12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

V33
Evolutionär

Evolvierbarkeit von Code

Golo Roden

Evolvierbaren Code schreiben

Golo Roden

www.goloroden.de
www.des-eisbaeren-blog.de

Über mich

- Wissensvermittler und Technologieberater
  - .NET, Codequalität und agile Methoden
  - MVP für C#, zweifacher MCP und CCD
- Autor, Sprecher und Trainer
  - dotnetpro, heise Developer
  - prio.conference, dotnetpro Powerday
- Kontakt
  - www.goloroden.de
  - www.des-eisbaeren-blog.de

Unangekündigter Test

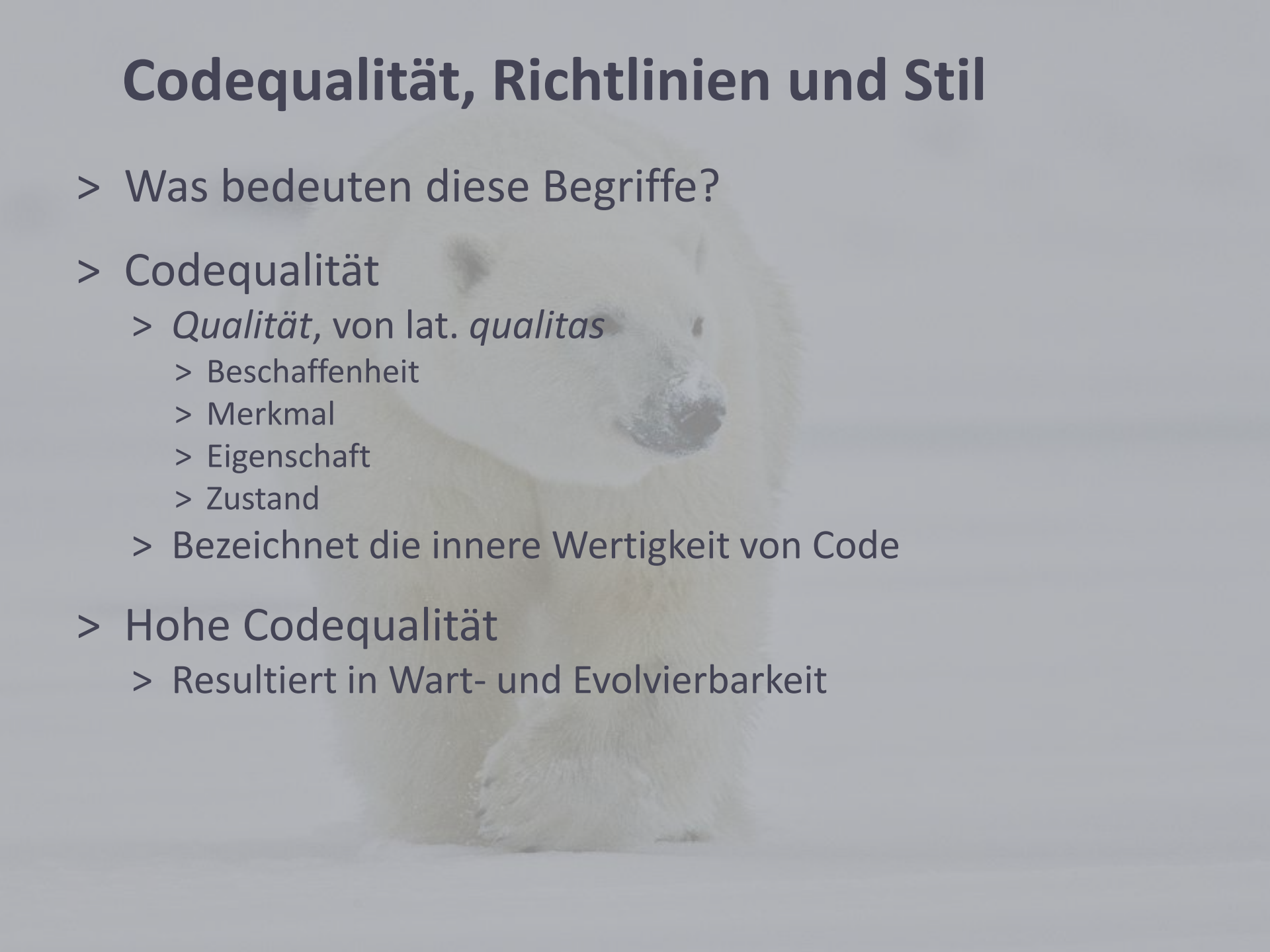
- Was bewirkt dieser Code?

```
#include <stdio.h>
main(t,_,a)char *a;{return!0<t?t<3?main(-79,-13,a+main(-
87,1-_,
main(-86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-
27+t,a)&&t==2?_<13?
main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#
{l,+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){n
l]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw'
iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i;
:{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/
') }+}{rl#'{n' ')# \
}'+}##(!!/")
:t<-50?_==*a?putchar(31[a]):main(-
65,_,a+1):main((*a=='/')+t,_,a+1)
  :0<t?main(2,2,"%s"):*a=='/'||main(0,main(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m
.vpbks,fxntdCeghiry"),a+1);}
```

Lösungsvarianten

- Seriöse Varianten
  - Nachdenken
  - Debuggen
- Weniger seriöse Variante
  - Hoffen und Beten ;-)

