DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Tutorium Programmierung 2

15.05.2025

Janens Kurzke und Fabian Bauriedl

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Inhalt

1. Exceptions
2. Generics und Optionals
3. Datenstrukturen
4. Big-O-Notation
5. Git

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Exceptions

- Fehler die während der Laufzeit auftreten
- Dienen zur Kommunikation
- Manuelles auslösen über throw
- Abfangen über catch
- Exceptions sind Klassen/ Objekte

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Exceptions werfen

```
public static void checkAge(int age) throws Exception {
    if(age < 18) {
        throw new Exception("You're to young");
    }
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Eigene Exceptions

```
public class CustomExceptions extends Exception{
    CustomExceptions(String message){
        super(message);
        System.out.println("I am a custom exception");
    }
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Exceptions fangen

```
public static void main(String[] args) {
    try {
        checkAge(17);
    } catch (ToYoungException e) {
        System.out.println("User appears to be to young");
    } catch (Exception e) {
        System.out.println("An unknown error occured");
    } finally {
        System.out.println("finished try catch block");
    }
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Generics

- Generische Klassen und Interfaces
- Bezieht sich auf Parameter/ Argumente
- Funktionalität unabhängig von Typen implementieren

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Generics Anwendung

- Verwendung des komplexen Datentypes

```
public static void main(String[] args) {
    ArrayList<String> names = new ArrayList<>();
    names.add("Christian");
    names.add("Sabine");
    int len = names.size();
    names.remove(0);
}
```

Generics Implementierung

DHBW
Duale Hochschule
Baden-Württemberg

```
public class Paar<T, U> {
    private final T firstElement;
    private final U secondElement;

    public Paar(T firstElement, U secondElement){
        this.firstElement = firstElement;
        this.secondElement = secondElement;
    }

    @Override
    public String toString(){
        return firstElement + " and " + secondElement;
    }
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Optionals

- Wrapper Klasse
- Vermeidung von Nullpointer Exceptions
- Zugriff auf Werte über umwege

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Optionals Anwendung I

```
Optional<String> name = Optional.of("Lindner");
if (name.isPresent()){
    System.out.println("there is a name");
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Optionals Anwendung II

```
String userInput = null;
Optional<String> userName = Optional.ofNullable(userInput);

userName.ifPresentOrElse(
    (value) -> System.out.println(value),
    () -> System.out.println("there is no name"));
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Datenstrukturen

1. Stack
2. Queue
3. List
4. Tree
5. HashMap

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Stack

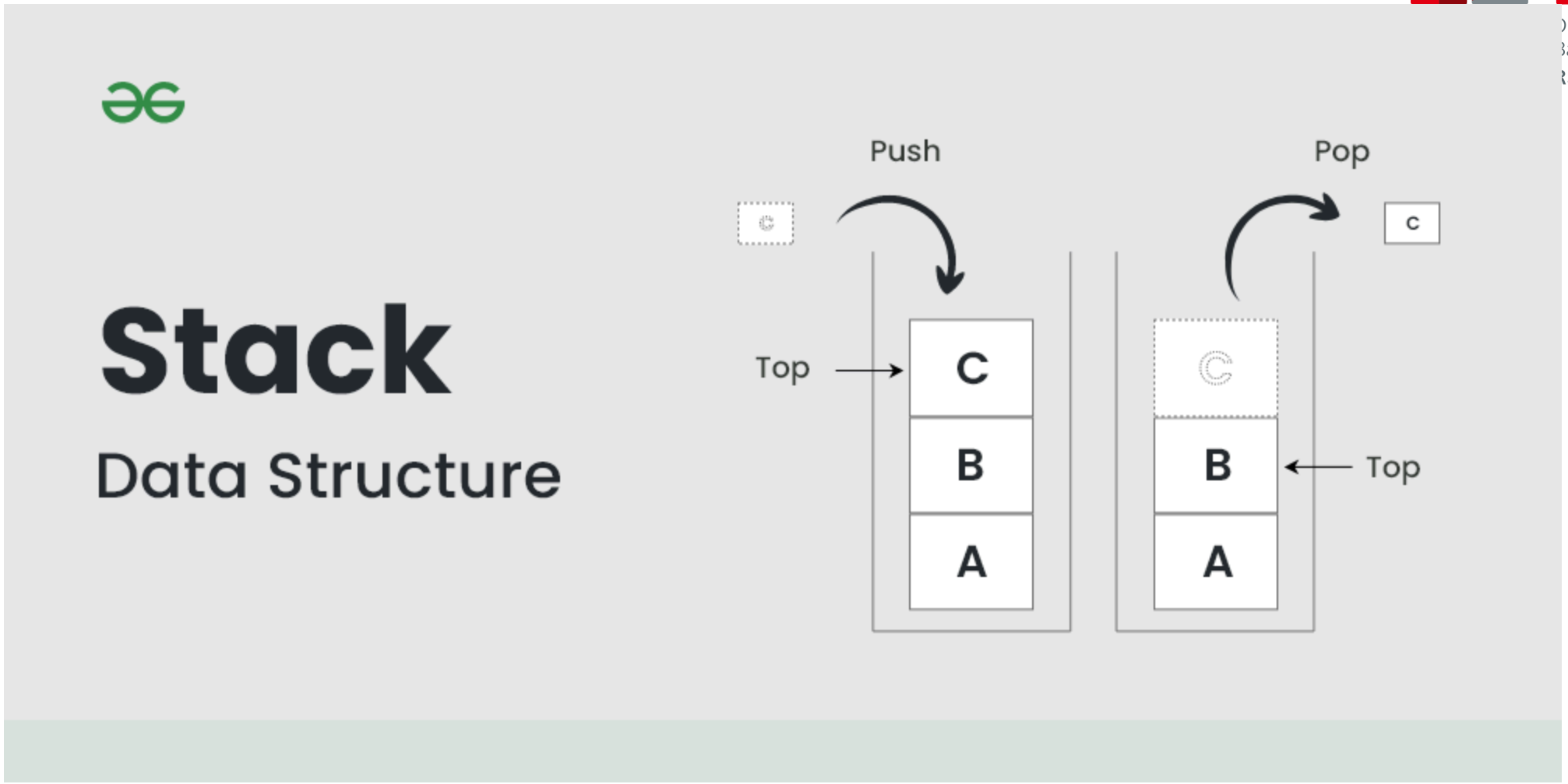
- Stapel von Elementen
- Neue Elemente werden oben angefuegt
- Zugriff nur von oben moeglich
- Neueste Elemente zuerst --> Last in First Out
- z.B. Undo/ Redo, Browserverlauf

