Code
    - NCLOC - None Comment Lines Of Code
  - Number Of *
    - NOC - Number Of Classes
    - NOM - Number Of Methods
    - NOP - Number Of Packages

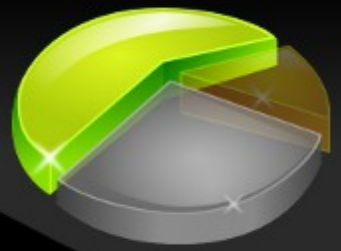
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

- Lines Of *
  - LOC =
  - ELOC =
  - CLOC =
  - NCLOC =
- Number Of *
  - NOC =
  - NOM =
  - NOP =

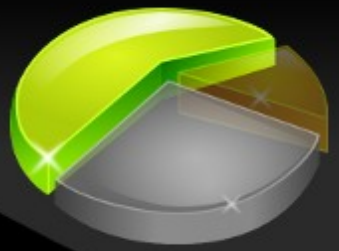
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

- Lines Of *
  - LOC = 16
  - ELOC =
  - CLOC =
  - NCLOC =
- Number Of *
  - NOC =
  - NOM =
  - NOP =

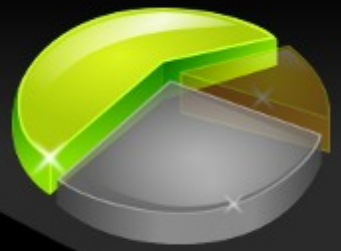
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

- Lines Of *
  - LOC = 16
  - ELOC = 3
  - CLOC =
  - NCLOC =
- Number Of *
  - NOC =
  - NOM =
  - NOP =

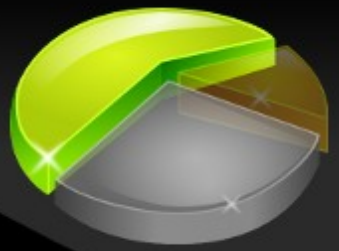
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

- Lines Of *
  - LOC = 16
  - ELOC = 3
  - CLOC = 2
  - NCLOC =
- Number Of *
  - NOC =
  - NOM =
  - NOP =

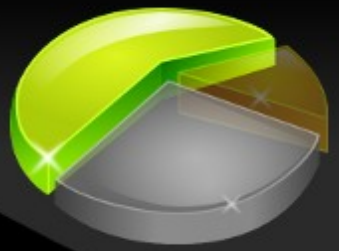
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

- Lines Of *
  - LOC = 16
  - ELOC = 3
  - CLOC = 2
  - NCLOC = 14
- Number Of *
  - NOC =
  - NOM =
  - NOP =

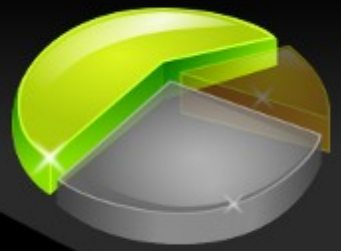
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

- Lines Of *
  - LOC = 16
  - ELOC = 3
  - CLOC = 2
  - NCLOC = 14
- Number Of *
  - NOC = 3
  - NOM =
  - NOP =

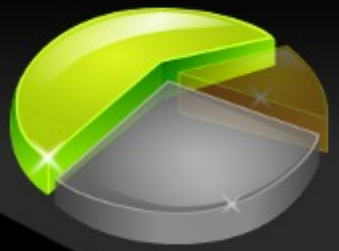
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

- Lines Of *
  - LOC = 16
  - ELOC = 3
  - CLOC = 2
  - NCLOC = 14
- Number Of *
  - NOC = 3
  - NOM = 4
  - NOP =

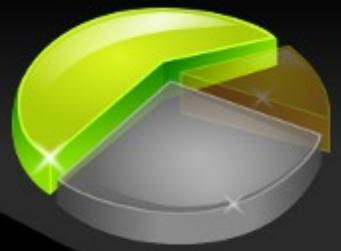
Lines Of *, Number Of *



```php
<?php
namespace foo\bar;

abstract class FooBar {
    abstract function bar();
}

class Foo extends FooBar {
    /* Does this ... */
    public function bar() {}
    /* Does that ... */
    public function baz() {}
}

class Bar extends Foo {
    public function foo(Foo $f) {}
}
```

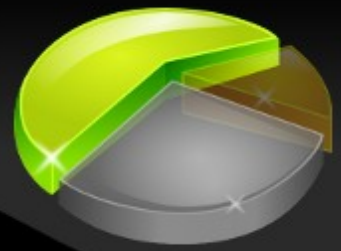
- Lines Of *
  - LOC = 16
  - ELOC = 3
  - CLOC = 2
  - NCLOC = 14
- Number Of *
  - NOC = 3
  - NOM = 4
  - NOP = 1

Beispiel



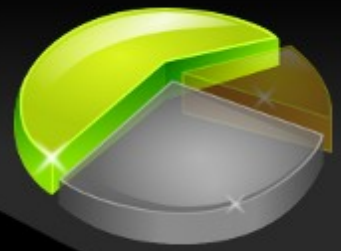
...

Agenda



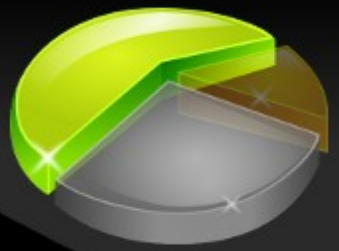
- Umfang
- Komplexität
- Klassische OO-Metriken
- Kopplung, Abstraktion & Instabilität
- Code Rank

Komplexitätsmetriken



- Maß für Komplexität sind Kontrollstrukturen
  - if, elseif, for, while, foreach, catch, case, xor, and, or, &&, ||, ?
- Cyclomatic Complexity (CCN)
  - Anzahl der *Verzweigungen*
- NPath Complexity
  - Anzahl der *Ausführungspfade*
  - Berücksichtigt die Struktur von Blöcken

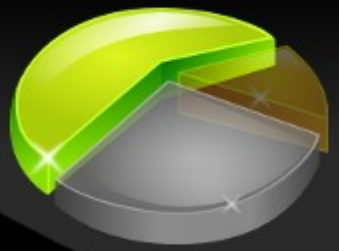
Cyclomatic Complexity



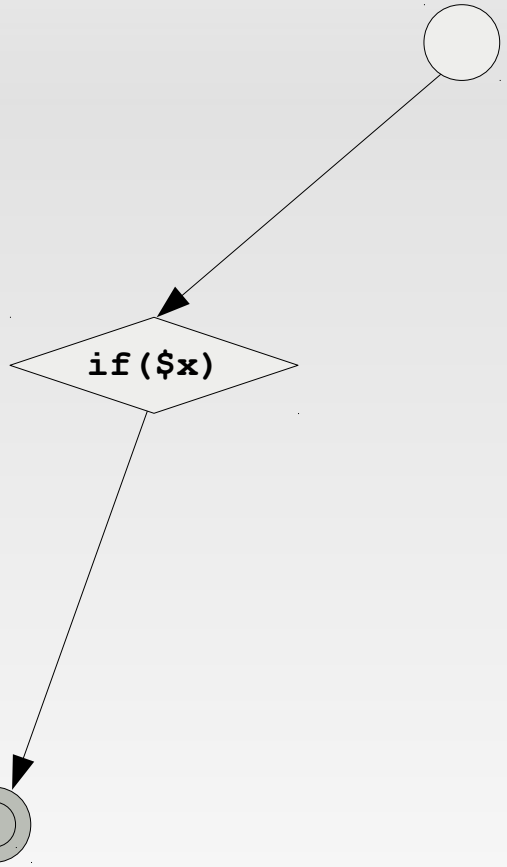
```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```

CCN = 0

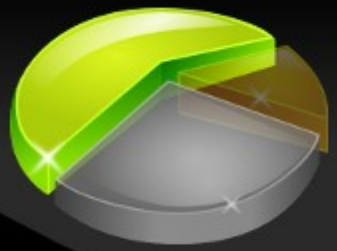
Cyclomatic Complexity



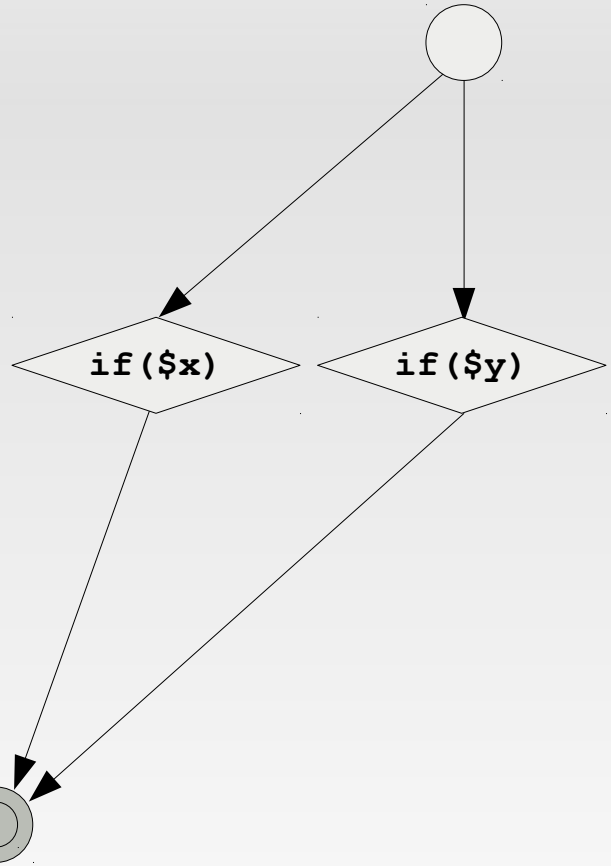
```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