Codequalität, Richtlinien und Stil



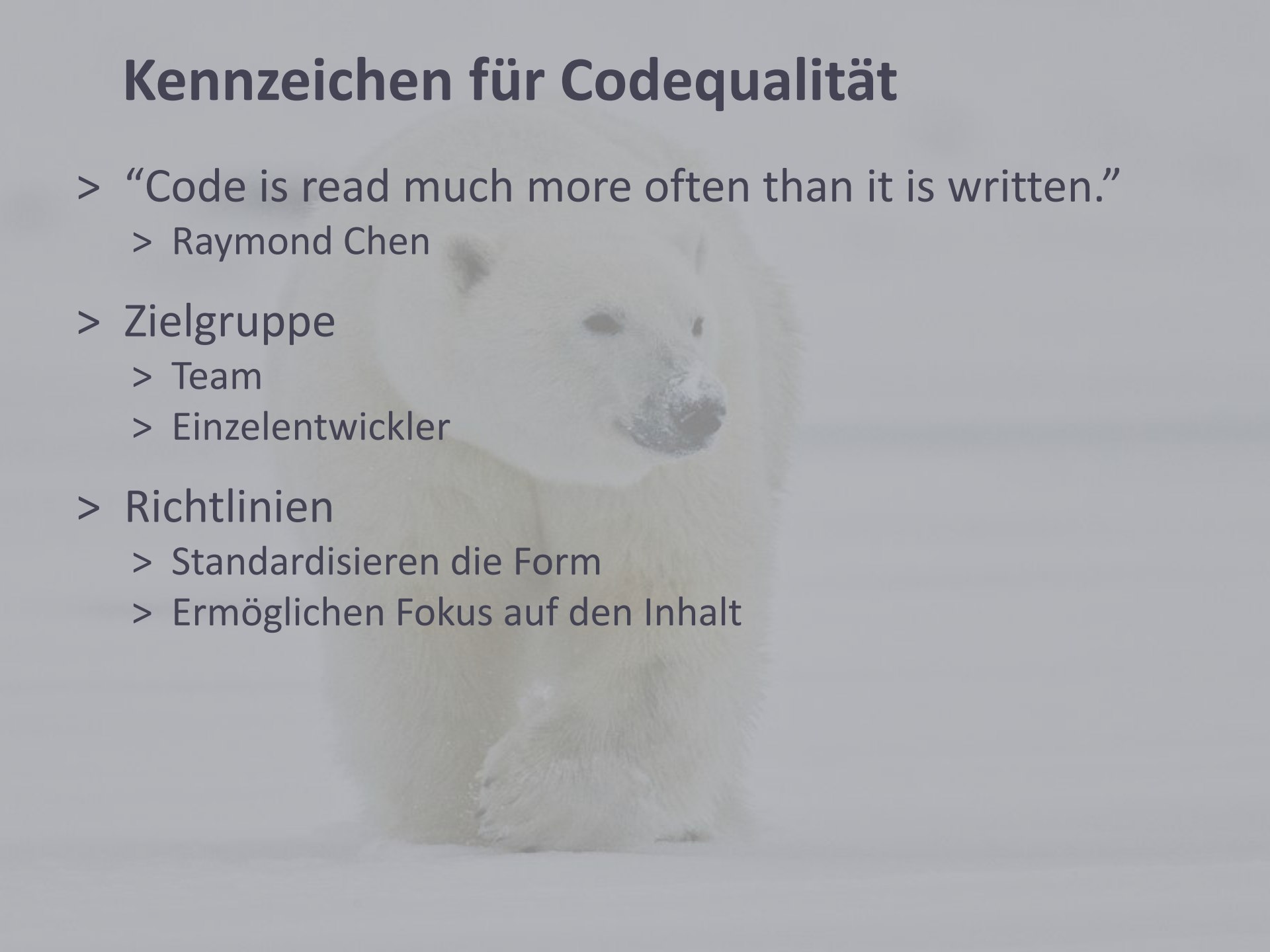
- Was bedeuten diese Begriffe?
- Codequalität
  - *Qualität*, von lat. *qualitas*
    - Beschaffenheit
    - Merkmal
    - Eigenschaft
    - Zustand
  - Bezeichnet die innere Wertigkeit von Code
- Hohe Codequalität
  - Resultiert in Wart- und Evolvierbarkeit

Kennzeichen für Codequalität

- “You can’t control what you can’t measure.”
  - Tom de Marco
- Kennzeichen für Qualität
  - Qualität orientiert sich in der Regel an Standards
  - Standards sind kontextabhängig
- Kennzeichen für Codequalität
  - Mathematisch elegant
  - Leicht zu verstehen
  - Kompakt und prägnant
- Standards für Code?

Kennzeichen für Codequalität

- “Code is read much more often than it is written.”
  - Raymond Chen
- Zielgruppe
  - Team
  - Einzelentwickler
- Richtlinien
  - Standardisieren die Form
  - Ermöglichen Fokus auf den Inhalt



Beispiele für Richtlinien

- Richtlinien entsprechen erprobten Vorgehensweisen
  - DO NOT throw exceptions from static constructors
    - … weil das Werfen einer Ausnahme in einem Typinitialisierer den Typ für die gesamte Anwendungsdomäne unbrauchbar macht.
  - DO prefer events over plain callbacks
    - … weil Ereignisse von mehr Entwicklern besser verstanden werden und Visual Studio zudem Unterstützung im Editor bietet.