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Stack Operationen

Return	Method	Description
boolean	empty()	check ob stack leer ist
E	peek()	gibt oberstes Objekt und behält es
E	pop()	gibt oberstes Objekt und entfernt es
E	push(E item)	fügt Objekt oben an
int	search(Object o)	sucht Objekt und gibt Position zurück

Stack



DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Queue

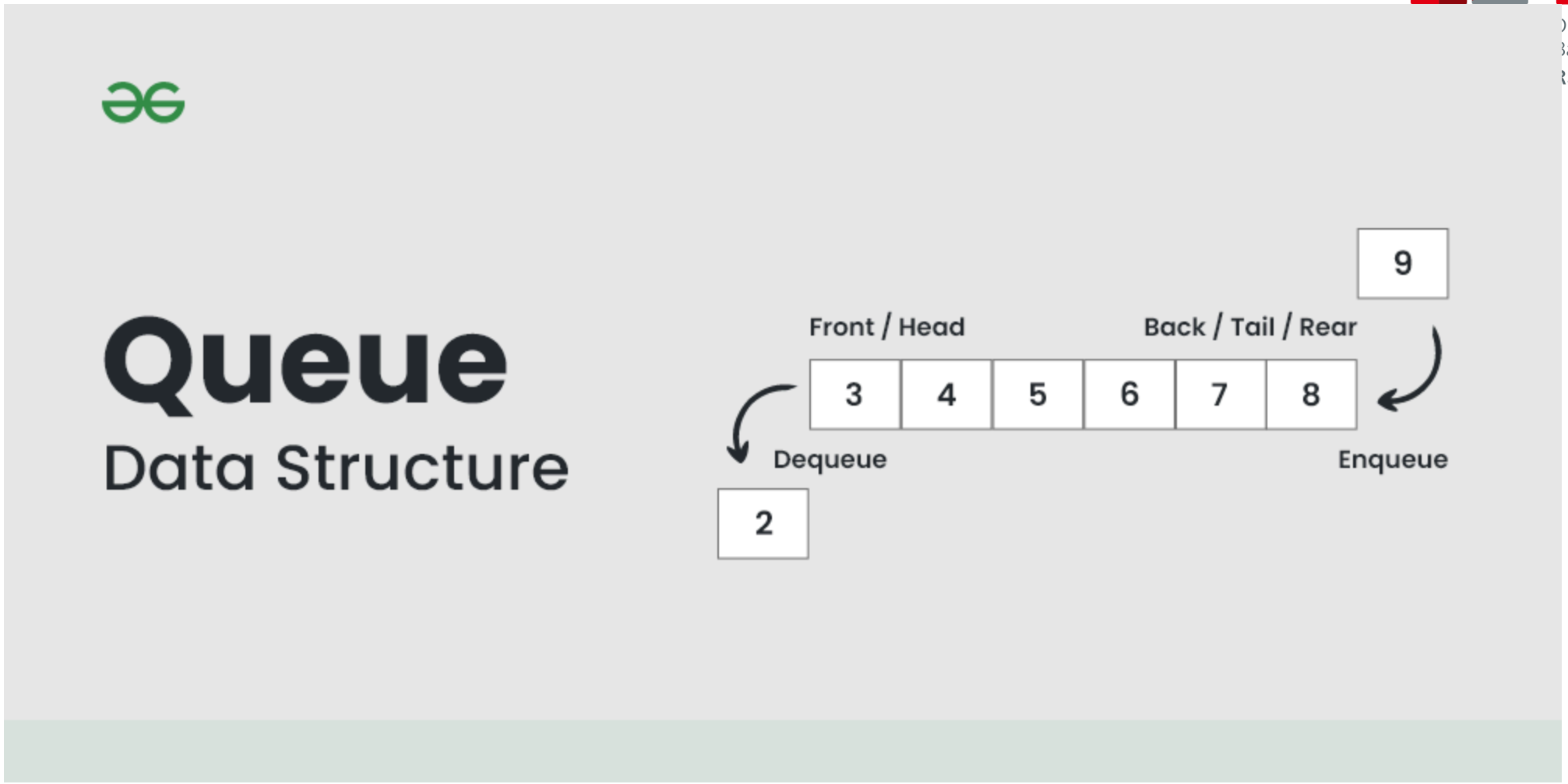
- Warteschlange von Elementen
- Neue Elemente werden unten angefuegt
- Zugriff nur von oben moeglich
- Älteste Elemente zuerst --> First in First Out
- z.B. Datenübertragung, Warteschlangen, Schedular (Betriebssysteme)

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Queue Operationen

Return	Method	Description
boolean	enqueue(E e)	Objekt hinten anfügen
E	dequeue()	Objekt vorne entfernen
E	peek()	gibt vorderstes Element und behält es
boolean	isEmpty()	check ob Queue leer ist

Queue


Queue
Data Structure
9
Front / Head
Back / Tail / Rear
3
4
5
6
7
8
Dequeue
Enqueue
2