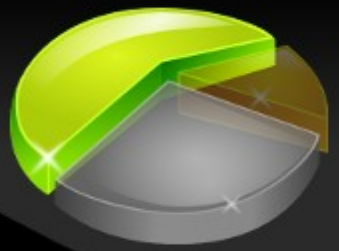
Cyclomatic Complexity



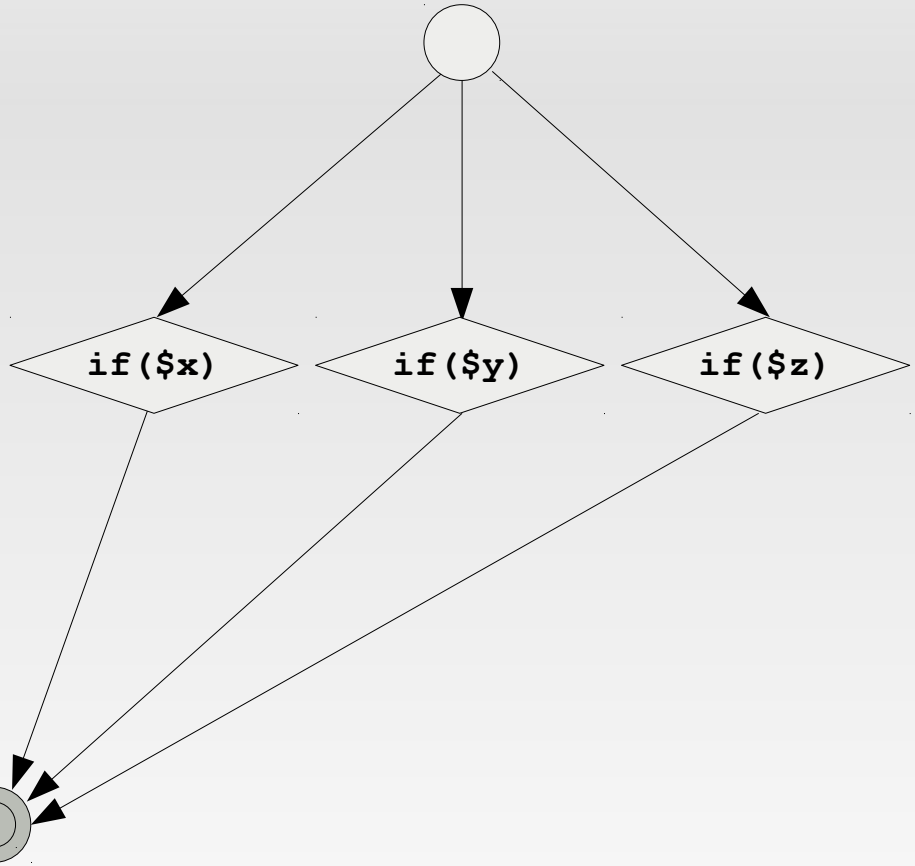
```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



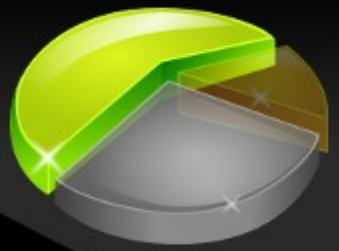
Cyclomatic Complexity



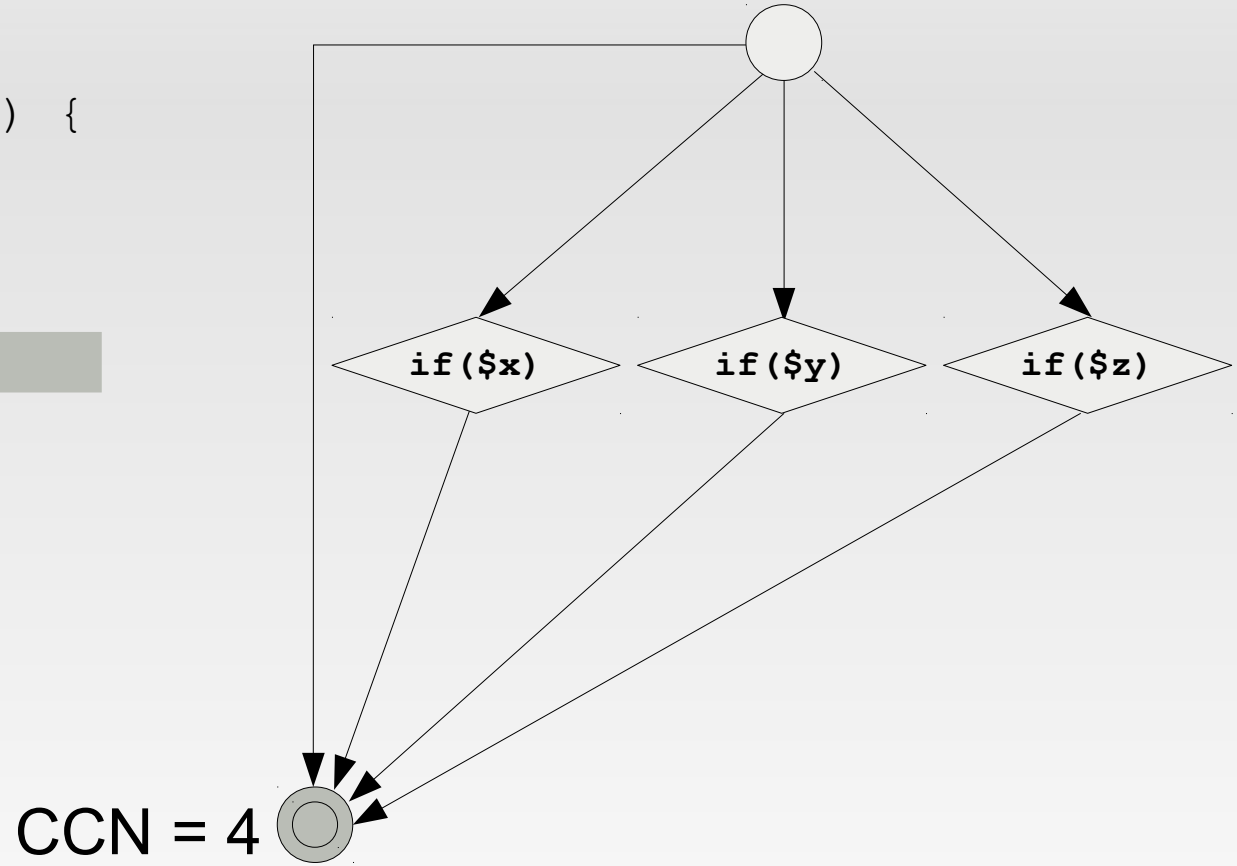
```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



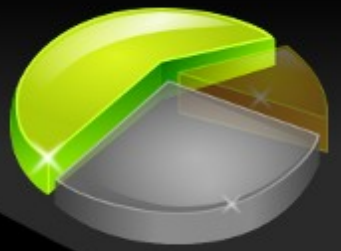
Cyclomatic Complexity



```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



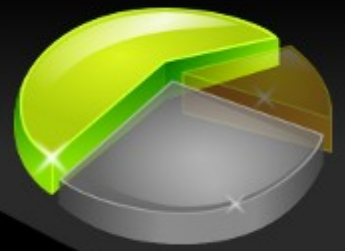
NPath Complexity



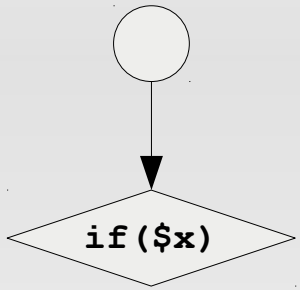
```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```

NPath = 0

NPath Complexity

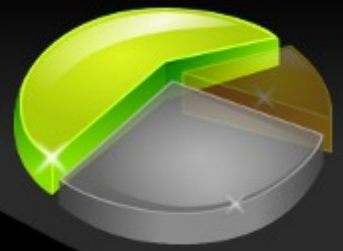


```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```

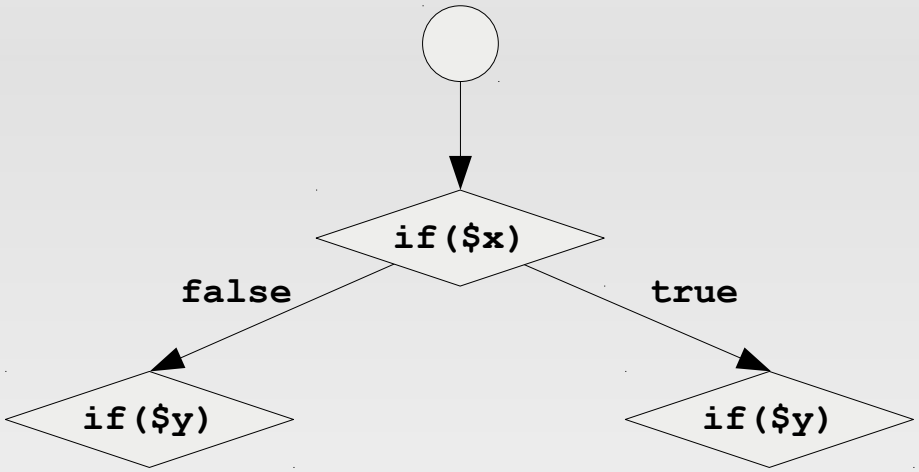


NPath = 0

NPath Complexity



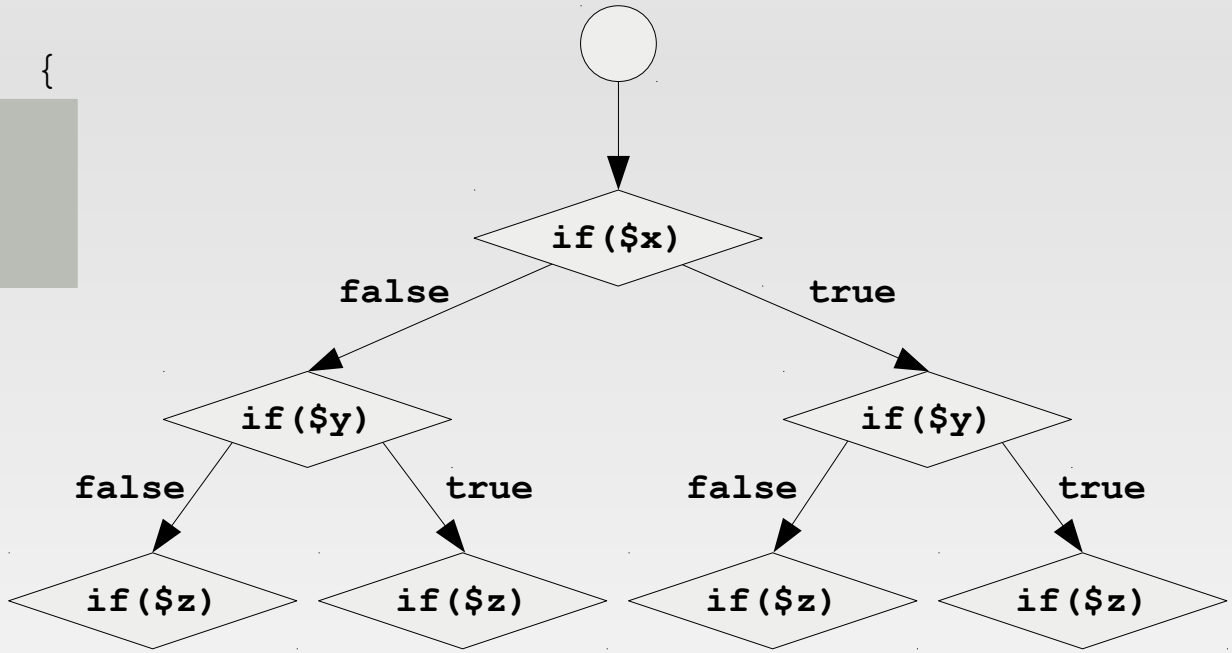
```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



NPath = 0

NPath Complexity

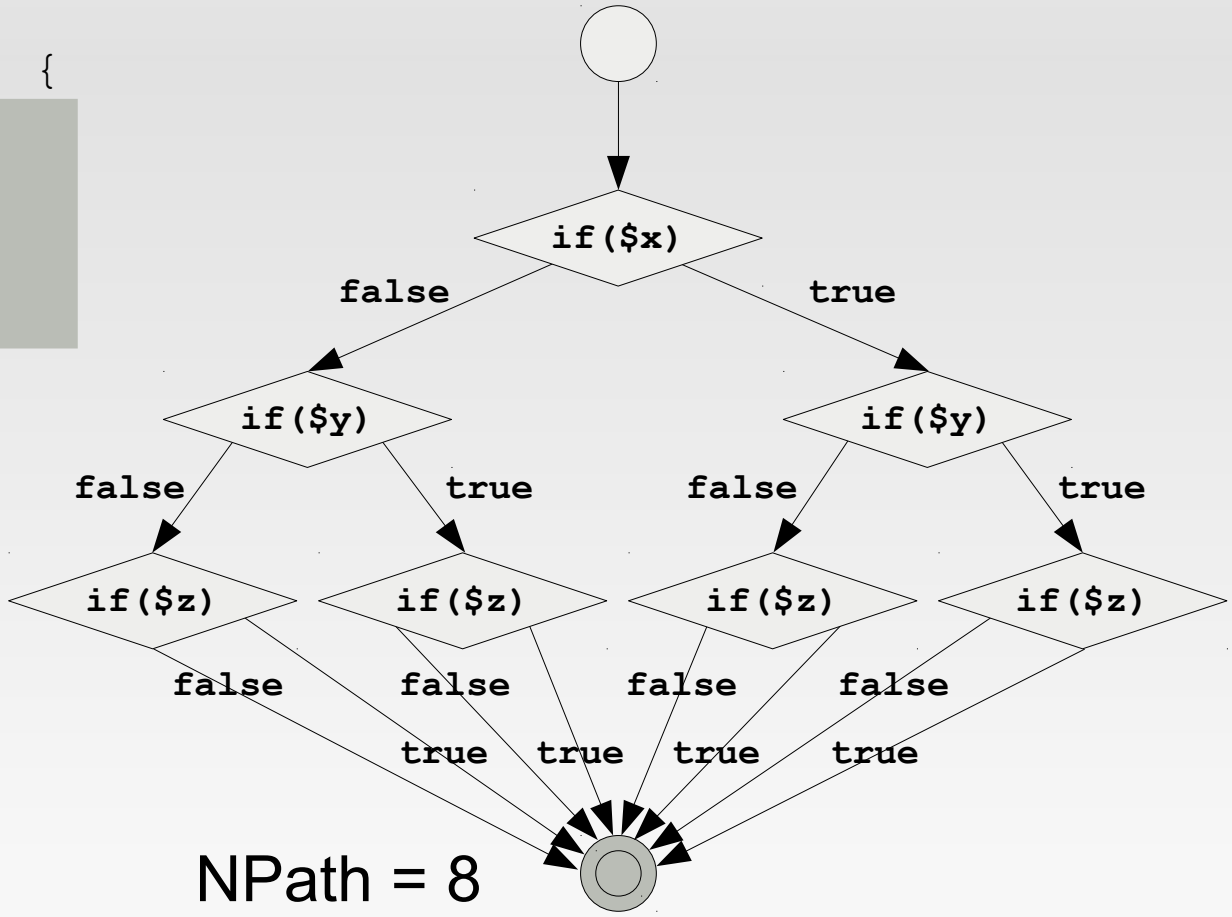
```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



NPath = 0

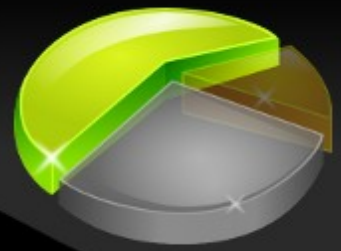
NPath Complexity