Richtlinien vs Stil

- Richtlinien sind in der Regel weder “richtig” noch “falsch”
  - Zugriff auf private Felder – mit oder ohne this?
  - Verschiedene Argumente und Vorlieben
- Stil kennzeichnet …
  - … welche Richtlinien formuliert werden
  - … wie Richtlinien formuliert werden
- Richtlinien dürfen dem persönlichen Stil folgen
  - Unter gewissen Auflagen

Ausleben des eigenen Stils

- Die Freiheit des Einzelnen endet dort, wo die Freiheit des Anderen beginnt
- Auflagen
  - Konsistenz
    - Innerhalb des eigenen Codes
    - Innerhalb des Teams
  - Weitere Aspekte
    - Konsistenz mit Dokumentation, Beispielen, …
    - Hilfreich im Austausch in Blogs, Foren, …

Richtlinien in agilen Methoden

- Extreme Programming (XP)
  - “Code must be formatted to agreed coding standards. Coding standards keep the code consistent and easy for the entire team to read and refactor. Code that looks the same encourages collective ownership.”
- Collective Ownership
  - Gemeinsame Verantwortlichkeit
  - Nach XP essenziell für die erfolgreiche Zusammenarbeit im Team

Richtlinien in CCD

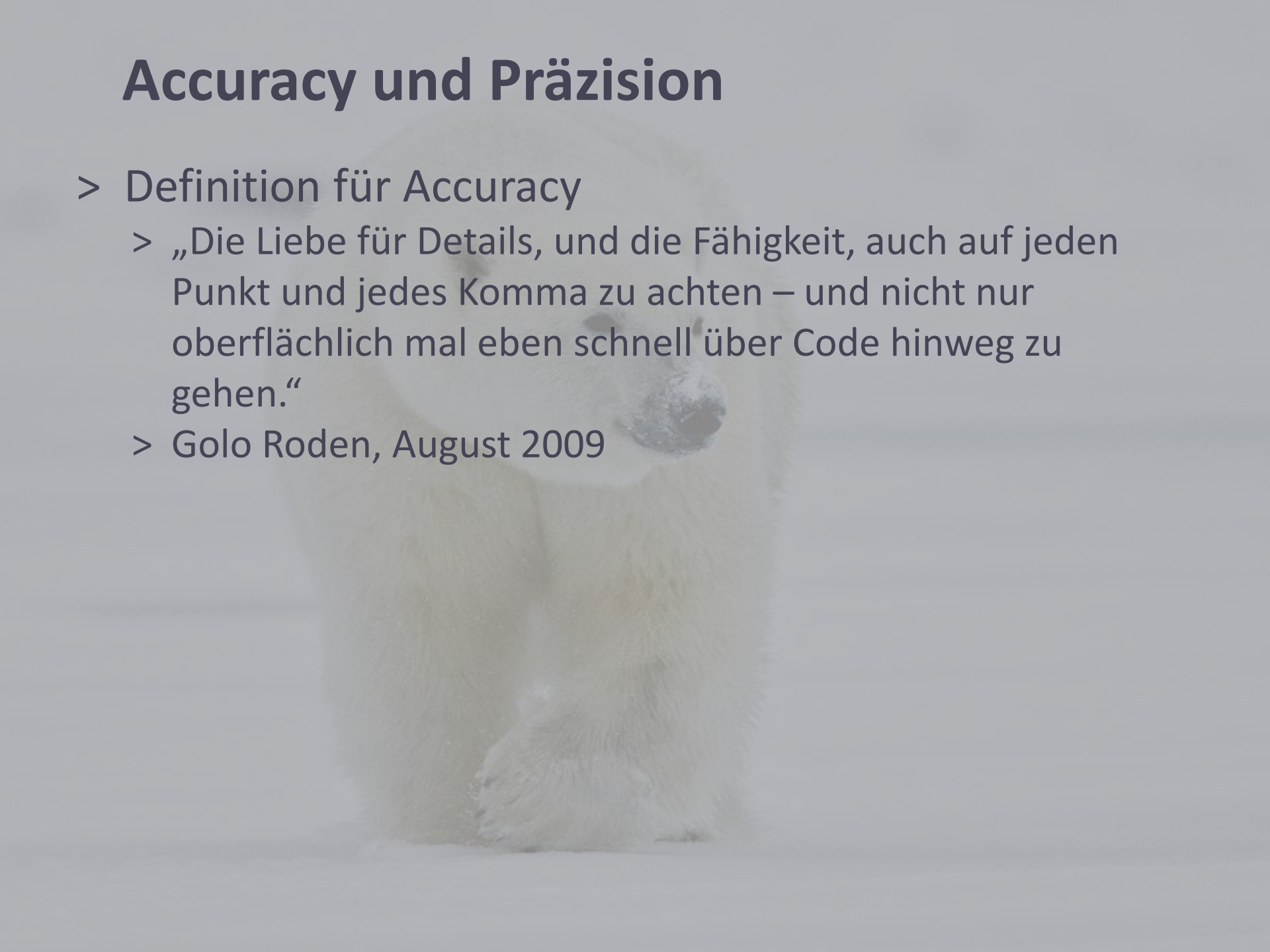
- Oranger Grad
  - “Code wird häufiger gelesen als geschrieben. Daher sind Konventionen wichtig, die ein schnelles Lesen und Erfassen des Codes unterstützen.”
  - Fokus
    - Namensregeln
    - Richtig kommentieren

Einhalten der Richtlinien

- Richtlinien müssen eingehalten werden
  - Eigenes Augenmerk
    - Accuracy
    - Präzision
  - Unterstützung durch Werkzeuge

Accuracy und Präzision

- Definition für Accuracy
  - „Die Liebe für Details, und die Fähigkeit, auch auf jeden Punkt und jedes Komma zu achten – und nicht nur oberflächlich mal eben schnell über Code hinweg zu gehen.“
  - Golo Roden, August 2009



Accuracy und Präzision

- Definition in Analogie zur Waffenkunde
  - *Accuracy* bezeichnet die Treffgenauigkeit, wie weit Treffer also vom gewünschten Ziel abweichen
  - *Präzision* hingegen bezeichnet die Wiederholbarkeit, wie groß die Varianz der Treffer also ist
- Accuracy und Präzision sind unabhängig voneinander

Accuracy und Präzision

- Übertragen auf Codequalität
  - *Accuracy* bezeichnet die Abwesenheit von Formfehlern
    - Korrekte Einrückung
    - Korrekte Leerzeilen
    - Korrekte Klammerung
    - Korrekte Schreibweise
    - Korrekte Rechtschreibung
    - Korrekte Grammatik
  - *Präzision* bezeichnet die Varianz von Fehlern
    - Eine Art von Fehler oder verschiedene Arten von Fehlern?