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

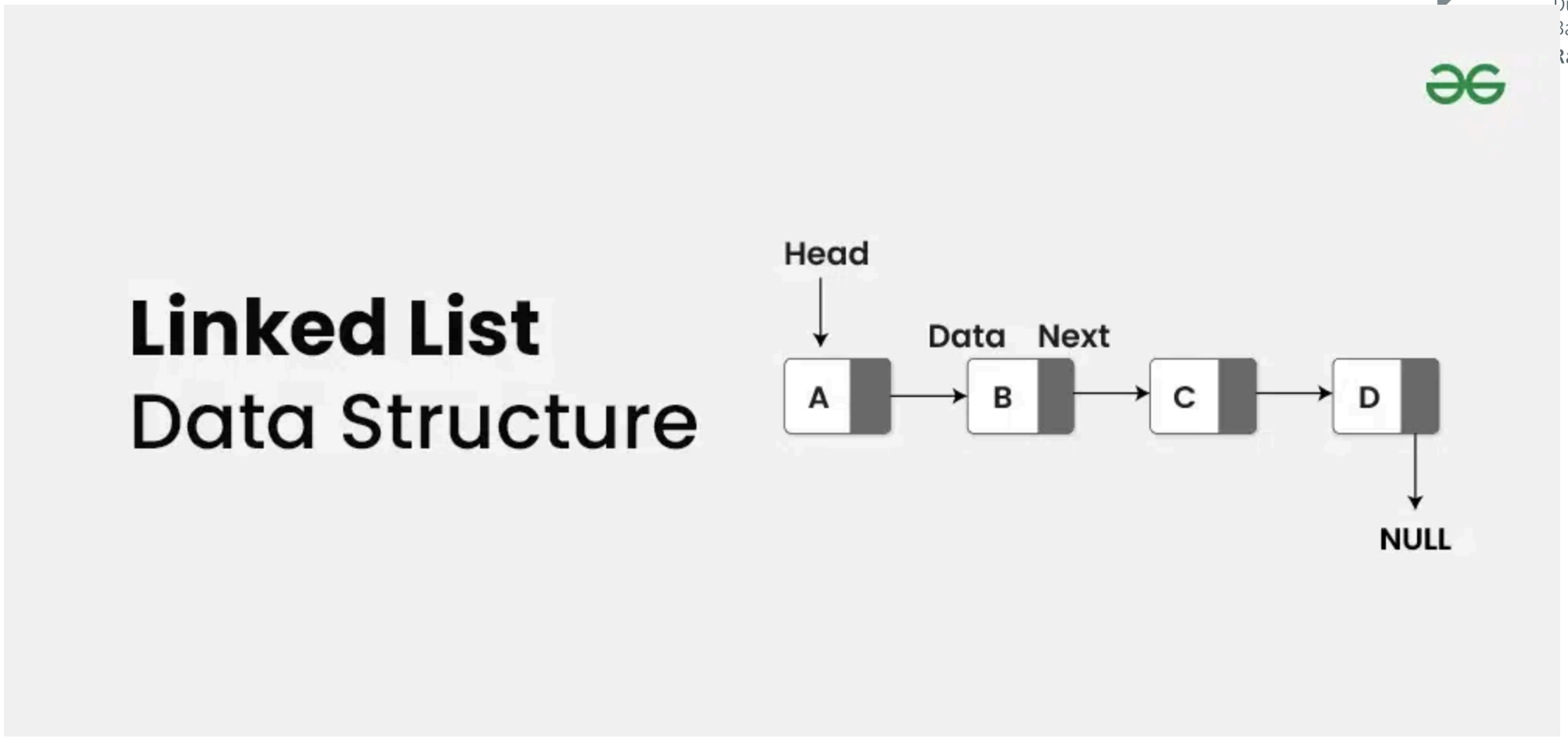
List

- Liste von Elementen
- Standard anfügen von hinten
- Einfügen an bestimmter Stelle möglich
- Zugriff auf Elemente über Index

List Operationen

Return	Method	Description
int	length()	Länge der Liste
void	prepend(E e)	Objekt vorne anfügen
void	append(E e)	Objket hinten anfügen
boolean	insertAt(E e, int i)	Objekt bei Index einfügen
boolean	remove(E e)	gegebenes Objekt löschen
void	removeAt(int i)	Objekt an Index löschen
Object	get(int i)	Gibt Objekt an Index aus