```php
<?php
class Foo {
    public function foo() {
        if ($x) { }
        if ($y) { }
        if ($z) { }
        return $x;
    }
}
```



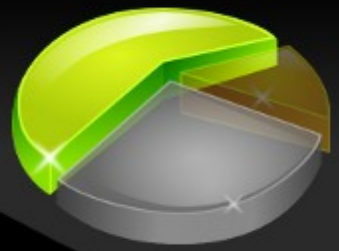
NPath = 8

Grenzwerte



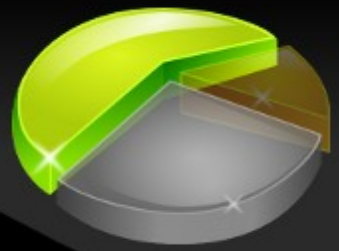
- Zahlen allein sagen noch nichts aus
  - Oder was bedeuten die Werte 4 und 8 ?
- Für eine Beurteilung benötigt man Grenzwerte
  - Cyclomatic Complexity
    - 1-4: low, 5-7: medium, 8-10: high, 11+: hell
  - NPath Complexity
    - 200: critical mass
- Grenzwerte sind immer Ermessenssache

Beispiel



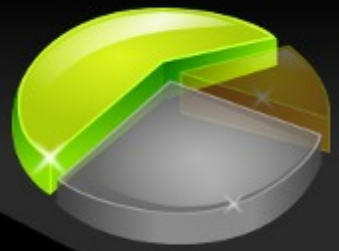
...

Metriken kombinieren



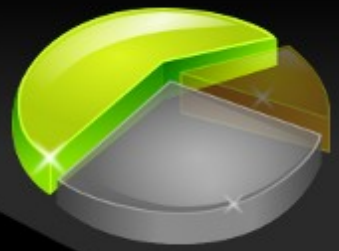
- Die Kombination von Metriken erlaubt einen sehr tiefen Einblick in ein System.
  - LOC: 300; CCN: 42; NOC: 5; NOM: 15
    - CCN / LOC = 0,14
      - Jede sechste Zeile eine Kontrollstruktur
    - LOC / NOC = 60
      - Primär prozedural oder große Klassen
    - LOC / NOM = 20
      - Große Methoden/Funktionen oder prozedural
    - CCN / NOM = 2,8
      - Übermäßig komplexe Methoden/Funktionen

Agenda



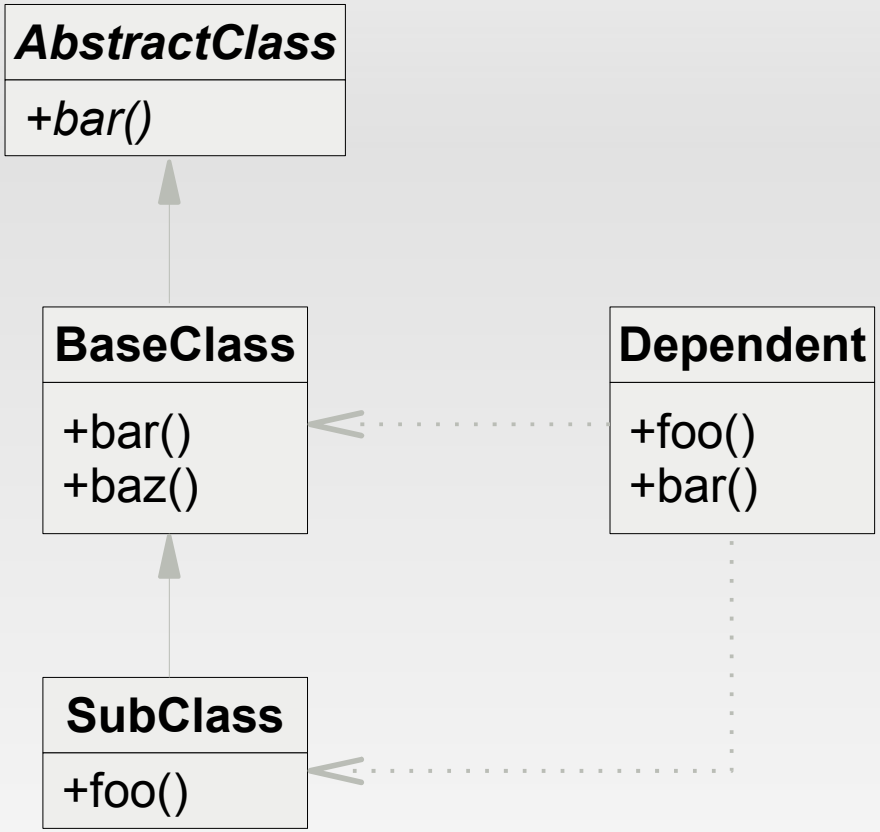
- Umfang
- Komplexität
- Klassische OO-Metriken
- Kopplung, Abstraktion & Instabilität
- Code Rank

Chidamber & Kemerer



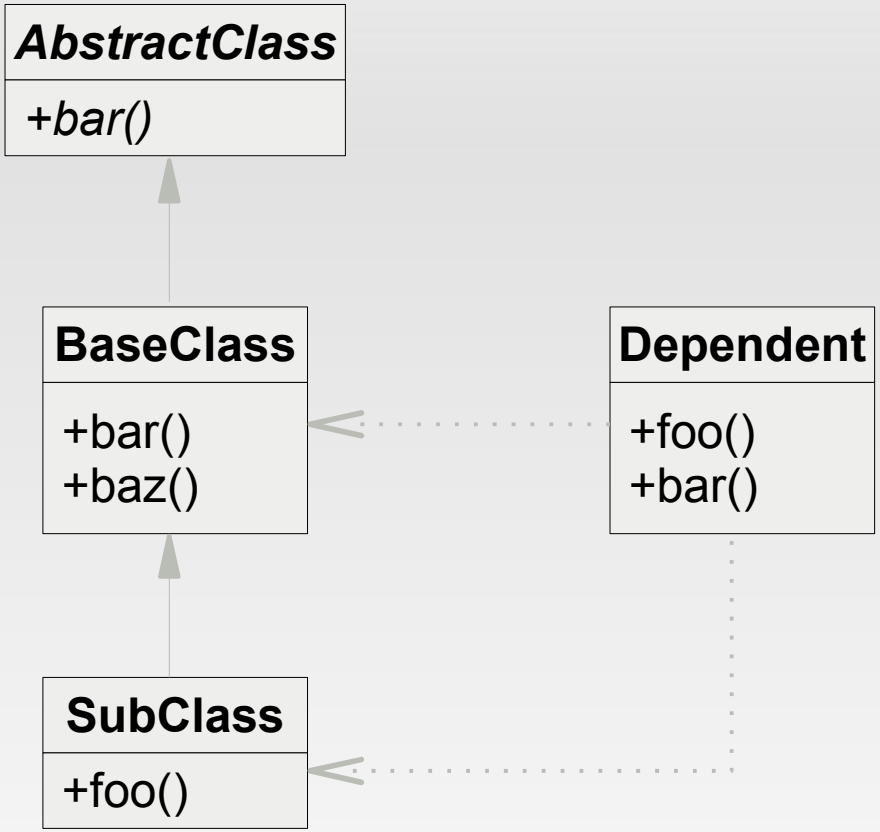
- Weighted Methods per Class (WMC)