Äußere vs innere Form

- Äußere Form
  - Erprobte Vorgehensweisen für Codeform
    - Namensgebung
    - Formatierung
    - Projekthierarchie
    - Projekteinstellungen
- Innere Form
  - Erprobte Vorgehensweisen für Code an sich
    - Klassen vs Strukturen
    - Schnittstellen vs abstrakte Basisklassen vs Klassen
    - Vererbung vs Komposition

Äußere Form

- Accuracy und Präzision
  - Auch für äußere Form relevant
  - Definitionen von Accuracy und Präzision können übertragen werden
    - *Accuracy* bezeichnet die Abwesenheit von Formfehlern
    - *Präzision* bezeichnet die Varianz von Fehlern
- Daher
  - „Die Liebe für Details, und die Fähigkeit, auch auf jeden Punkt und jedes Komma zu achten – und nicht nur oberflächlich mal eben schnell über Code hinweg zu gehen.“

Unterstützung durch Werkzeuge

- Arten von Werkzeugen
  - Eingabehilfen
  - Refaktorisieren
  - Codearrangement
  - Statische Codeanalyse

Projekteinstellungen

- Richtlinien
  - Treat Warnings as Errors
  - Warning Level 4
  - Generate XML Output
- DEMO

Ordner- und Dateistruktur

- Richtlinien
  - Ein Typ pro Datei – eine Datei pro Typ
  - Partielle Klassen
  - Namensräume auf Ordner mappen
  - Usings
- DEMO

Zeilenumbrüche und Einrückung

- Richtlinien
  - Zeilenumbrüche
  - Einrückung
- DEMO

Whitespace

- Richtlinien
  - Leerzeilen
  - Leerstellen
  - Tabellenartiges Formatieren
- DEMO

Kommentare

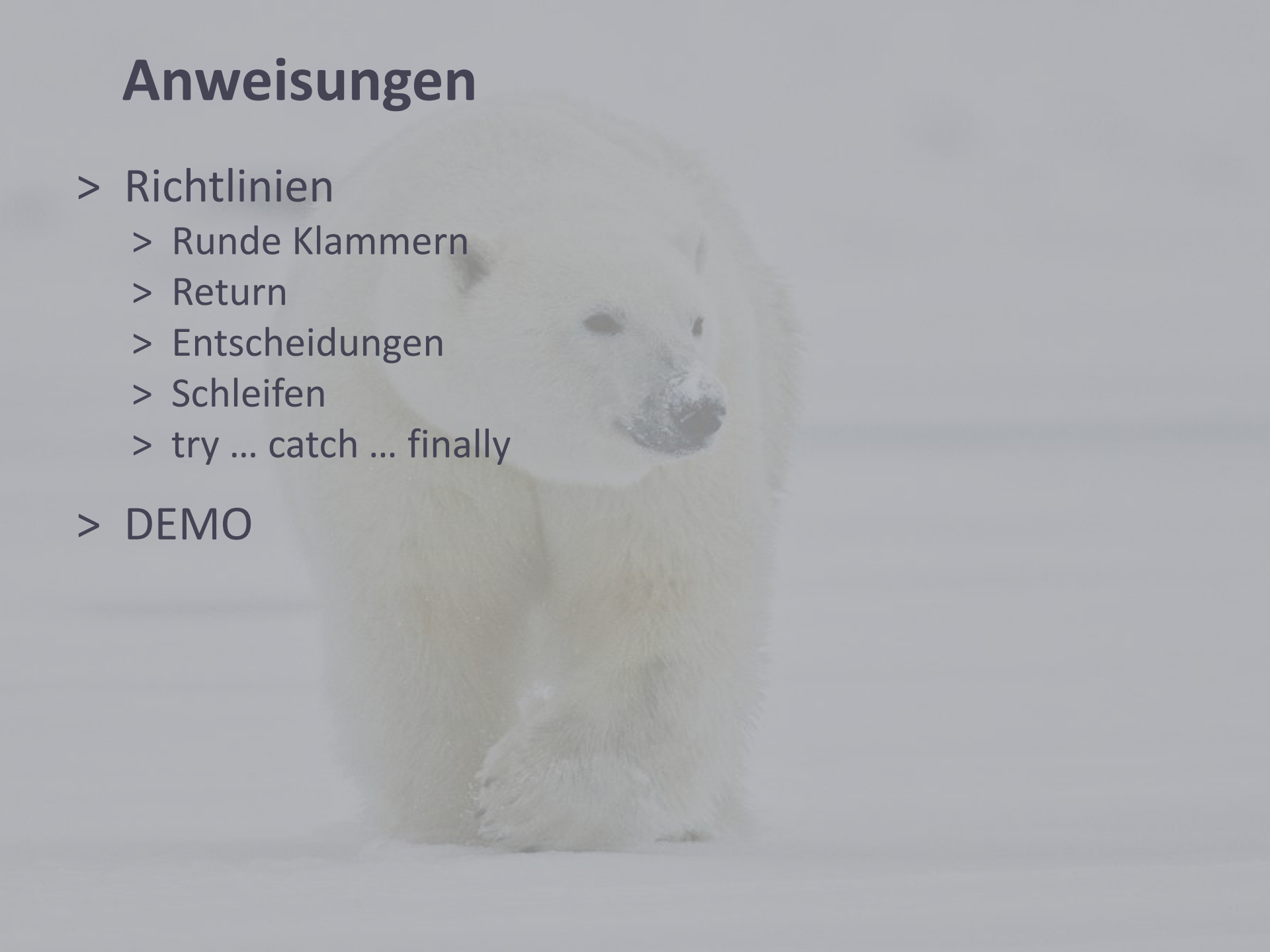
- Richtlinien
  - Einzeilige Kommentare
  - Blockkommentare
  - XML-Kommentare
- DEMO