Linked List



DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Tree

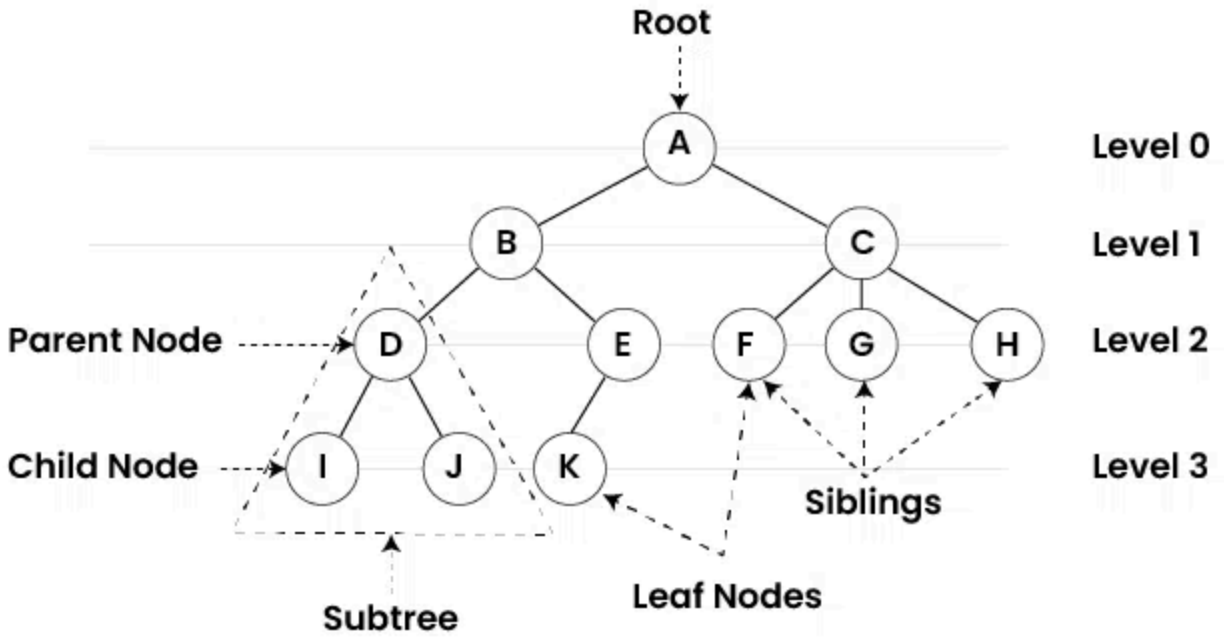
- hierarchische Datenstruktur
- effiziente suche und navigation des Baums
- Daten liegen in Knoten
- Knoten sind über Kanten verbunden
- z.B. Dateisysteme