  - Summe der Komplexität aller Methoden
  - Grenzwert 20 - 50
- Number Of Children (NOC)
  - Anzahl der direkten Ableitungen
  - Falsch gewählte Abstraktion
- Depth of Inheritance Tree (DIT)
  - Vererbungsstrukturen erhöhen die Komplexität
  - Grenzwert <= 5

OO-Metriken



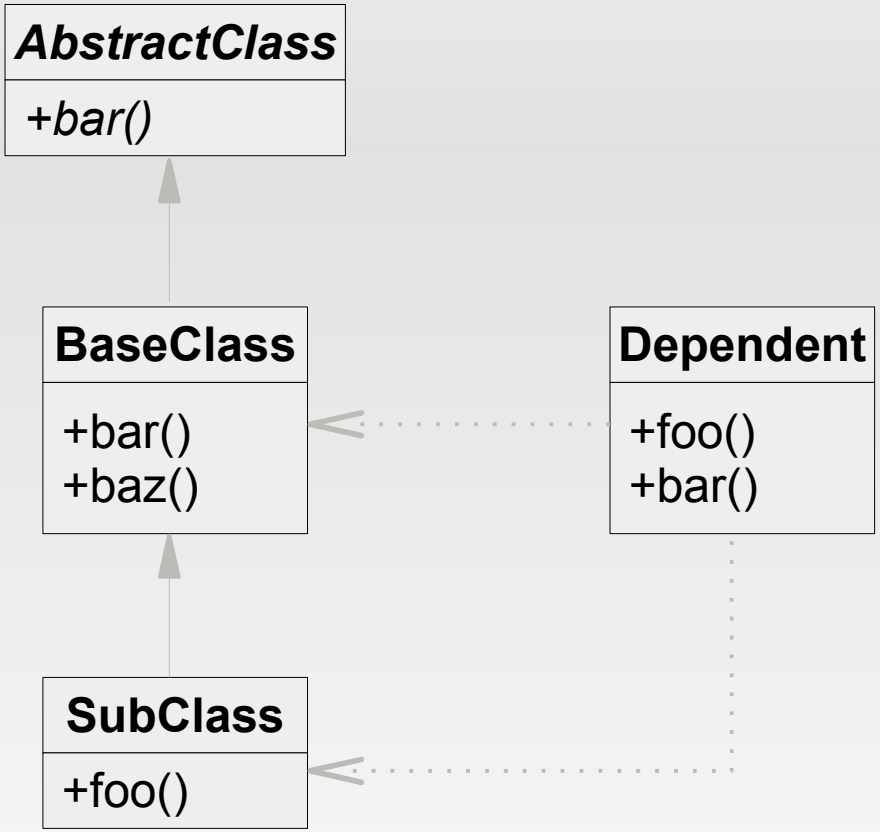
- *AbstractClass*
  - WMC = ; DIT =
- BaseClass
  - WMC = ; DIT =
- SubClass
  - WMC = ; DIT =
- Dependent
  - WMC = ; DIT =

OO-Metriken



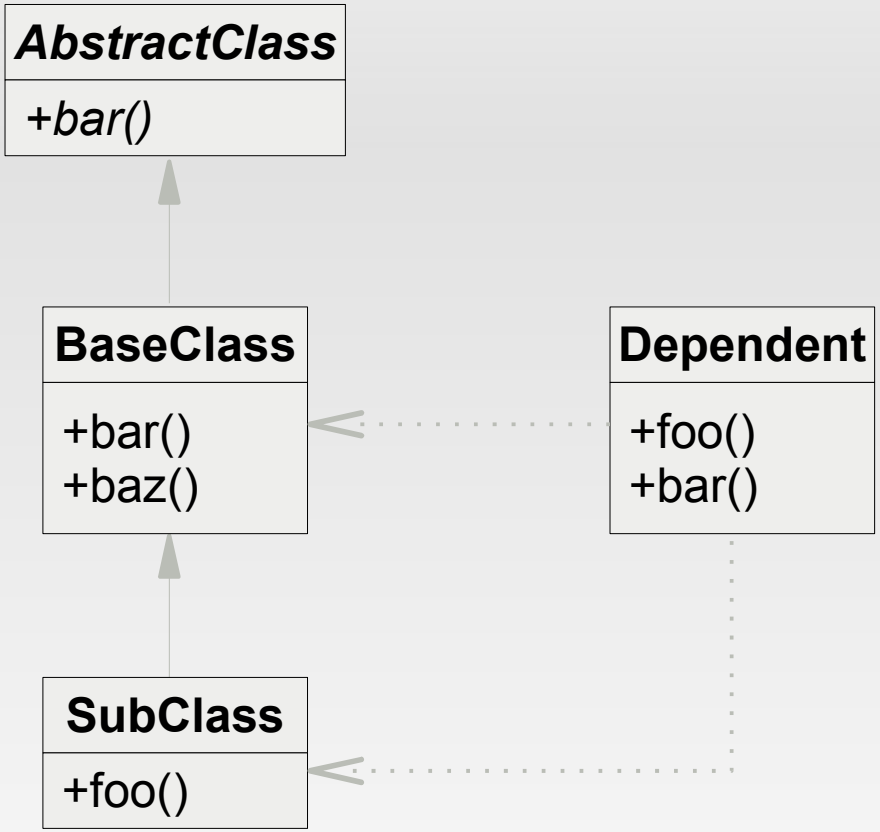
- *AbstractClass*
  - WMC = 0; DIT =
- BaseClass
  - WMC = ; DIT =
- SubClass
  - WMC = ; DIT =
- Dependent
  - WMC = ; DIT =

OO-Metriken



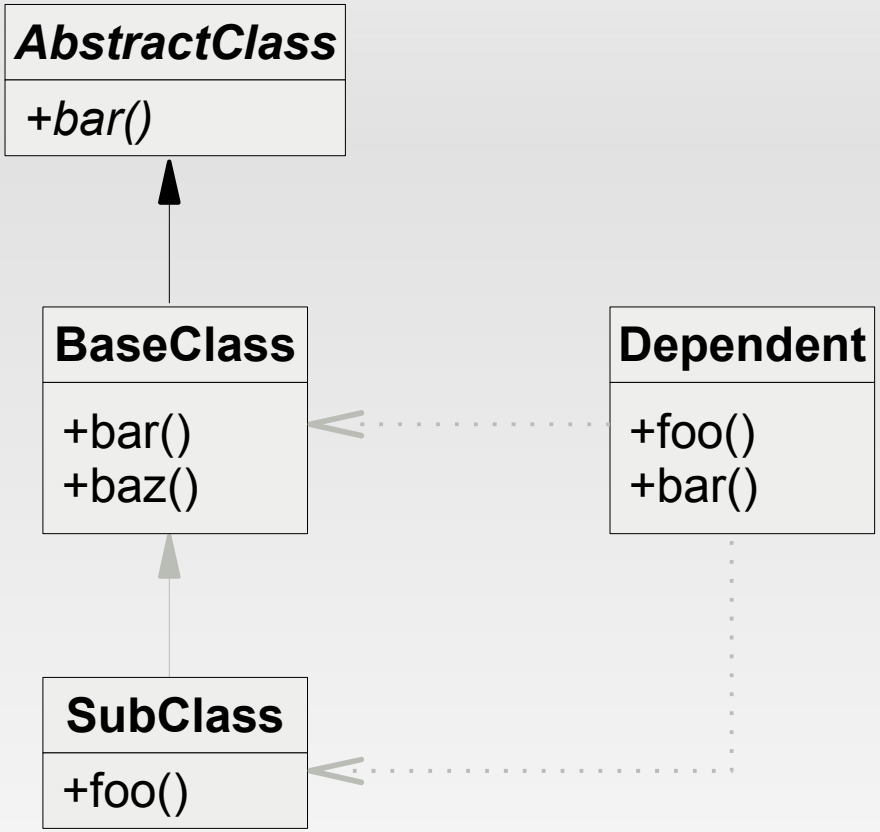
- *AbstractClass*
  - WMC = 0; DIT = 0
- BaseClass
  - WMC = ; DIT =
- SubClass
  - WMC = ; DIT =
- Dependent
  - WMC = ; DIT =

OO-Metriken



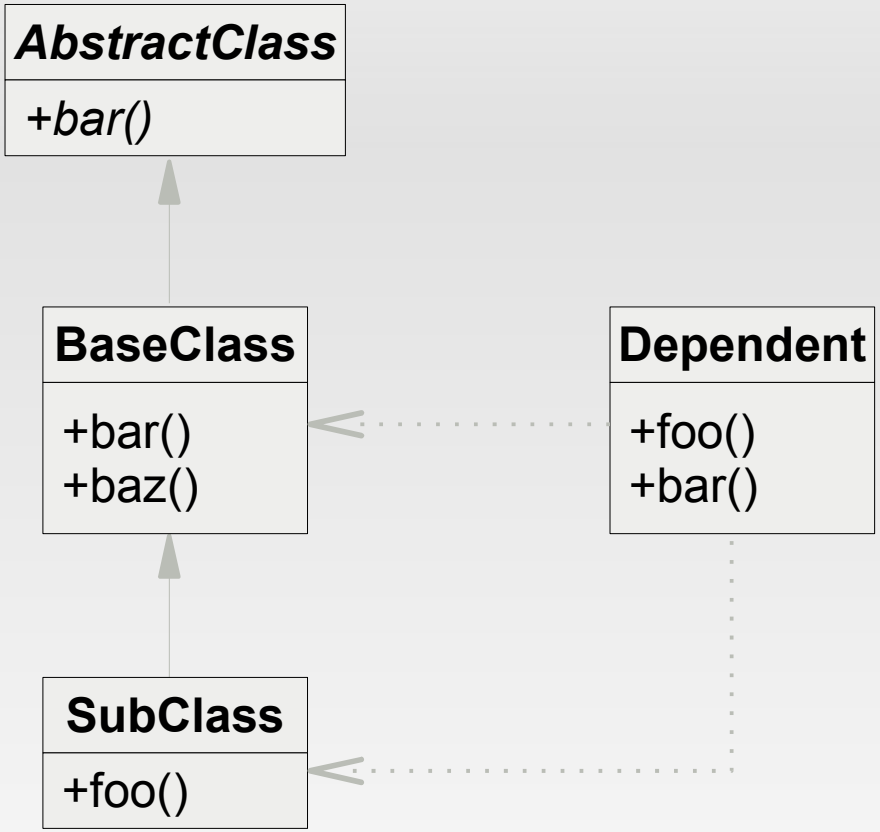
- *AbstractClass*
  - WMC = 0; DIT = 0
- BaseClass
  - WMC = 2; DIT =
- SubClass
  - WMC = ; DIT =
- Dependent
  - WMC = ; DIT =

OO-Metriken



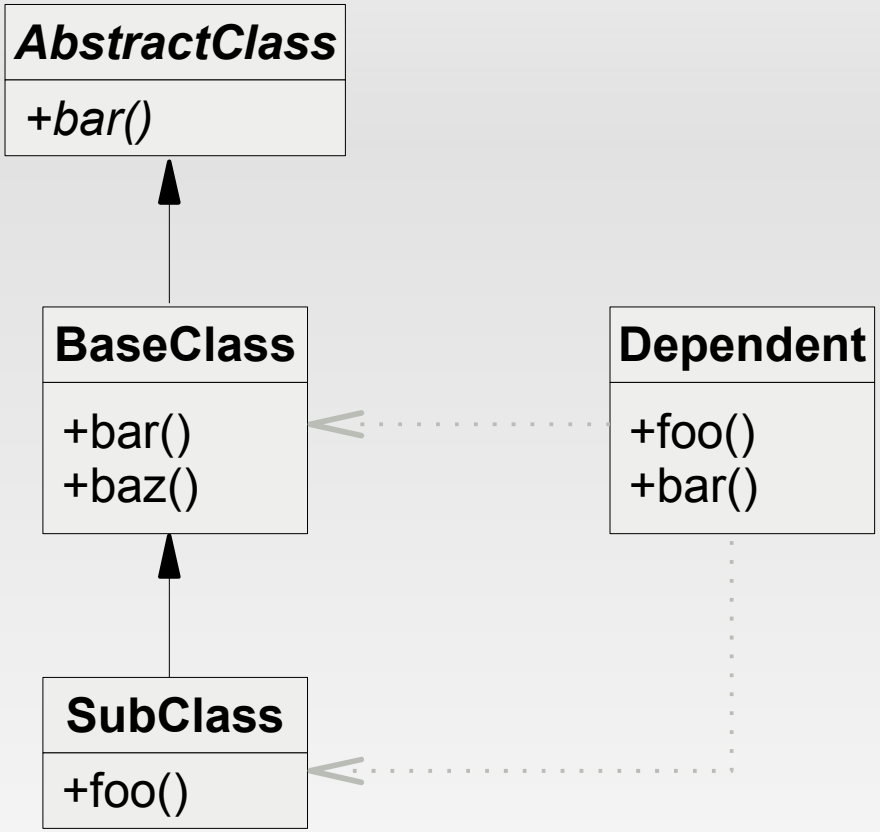
- *AbstractClass*
  - WMC = 0; DIT = 0
- BaseClass
  - WMC = 2; DIT = 1
- SubClass
  - WMC = ; DIT =
- Dependent
  - WMC = ; DIT =

OO-Metriken



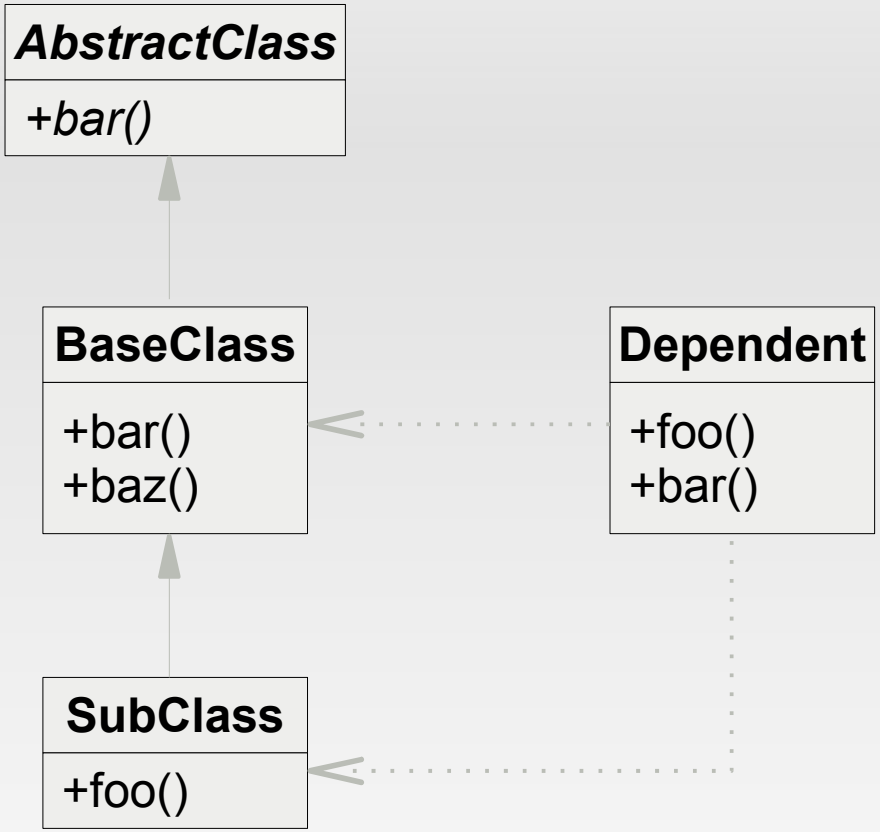
- *AbstractClass*
  - WMC = 0; DIT = 0
- BaseClass
  - WMC = 2; DIT = 1