Deklarationen

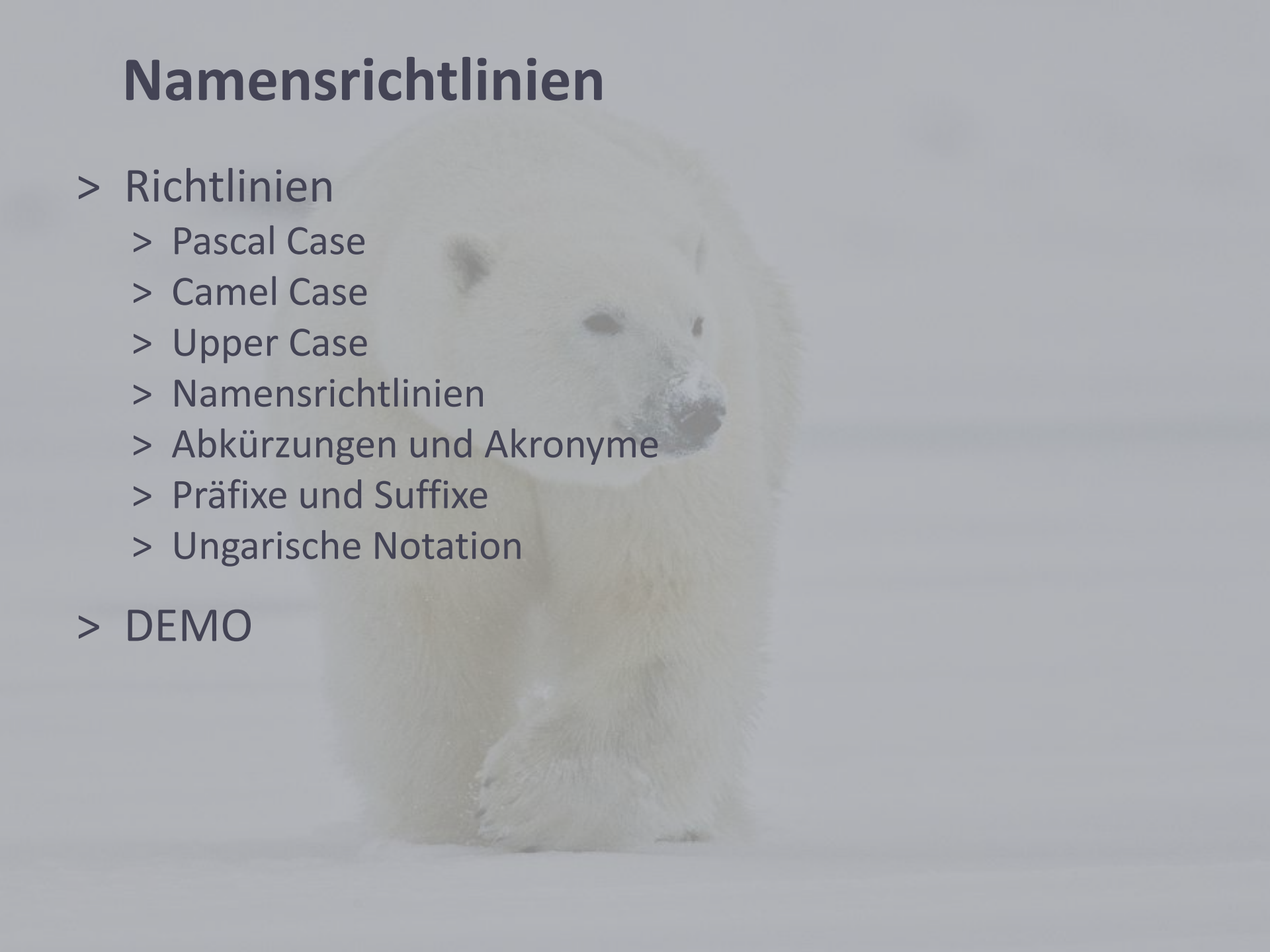
- Richtlinien
  - Deklaration vs Definition
  - Geschweifte Klammern
  - Zugriffsmodifizierer
- DEMO

Anweisungen

- Richtlinien
  - Runde Klammern
  - Return
  - Entscheidungen
  - Schleifen
  - try … catch … finally
- DEMO



Namensrichtlinien



- Richtlinien
  - Pascal Case
  - Camel Case
  - Upper Case
  - Namensrichtlinien
  - Abkürzungen und Akronyme
  - Präfixe und Suffixe
  - Ungarische Notation
- DEMO

Klassen und Strukturen

- Richtlinien
  - Referenz- vs Wertetypen
  - Managed Heap vs Stack
  - Call by Reference vs Call by Value
  - Boxing
- DEMO

Schnittstellen und abstrakte Klassen

- Richtlinien
  - Schnittstellen sind unveränderlich
  - Schnittstellen vererben
  - Schnittstellen oder abstrakte Klassen?
  - Abstrakte um konkrete Klassen ergänzen
- DEMO

Statische Klassen

- Richtlinien
  - Vor- und Nachteile
  - Einsatz als Singleton
- DEMO

Konstruktoren

- Richtlinien
  - Expliziter Standardkonstruktur
  - Fabrikmethode in Betracht ziehen
  - Konstruktoren verketten
  - Virtuelle Methoden aufrufen
  - Keine Ausnahmen in Typinitialisierern
- DEMO

Methoden

- Richtlinien
  - Methoden überschreiben
  - Parameterreihenfolge beachten
  - null für optionale Parameter
- DEMO