Tree

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Tree

Data Structure



DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Tree Begriffe

- Root Node
- Child Node
- Parent Node
- Leaf Node
- Level

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Tree Begriffe

- Ancestor Node
- Descendant Node
- Sibling
- Neighbor
- Subtree

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Tree Arten

- Binary Tree
- Ternary Tree
- Quadtree
- N-Tree

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Binary Tree

- 0 bis 2 Child Nodes pro Parent Node
- Tree ist unsortiert

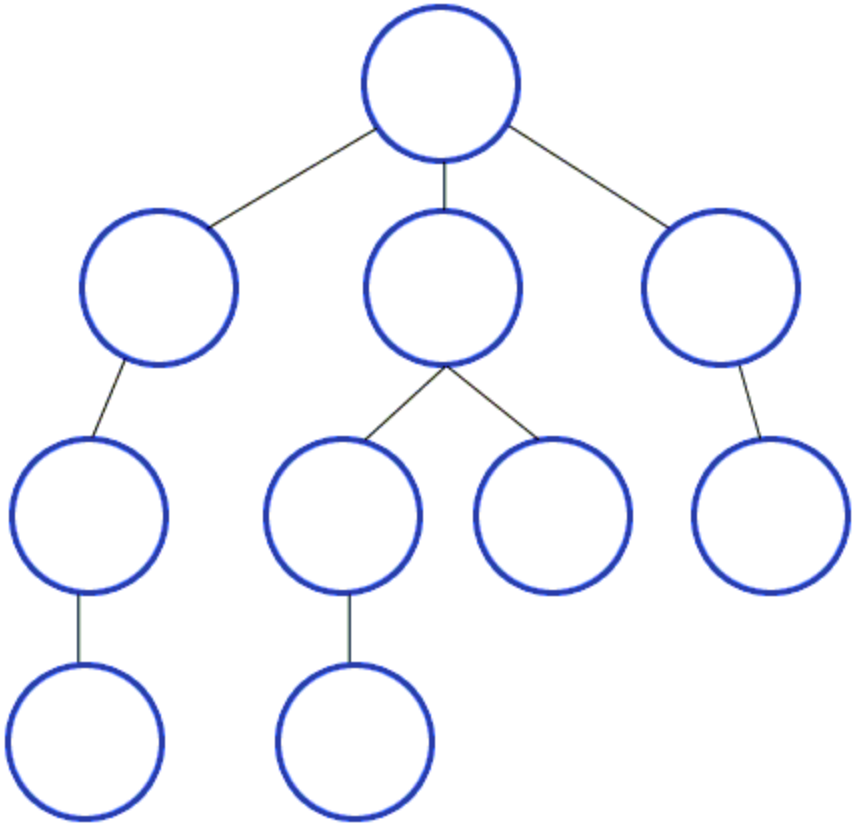
```
public class Node {
    public int number;
    public Node left;
    public Node right;
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Binary Tree Operations

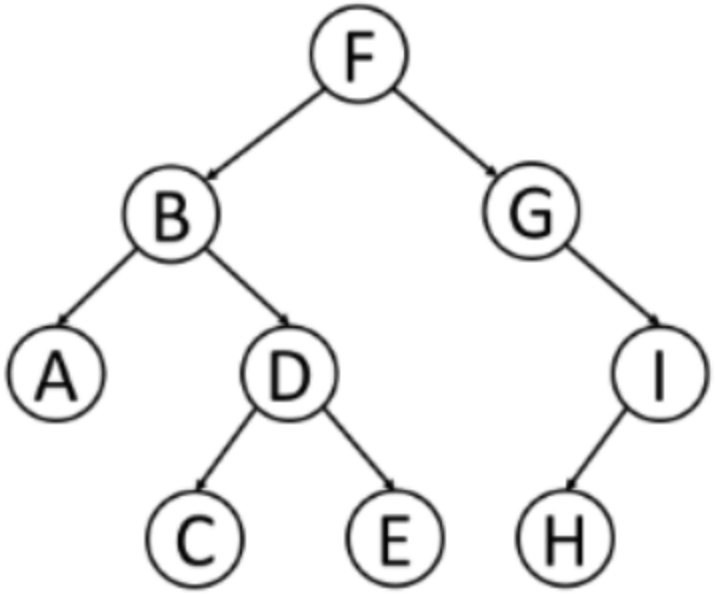
Return	Method	Description
int	traverse()	Länge der Liste
void	insert(E e)	Objekt einfügen
void	delete(E e)	Objekt löschen
int	search(E e)	Sucht Objekt und gibt Index zurück

Binary Tree Depth First Search



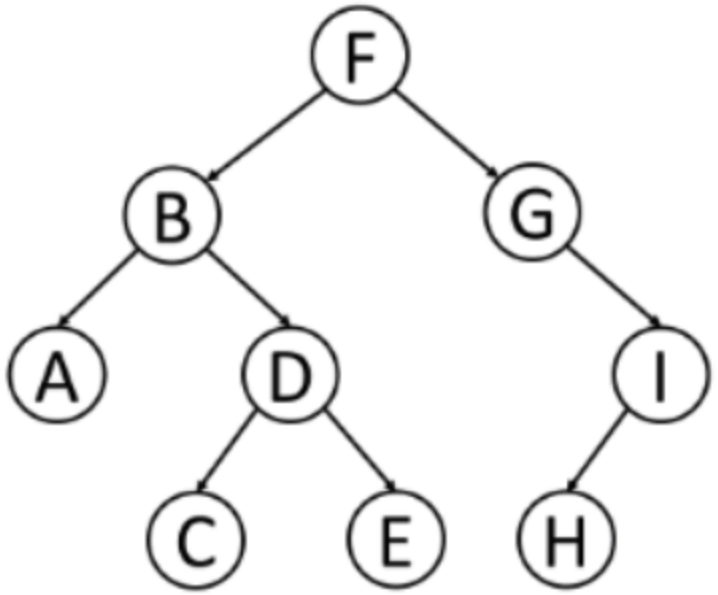
DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Binary Tree Pre-Order-Traversal



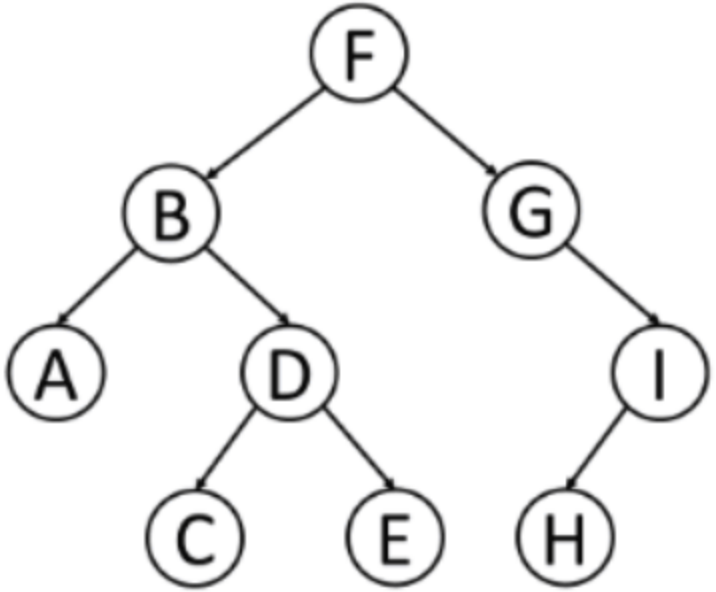
Preorder:

Binary Tree In-Order-Traversal



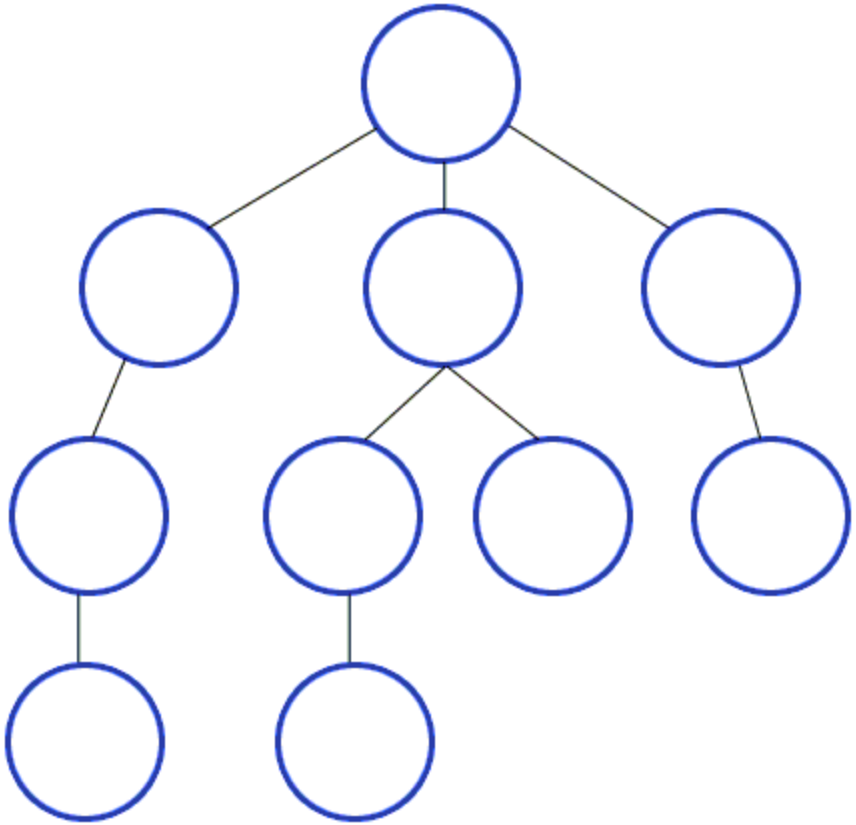
Inorder:

Binary Tree Post-Order-Traversal



Postorder:

Binary Tree Breadth First Search



DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Binary Search Tree

- Binary Tree mit Sortierung
- Left Child Node <= Parent Node
- Right Child Node > Parent Node

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Binary Search Tree Operations

Return	Method	Description
int	search(E e)	Sucht Objekt im Tree und gibt Index zurück
void	insert(E e)	Objekt einfügen
void	delete(E e)	Objekt löschen

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Set

- Interface
- Abbildung einer Menge

Operationen

- Vereinigung (Union)
- Durchschnitt (Intersection)
- Differenz (Difference)

HashSet Example

DHBW