- SubClass
  - WMC = 1; DIT =
- Dependent
  - WMC = ; DIT =

OO-Metriken



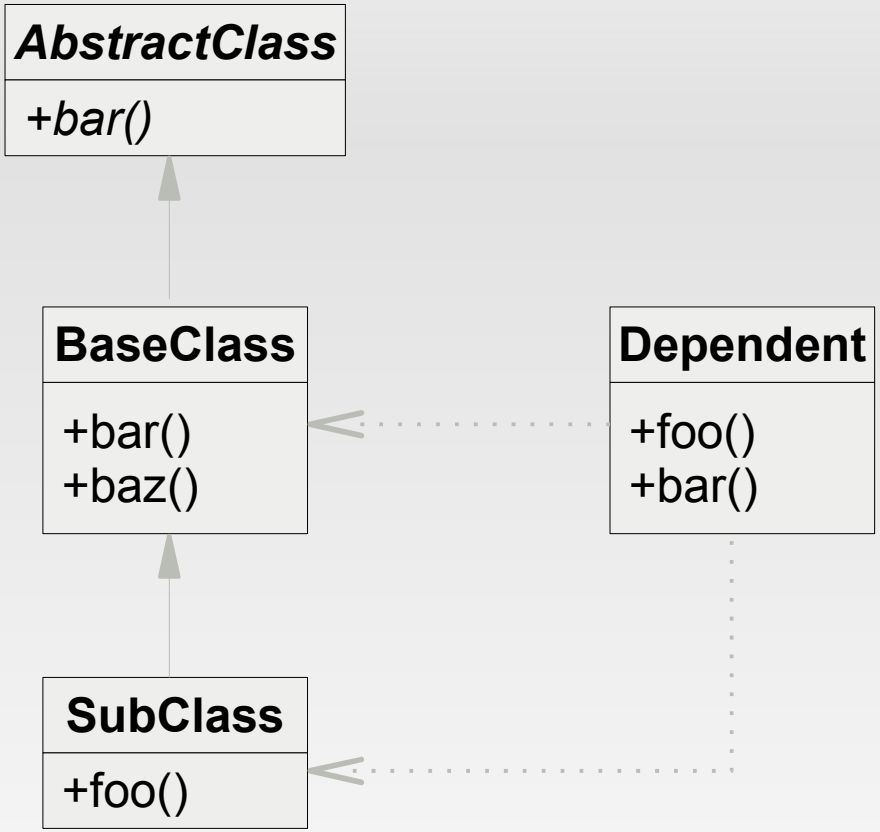
- *AbstractClass*
  - WMC = 0; DIT = 0
- BaseClass
  - WMC = 2; DIT = 1
- SubClass
  - WMC = 1; DIT = 2
- Dependent
  - WMC = ; DIT =

OO-Metriken



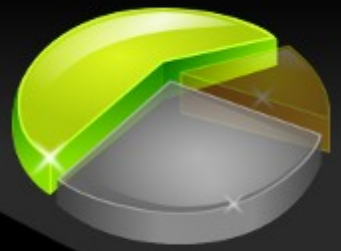
- *AbstractClass*
  - WMC = 0; DIT = 0
- BaseClass
  - WMC = 2; DIT = 1
- SubClass
  - WMC = 1; DIT = 2
- Dependent
  - WMC = 2; DIT =

OO-Metriken



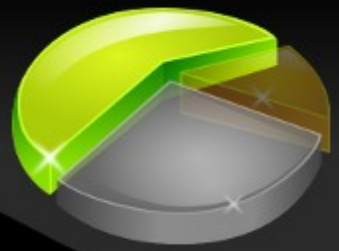
- *AbstractClass*
  - WMC = 0; DIT = 0
- BaseClass
  - WMC = 2; DIT = 1
- SubClass
  - WMC = 1; DIT = 2
- Dependent
  - WMC = 2; DIT = 0

Beispiel



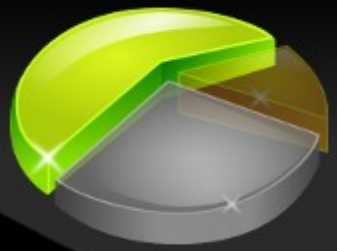
...

Agenda



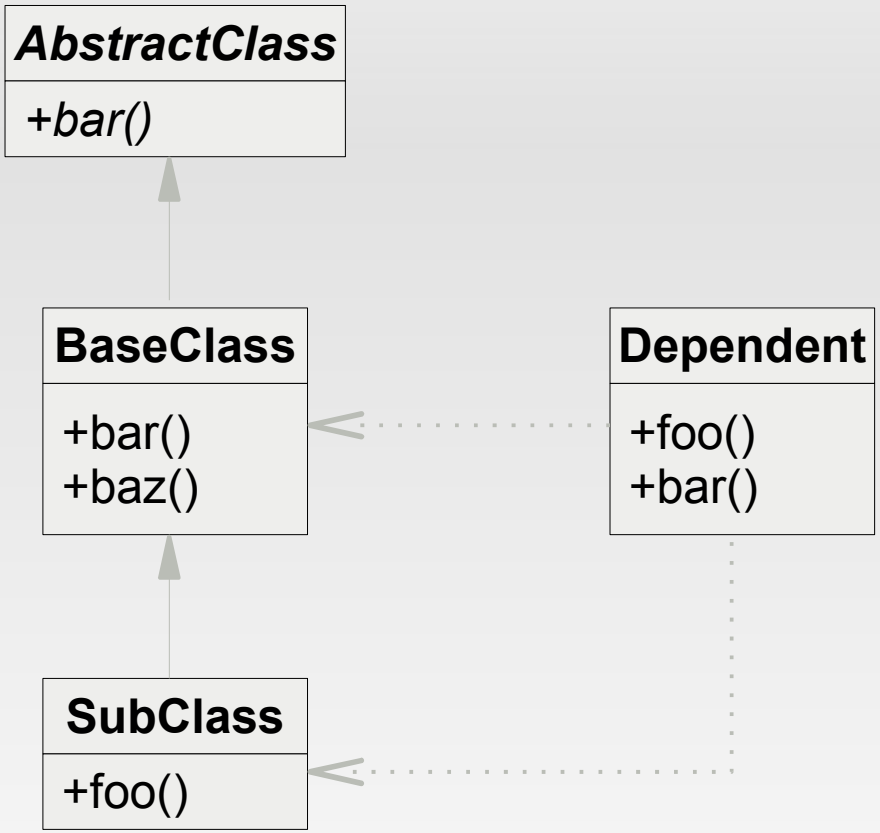
- Umfang
- Komplexität
- Klassische OO-Metriken
- Kopplung, Abstraktion & Instabilität
- Code Rank

Afferent- & Efferent-Coupling



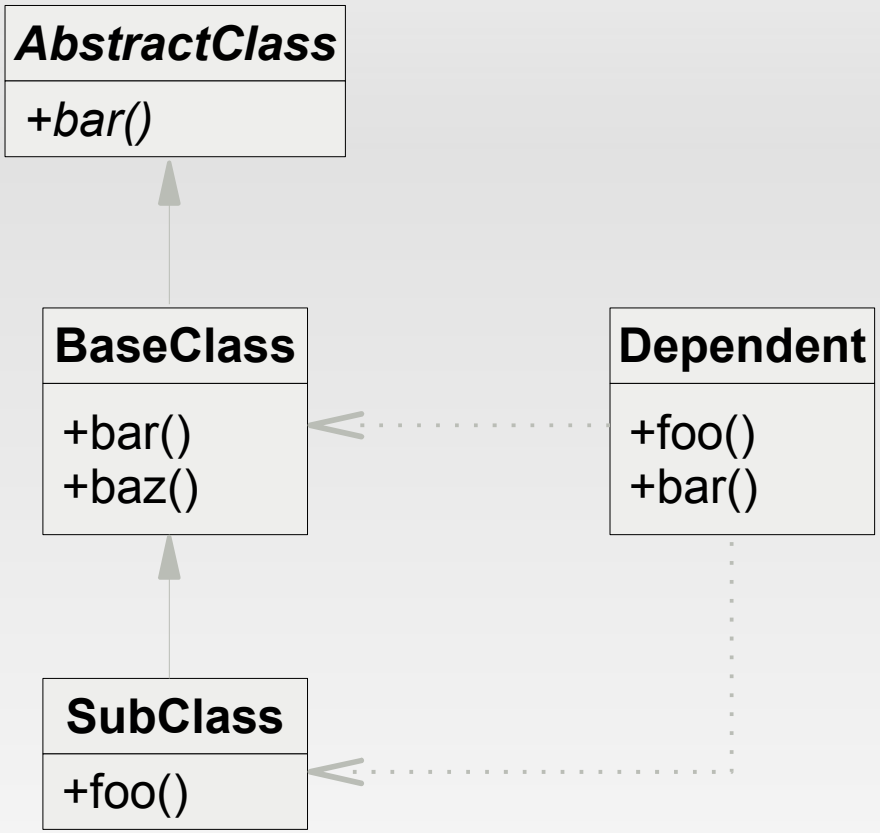
- Afferent Coupling (Ca)
  - Eingehende Abhängigkeiten anderer Komponenten
  - Großer Einfluss aus die Stabilität des Systems
- Efferent Coupling (Ce)
  - Ausgehende Abhängigkeiten:
    - Vererbung, Parameter, Exceptions, Allokationen
  - Abhängigkeit von Stabilität anderer Komponenten

Afferent- & Efferent-Coupling



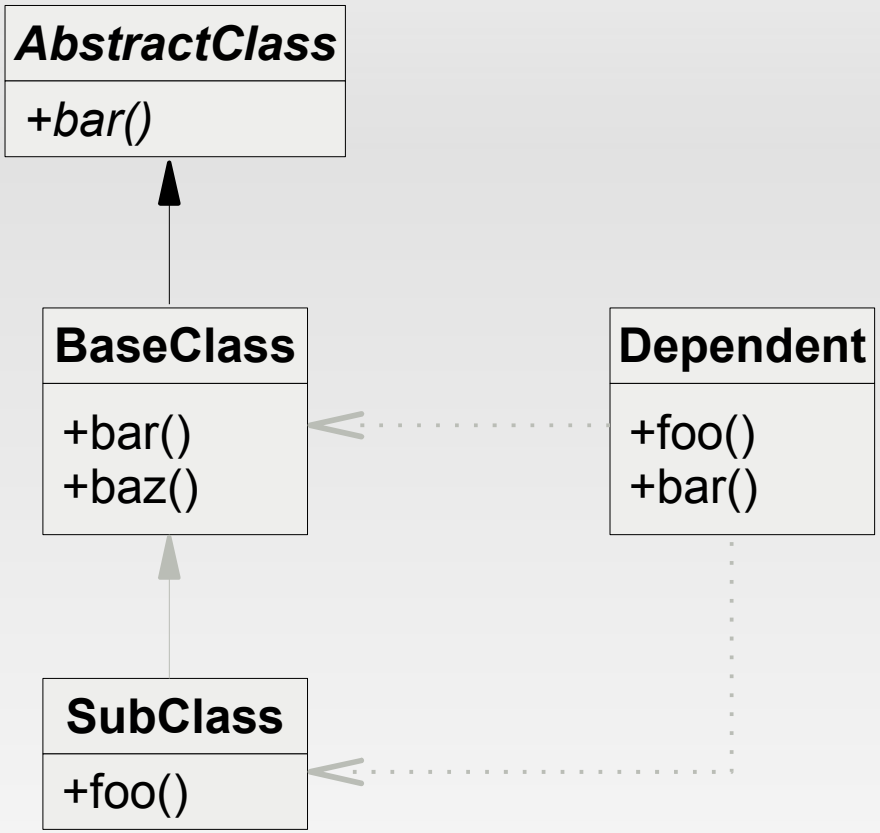
- *AbstractClass*
  - Ce = ; Ca =