Eigenschaften

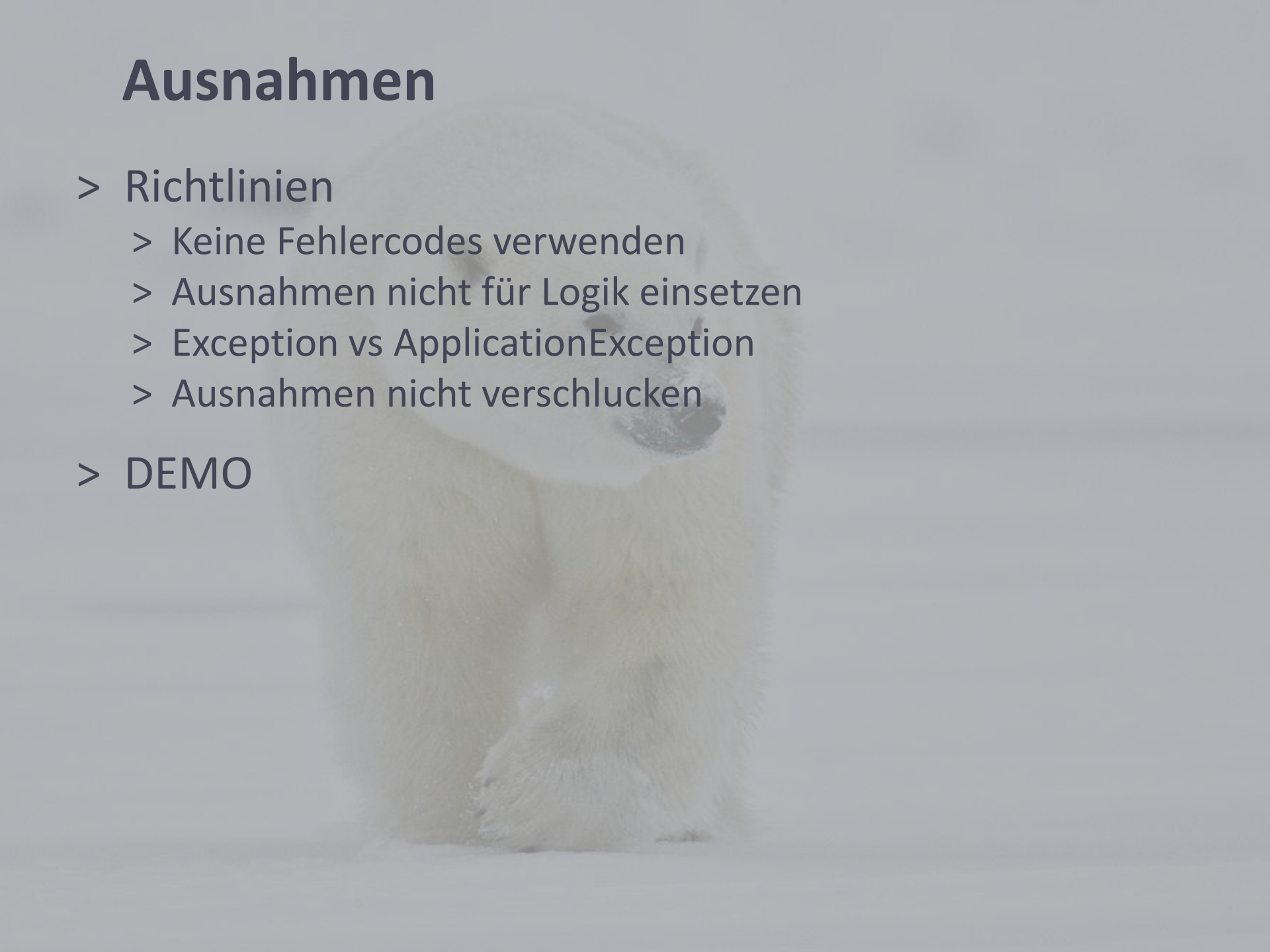
- Richtlinien
  - get
  - set
  - INotifyPropertyChanged
  - Eigenschaften vs Methoden
- DEMO

Vererbung

- Richtlinien
  - Versiegelte Typen
  - Template-Entwurfsmuster
  - virtual mit Bedacht
- DEMO