```
public static void main(String[] args) {
    Set<Integer> numbers = new HashSet<Integer>();
    Set<Integer> otherNumders = new HashSet<Integer>();

    for (int i = 0; i < 10; i++) {
        numbers.add(i);
        //[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
        otherNumders.add(i*2);
        //[0, 16, 2, 18, 4, 6, 8, 10, 12, 14]
    }
    numbers.size();
    //10
    otherNumders.addAll(numbers);
    //[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18]
    otherNumders.retainAll(numbers);
    //[0, 2, 4, 6, 8]
    otherNumders.removeAll(numbers);
    //[16, 18, 10, 12, 14]
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Map

- Interface
- Schlüssel-Wert-Paare
- Keine doppelten Schlüssel

HashMap Example

DHBW
Duale Hochschule

```
public static void main(String[] args) {
    Map<Integer, String> idNameMap = new HashMap<Integer, String>();

    idNameMap.put(123, "Christian");
    idNameMap.put(1234, "Sabine");

    idNameMap.get(1234);
    idNameMap.size();
    idNameMap.remove(1234);
    idNameMap.containsKey(123);
    idNameMap.containsValue("Christian");
    idNameMap.keySet();     //[123]
    idNameMap.entrySet();   //[123=Christian]
    idNameMap.values();     //[Christian]
}
```

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Hash

- Abbildung großer Datenmenge auf kleine
- Ergebnis einer Hashfunktion
- Einwegsfunktion
- Deterministisch
- Kollisionen sind möglich (gleicher Hash bei unterschiedlicher Eingabe)

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

HashMap insert Funktionsweise

1. Hashwert aus Schlüssel berechnen --> Index
2. Wenn kein Wert an Index --> einfügen
3. Wenn Wert an Index --> Werte vergleichen
4. Wenn Werte gleich --> alten Wert ersetzen
5. Wenn Werte ungleich --> Speicher vergrößern

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

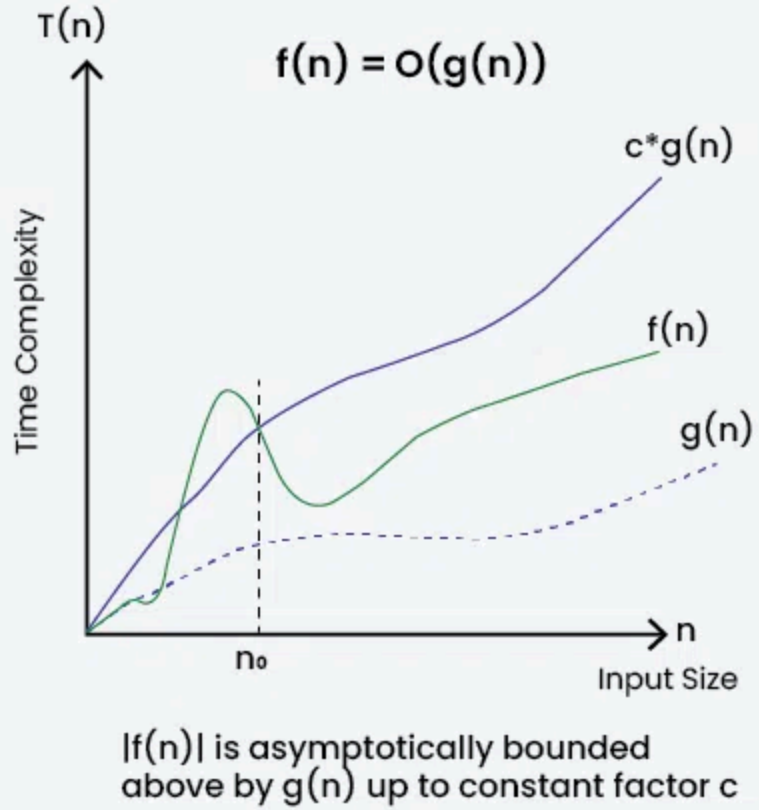
Big-O-Notation

- Bewertung der Speicher-Komplexität und Zeit-Komplexität
- Unterscheidung in Best-Case, Average-Case und Worst-Case
- Beschreibt wie die Speicher- und Zeit-Komplexität mit Problem skaliert
- Darstellung in Funktion O(f(n))
- n gibt die Größe des Problems an (z.B. Datenpunkte)
- Anwendbar auf Algorihmen und Datenstrukturen