- BaseClass
  - Ce = ; Ca =
- SubClass
  - Ce = ; Ca =
- Dependent
  - Ce = ; Ca =

Afferent- & Efferent-Coupling



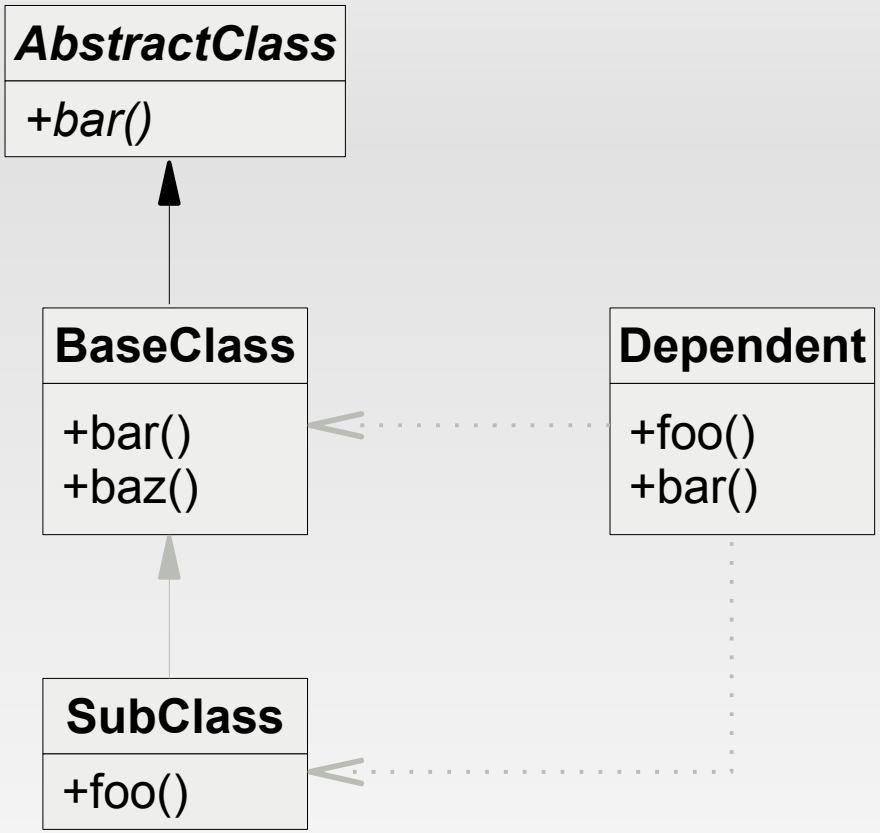
- *AbstractClass*
  - Ce = 0; Ca =
- BaseClass
  - Ce = ; Ca =
- SubClass
  - Ce = ; Ca =
- Dependent
  - Ce = ; Ca =

Afferent- & Efferent-Coupling



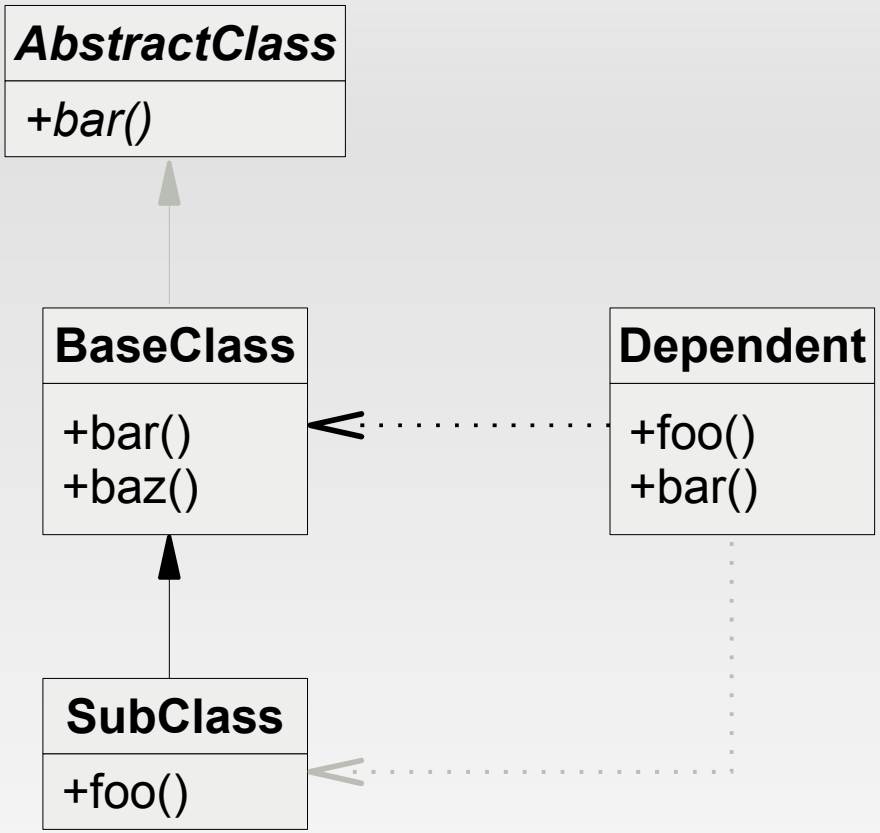
- *AbstractClass*
  - Ce = 0; Ca = 1
- BaseClass
  - Ce = ; Ca =
- SubClass
  - Ce = ; Ca =
- Dependent
  - Ce = ; Ca =

Afferent- & Efferent-Coupling



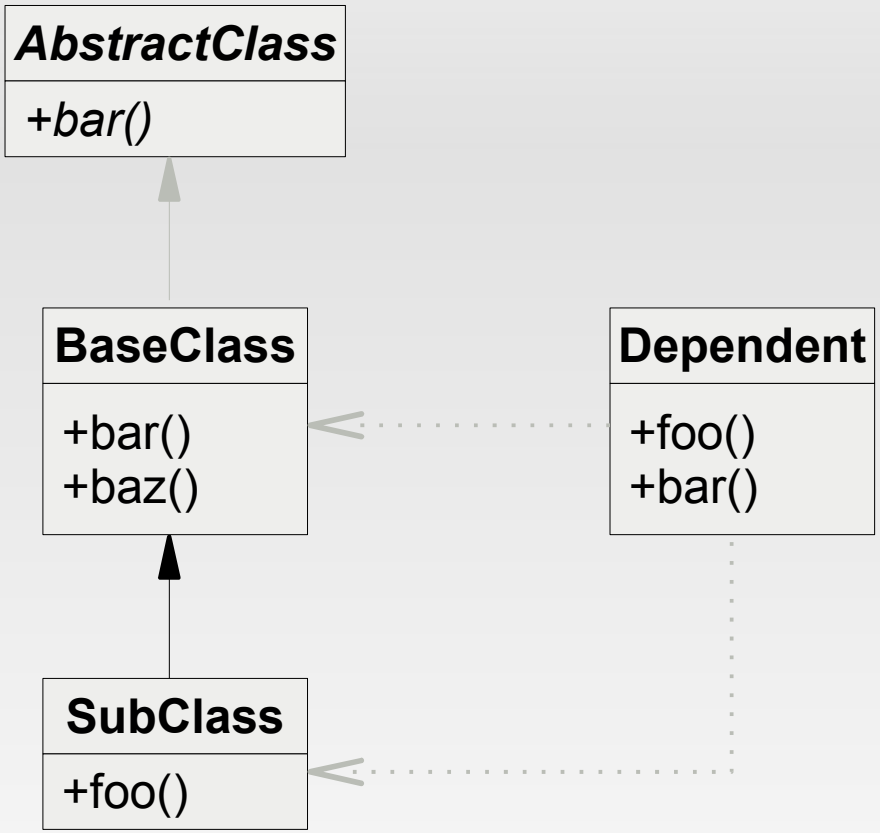
- *AbstractClass*
  - Ce = 0; Ca = 1
- BaseClass
  - Ce = 1; Ca =
- SubClass
  - Ce = ; Ca =
- Dependent
  - Ce = ; Ca =

Afferent- & Efferent-Coupling



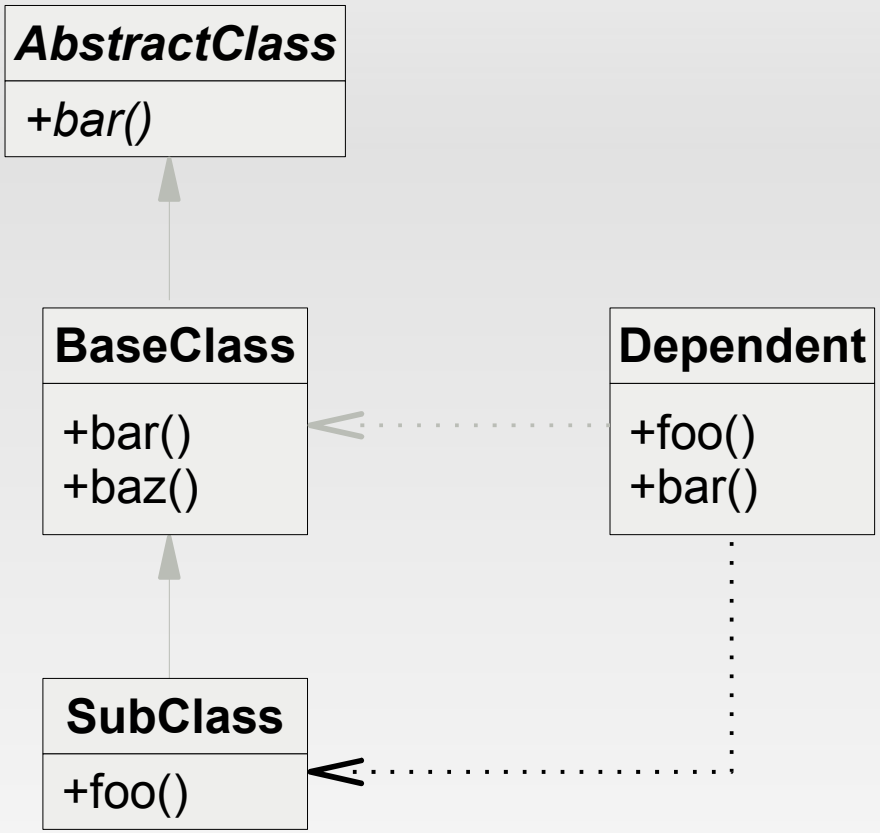
- *AbstractClass*
  - Ce = 0; Ca = 1
- BaseClass
  - Ce = 1; Ca = 2
- SubClass
  - Ce = ; Ca =
- Dependent
  - Ce = ; Ca =

Afferent- & Efferent-Coupling



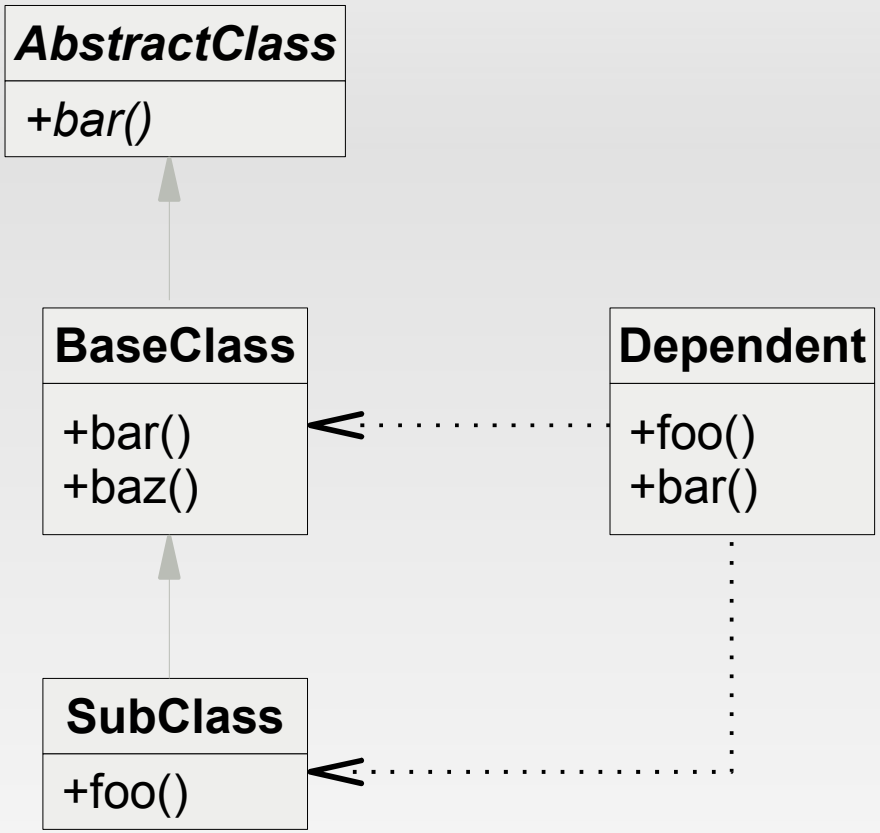
- *AbstractClass*
  - Ce = 0; Ca = 1
- BaseClass
  - Ce = 1; Ca = 2
- SubClass
  - Ce = 1; Ca =
- Dependent
  - Ce = ; Ca =

Afferent- & Efferent-Coupling



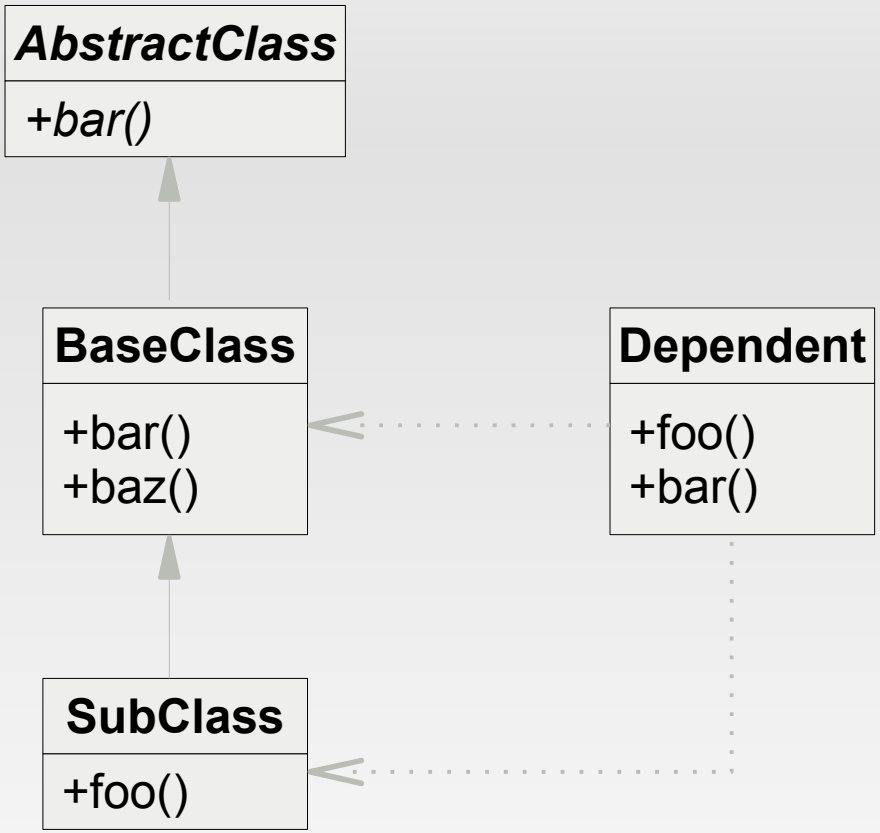
- *AbstractClass*
  - Ce = 0; Ca = 1
- BaseClass
  - Ce = 1; Ca = 2
- SubClass
  - Ce = 1; Ca = 1
- Dependent
  - Ce = ; Ca =

Afferent- & Efferent-Coupling



- *AbstractClass*