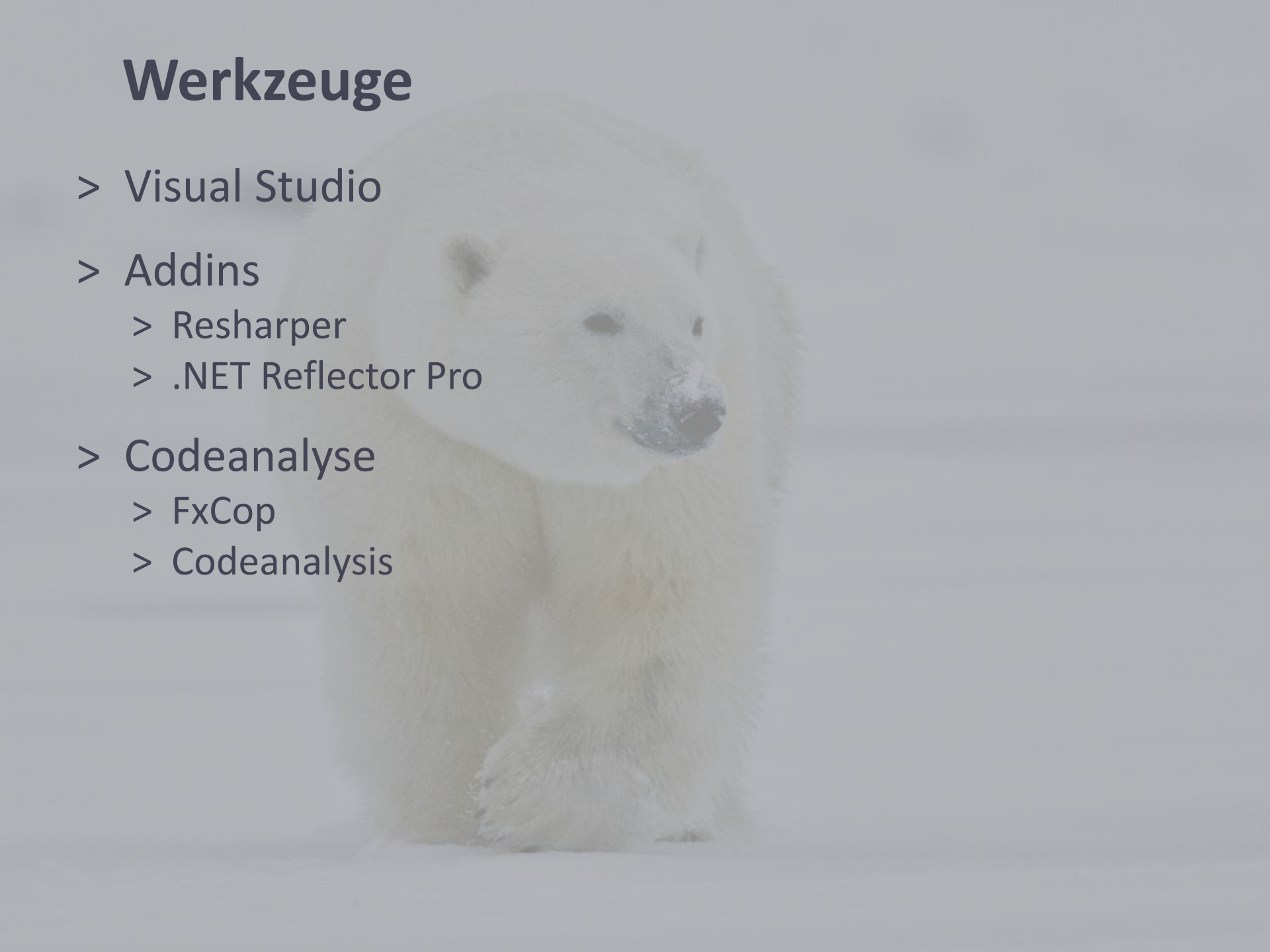
Ausnahmen

- Richtlinien
  - Keine Fehlercodes verwenden
  - Ausnahmen nicht für Logik einsetzen
  - Exception vs ApplicationException
  - Ausnahmen nicht verschlucken
- DEMO



Werkzeuge

- Visual Studio
- Addins
  - Resharper
  - .NET Reflector Pro
- Codeanalyse
  - FxCop
  - Codeanalysis



Fazit

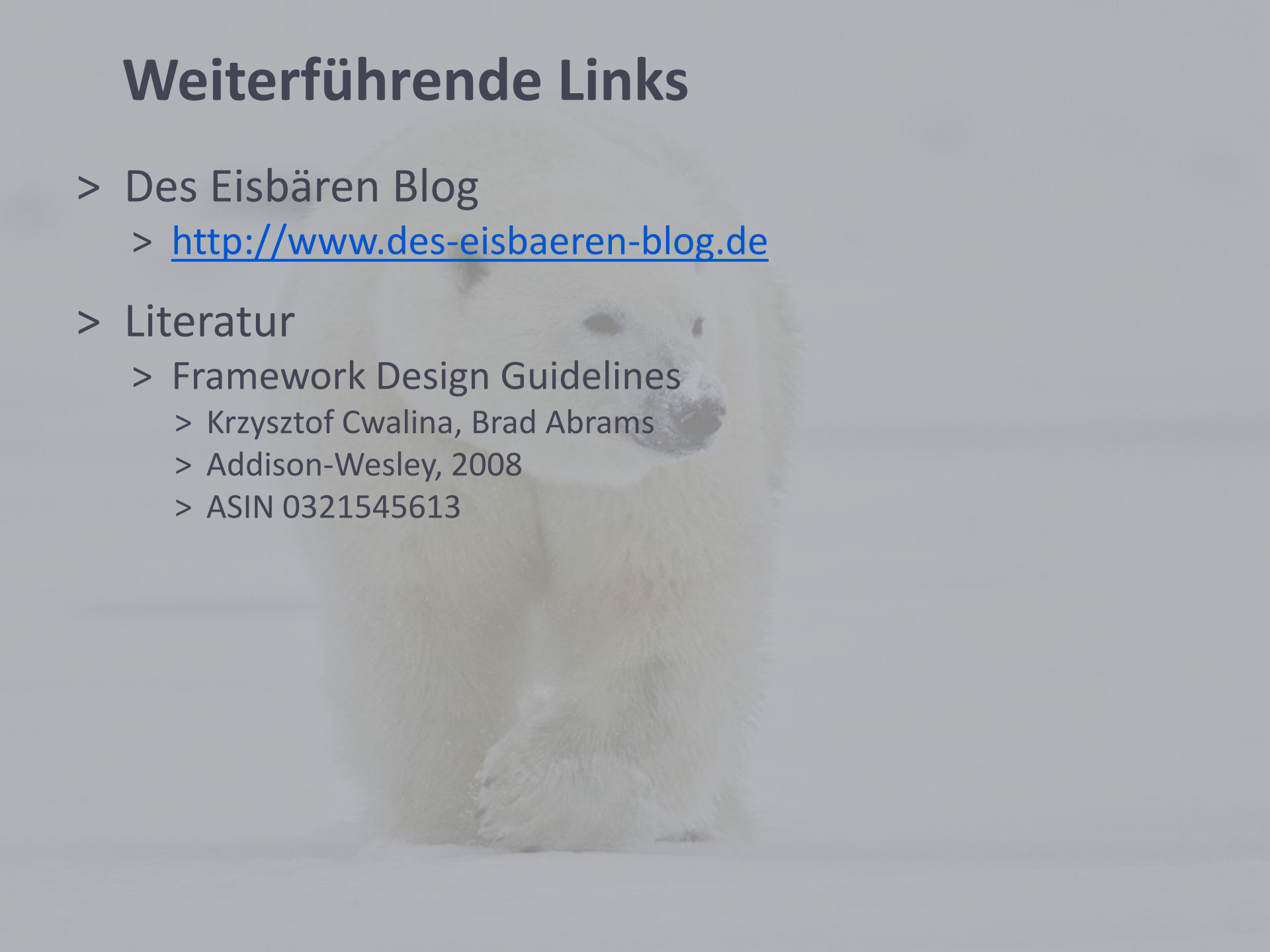
- Codequalität
  - Führt zu Wart- und Evolvierbarkeit
  - Erfordert Standards
- Richtlinien
  - Entsprechen erprobten Vorgehensweisen
  - Weder “richtig” noch “falsch”
  - Gefordert in XP und CCD
- Stil
  - Persönliche Note für Richtlinien
  - Konsistenz zu anderen beachten