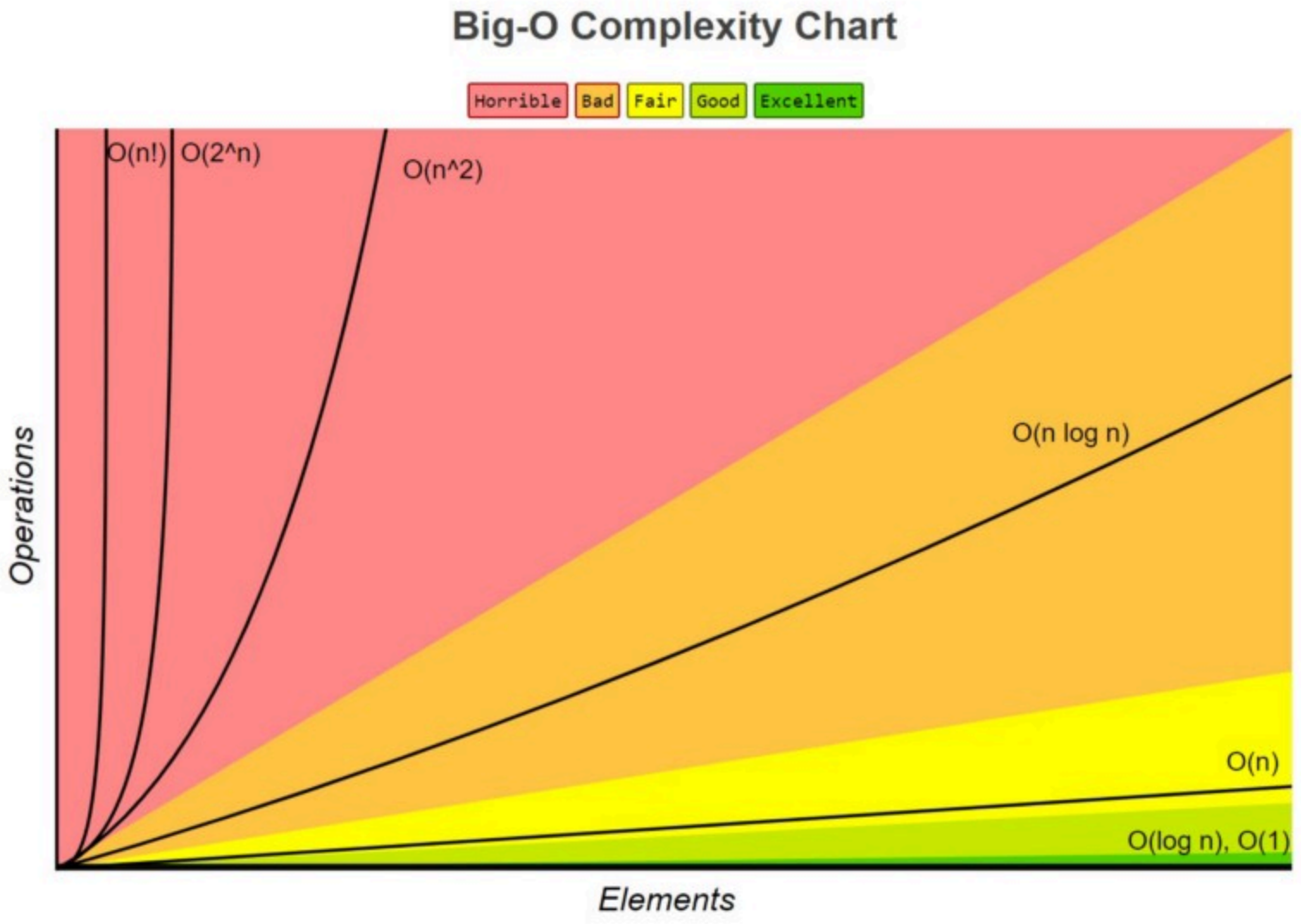
Big-O-Notation

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Big O
Analysis



Big-O-Notation



DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Big-O-Notation Sorting

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Big-O-Notaton Search

TIME COMPLEXITY

ALGORITHM	BEST CASE	AVERAGE CASE	WORST CASE
LINEAR SEARCH	O(1)	O(N)	O(N)
BINARY SEARCH	O(1)	O(LOG2N)	O(LOG2N)
TERNARY SEARCH	O(1)	O(LOG3N)	O(LOG3N)
JUMP SEARCH	O(1)	O(√N)	O(√N)
INTERPOLATION SEARCH	O(1)	O(LOG(LOG(N)))	O(N)

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Git

- Open Source Version Controll Software
- Verfolgung von Änderungen an Dateien, insbesondere Code
- Kolaboration an einem Projekt wird möglich
- Verwendung von lokalen und remote Repositories
- Erstellung von "Savepoints"
- Arbeit mit Branches ermöglicht Prototyping
- Standard Tool in der Softwareentwicklung

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Git-Workflow

1. repository initialisieren (git init)
2. Datei-Änderungen stagen/ snapshots erstellen (git add)
3. Datei-Änderungen commiten (git commit -m"")