  - Ce = 0; Ca = 1
- BaseClass
  - Ce = 1; Ca = 2
- SubClass
  - Ce = 1; Ca = 1
- Dependent
  - Ce = 2; Ca =

Afferent- & Efferent-Coupling

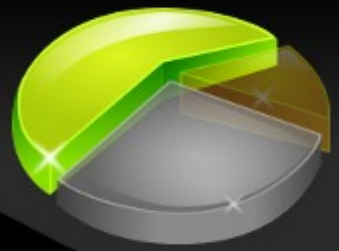


- *AbstractClass*
  - Ce = 0; Ca = 1
- BaseClass
  - Ce = 1; Ca = 2
- SubClass
  - Ce = 1; Ca = 1
- Dependent
  - Ce = 2; Ca = 0

Abstraction, Instability&Distance

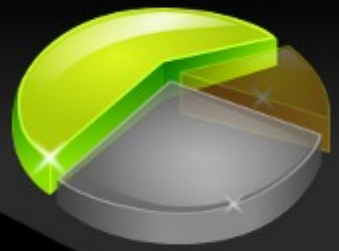
- Modell für eine ausgewogene Architektur
  - Basiert auf Ce, Ca, abstrakten(AC) und konkreten(CC) Softwareartefakten
  - Abstraction
    - $A = AC / (AC + CC)$
  - Instability
    - $I = Ce / (Ce + Ca)$
  - Distance
    - $D = |A + I| - 1$

Beispiel



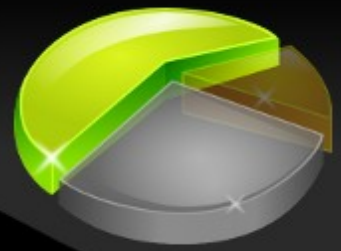
...

Agenda



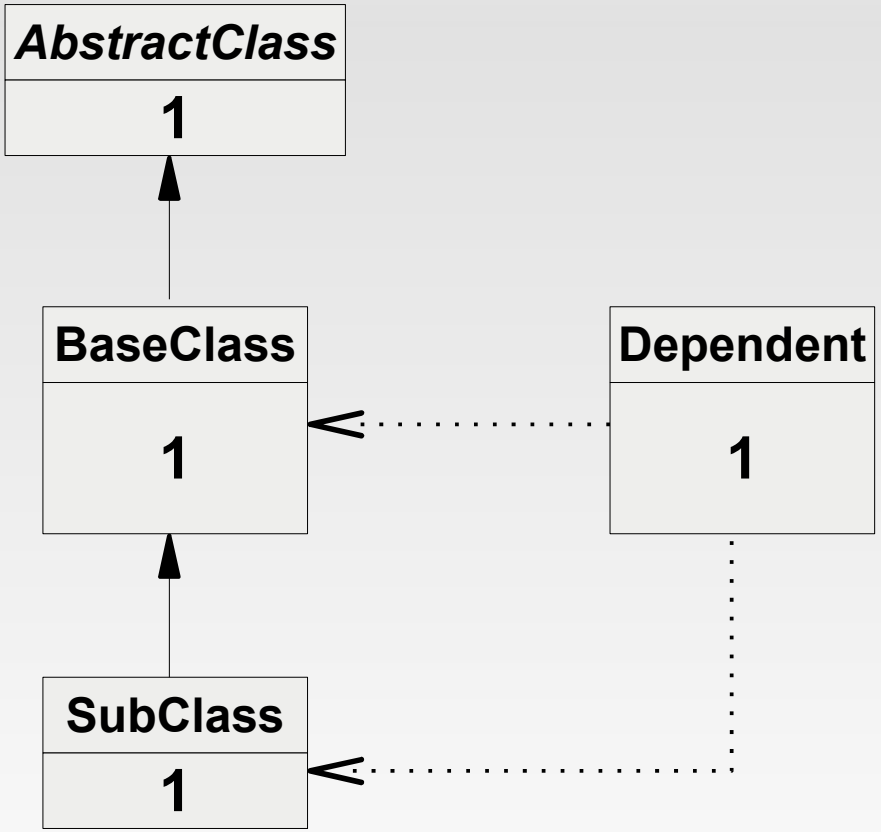
- Umfang
- Komplexität
- Klassische OO-Metriken
- Kopplung, Abstraktion & Instabilität
- CodeRank

CodeRank



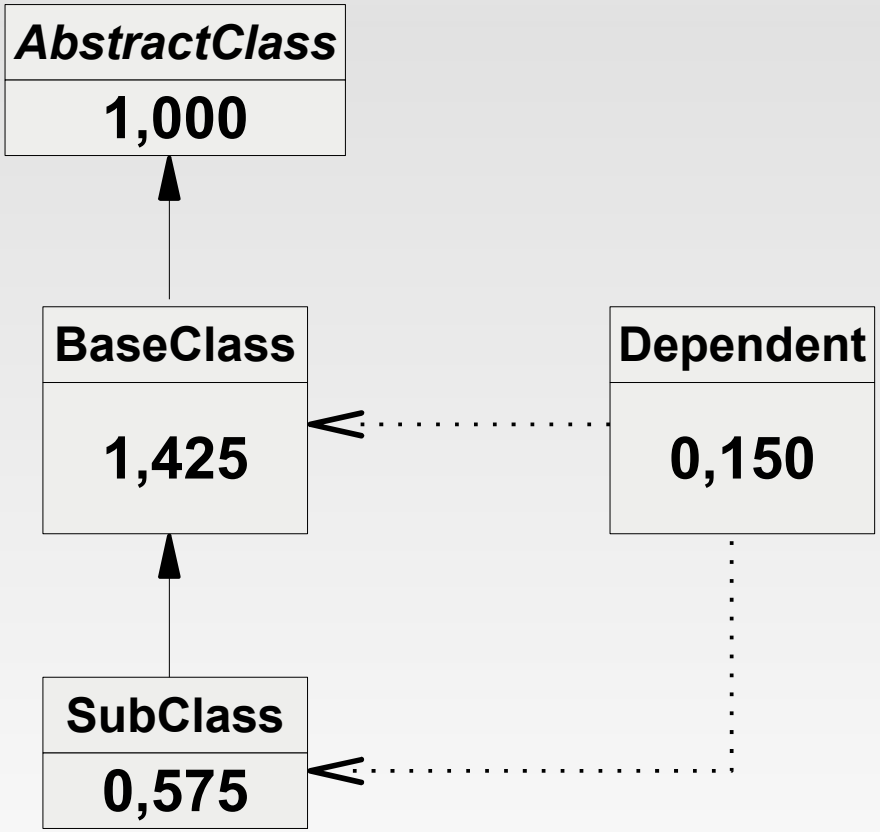
- Basiert auf dem Google PageRank
- Software wird auf eine Graphen abgebildet
  - Knoten(π) je Softwareartefakt
    - Pakete, Klassen, Methoden
  - Kanten(ρ) für jede Beziehung
    - Vererbung, Aufrufe, Parameter, Exceptions
- Für den CodeRank gilt:
  - $CR(\pi_i) = \sum_r((1 - d) + d \sum_r (CR(\pi_r) / \rho_r))$

CodeRank



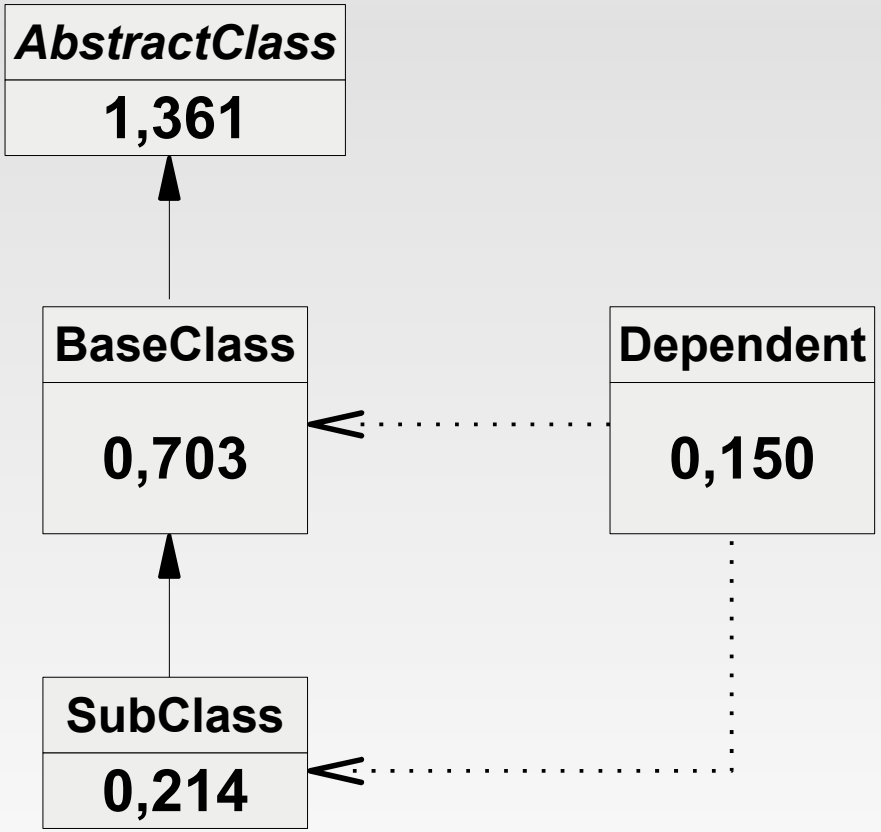
0

CodeRank


AbstractClass
1,000
BaseClass