Fazit

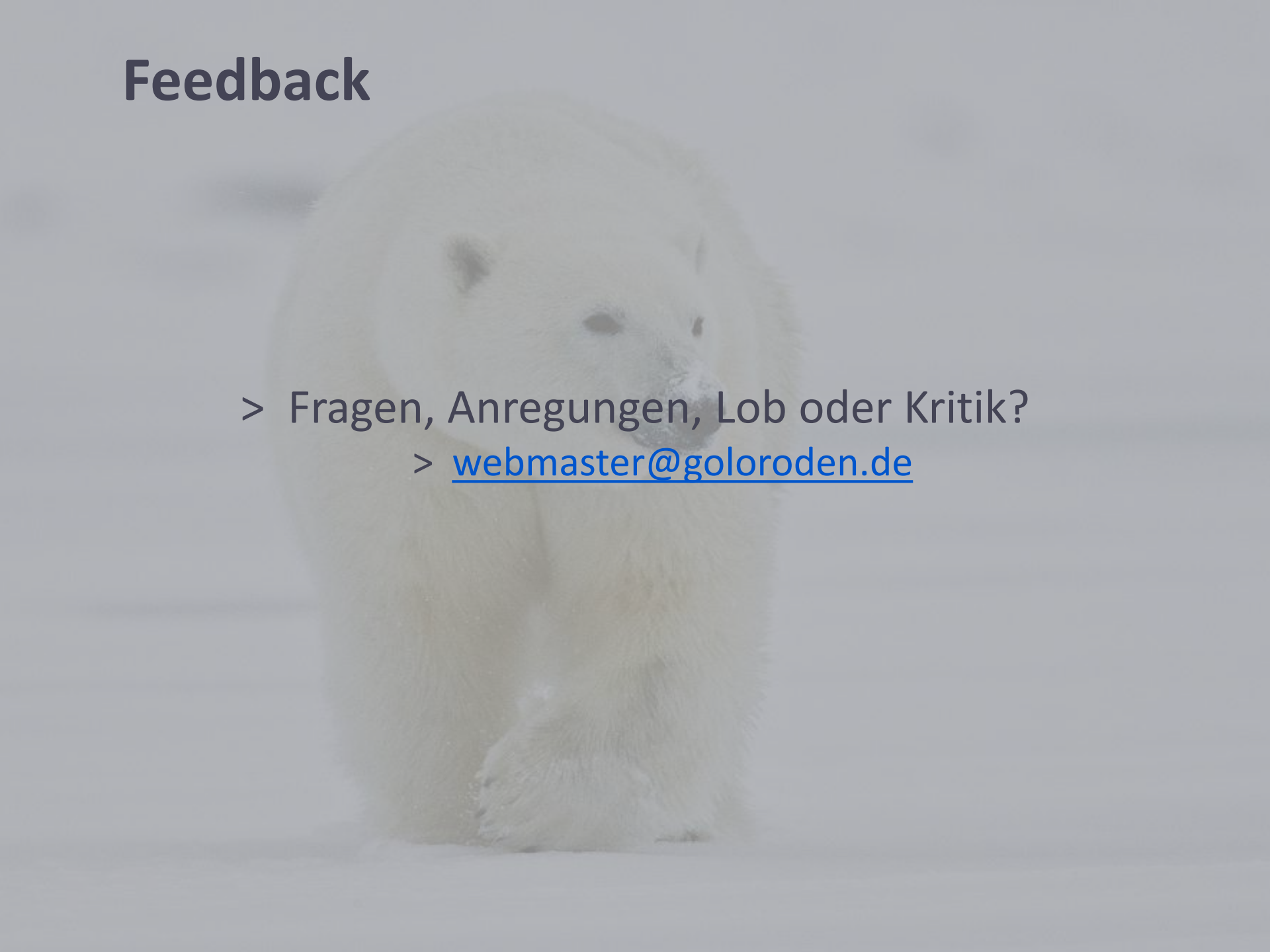
- Äußere Form
  - Optisches Erscheinungsbild
  - Namensvergabe
  - MSDN-Richtlinien als sinnvolle Basis
- Innere Form
  - Im mathematischen Sinne eleganter Code
  - Sprachkonstrukte sachgemäß angewendet
  - Erfahrung als wesentliche Richtlinie
- Visual Studio als Basis
  - Resharper und .NET Reflector Pro als ergänzende Addins
  - FxCop und Codeanalyse zur automatisierten Prüfung

Weiterführende Links

- Des Eisbären Blog
  - http://www.des-eisbaeren-blog.de
- Literatur
  - Framework Design Guidelines
    - Krzysztof Cwalina, Brad Abrams
    - Addison-Wesley, 2008
    - ASIN 0321545613



Feedback



> Fragen, Anregungen, Lob oder Kritik?

> webmaster@goloroden.de