Git-Remote-Repository - GitHub

- Repository wird in der Cloud gesichert
- Kolaboration wird möglich

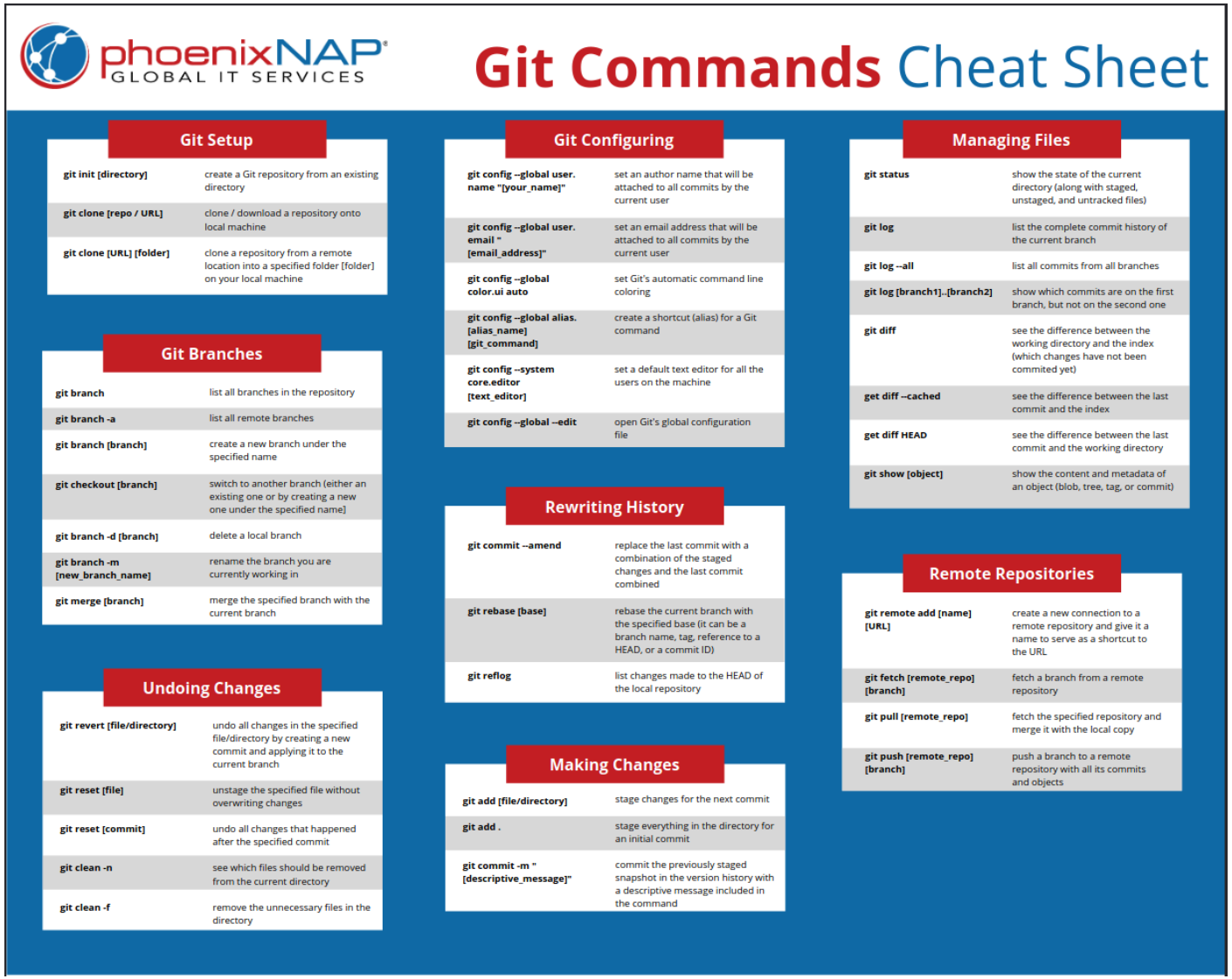
Beispiel Git-Workflow

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

1. GitHub Repository erstellen
2. "gh clone" befehl kopieren (GitHub CLI wird benötigt)
3. .gitignore beeinhaltet Dateiendungen, die nicht von Version Controll erfasst werden sollen
4. Projekt Struktur anlegen, Dateien erstellen und bearbeiten
5. Commit nach erreichen von Meilensteinen
6. Branch wechseln für Prototyping/ entwicklung neuer Features
7. git push um commits mit remote Repository zu synchronisieren
8. Branches über GitHub Webseite mergen
9. Merge Konflikte mit terminal behandeln (commands lassen sich copy pasten von GitHub Seite)

Git Cheat Sheet

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg



phoenixNAP GLOBAL IT SERVICES

Git Commands Cheat Sheet

Git Setup

git init [directory]	create a Git repository from an existing directory
git clone [repo / URL]	clone / download a repository onto local machine
git clone [URL] [folder]	clone a repository from a remote location into a specified folder [folder] on your local machine

Git Branches

git branch	list all branches in the repository
git branch -a	list all remote branches
git branch [branch]	create a new branch under the specified name
git checkout [branch]	switch to another branch (either an existing one or by creating a new one under the specified name)
git branch -d [branch]	delete a local branch
git branch -m [new_branch_name]	rename the branch you are currently working in
git merge [branch]	merge the specified branch with the current branch

Undoing Changes

git revert [file/directory]	undo all changes in the specified file/directory by creating a new commit and applying it to the current branch
git reset [file]	unstage the specified file without overwriting changes
git reset [commit]	undo all changes that happened after the specified commit
git clean -n	see which files should be removed from the current directory
git clean -f	remove the unnecessary files in the directory

Git Configuring

git config --global user. name "[your_name]"	set an author name that will be attached to all commits by the current user
git config --global user. email " [email_address]"	set an email address that will be attached to all commits by the current user
git config --global color.ui auto	set Git's automatic command line coloring
git config --global alias. [alias_name] [git_command]	create a shortcut (alias) for a Git command
git config --system core.editor [text_editor]	set a default text editor for all the users on the machine
git config --global --edit	open Git's global configuration file

Rewriting History

git commit --amend	replace the last commit with a combination of the staged changes and the last commit combined
git rebase [base]	rebase the current branch with the specified base (it can be a branch name, tag, reference to a HEAD, or a commit ID)
git reflog	list changes made to the HEAD of the local repository

Making Changes

git add [file/directory]	stage changes for the next commit
git add .	stage everything in the directory for an initial commit
git commit -m " [descriptive_message]"	commit the previously staged snapshot in the version history with a descriptive message included in the command

Managing Files

git status	show the state of the current directory (along with staged, unstaged, and untracked files)
git log	list the complete commit history of the current branch
git log --all	list all commits from all branches
git log [branch1]..[branch2]	show which commits are on the first branch, but not on the second one
git diff	see the difference between the working directory and the index (which changes have not been commited yet)
get diff --cached	see the difference between the last commit and the index
get diff HEAD	see the difference between the last commit and the working directory
git show [object]	show the content and metadata of an object (blob, tree, tag, or commit)

Remote Repositories

git remote add [name] [URL]	create a new connection to a remote repository and give it a name to serve as a shortcut to the URL
git fetch [remote_repo] [branch]	fetch a branch from a remote repository
git pull [remote_repo]	fetch the specified repository and merge it with the local copy
git push [remote_repo] [branch]	push a branch to a remote repository with all its commits and objects

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

Mentimeter