1,425
Dependent
0,150
SubClass
0,575

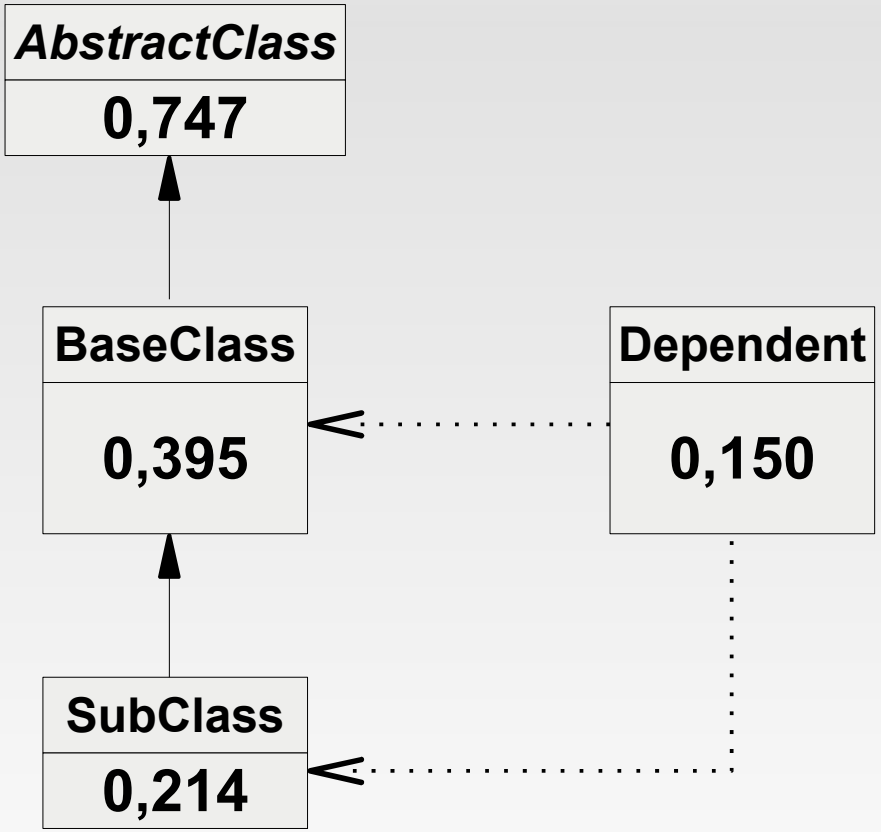
1

CodeRank


AbstractClass
1,361
BaseClass
0,703
Dependent
0,150
SubClass
0,214

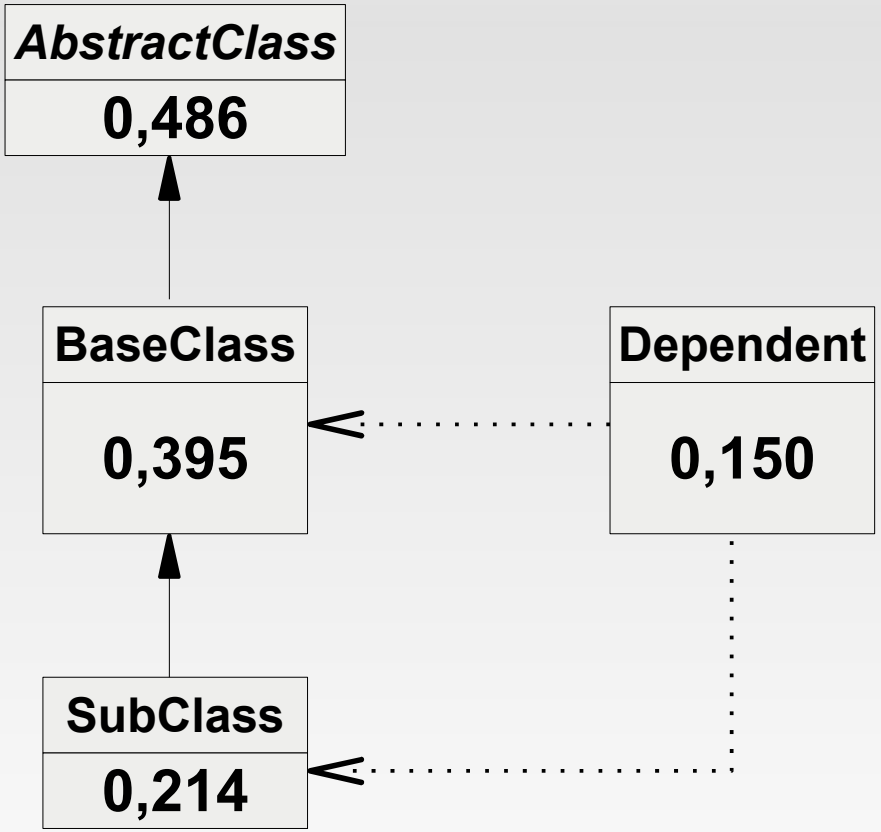
2

CodeRank


AbstractClass
0,747
BaseClass
0,395
Dependent
0,150
SubClass
0,214

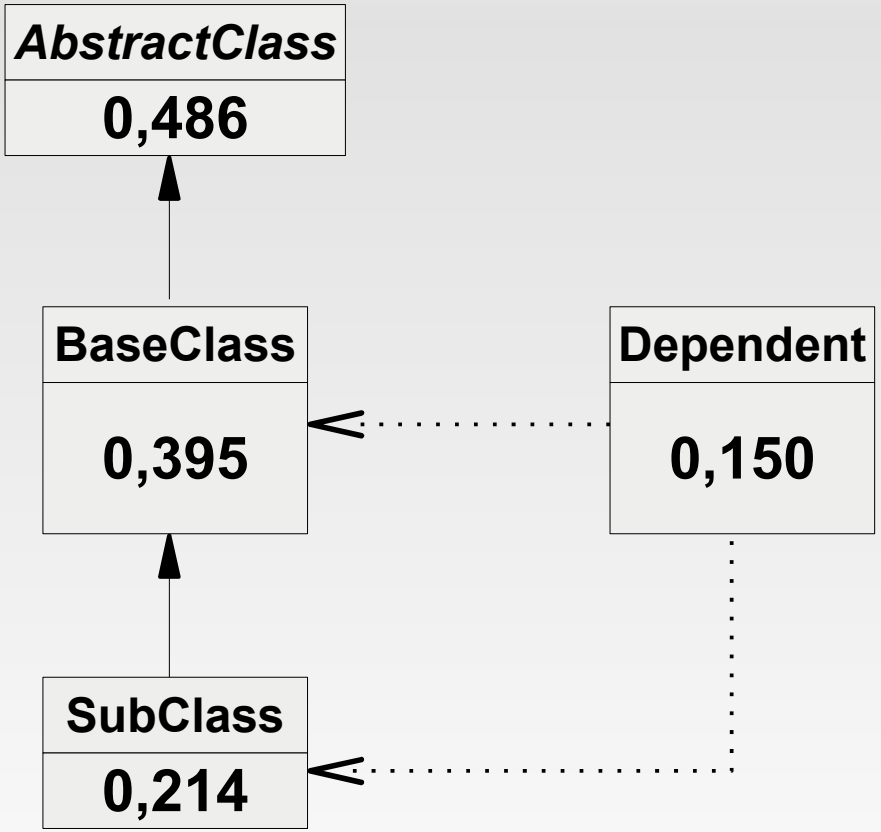
3

CodeRank


AbstractClass
0,486
BaseClass
0,395
Dependent
0,150
SubClass
0,214

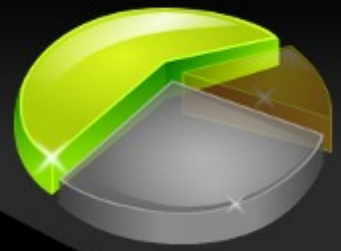
4

CodeRank


AbstractClass
0,486
BaseClass
0,395
Dependent
0,150
SubClass
0,214

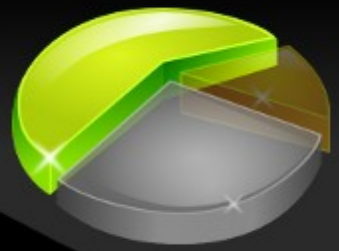
5

CodeRank

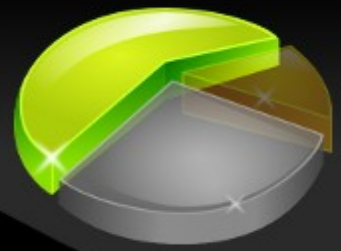


- Mit dem CodeRank können auch indirekte Abhängigkeiten bewertet werden.
- Logik mit Einfluss auf das gesamte System kann schnell gefunden werden
- Äquivalent der Reverse CodeRank
  - Kann mit dem selben Algorithmus berechnet werden
  - Gibt Aufschluss über stark abhängige Komponenten

Metriken sind ...



- ... keine Magie, sondern einfache Maßzahlen
- ... nutzlos, ohne Grenz- oder Referenzwerte
- ... skalierbar, wachsen mit der Projektgröße
- ... automatisierbar und reproduzierbar
- ... objektiv, da Software gestützt
- ... interpretierbar, abhängig vom Betrachter



?

Contact:
- Manuel Pichler: http://manuel-pichler.de @manuelp
- Kore Nordmann: http://kore-nordmann.de @koredn