Java SE 8

I/O und NIO Fundamentals

File

- Abstrakte Representation eines Pfads zu einer Datei oder einem Verzeichnis

- Pfad kann *absolut* oder *relativ* sein

- Kann Dateien und Verzeichnisse erzeugen, löschen, ihre Attribute abfragen und manipulieren

Streams

- Stream-Klassen sind für die Eingabe und Ausgabe von binär oder Byte-Daten gedacht

- Reader und Writer Klassen sind für Ein- und Ausgabe von Char- und String-Daten gedacht

Abstrakte Klassen

- InputStream
- OutputStream
- Reader
- Writer

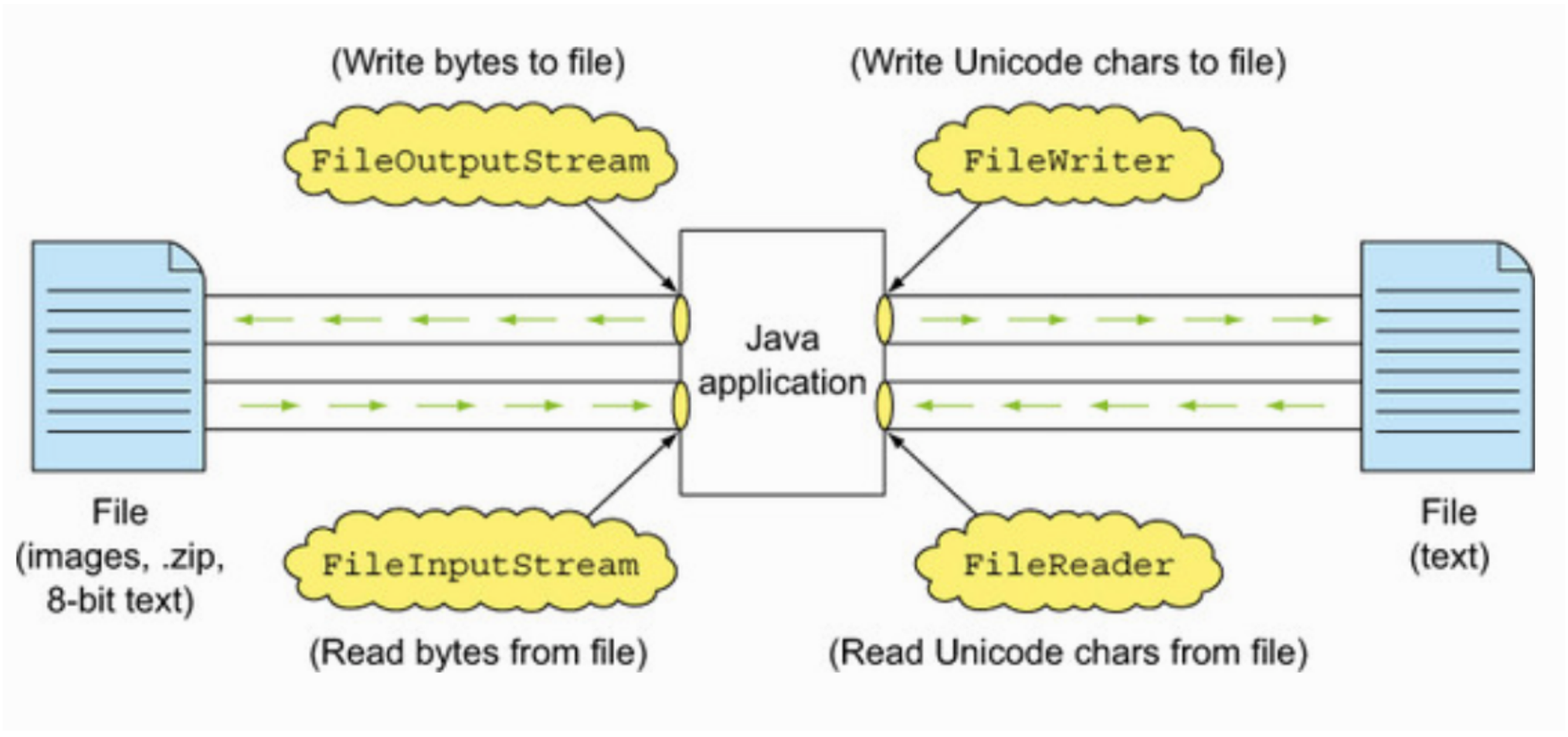
Streams

- Low-Level
  - Verbindet sich direkt mit der Ressource
- High-Level
  - Verwendet einem anderen Stream als Basis
  - einige Operationen auf dem high-level werden an die Basis weitergegeben (z.B. close)

Streams (Low-Level)

- FileInputStream
  - liest Inhalt der Datei als bytes
- FileOutputStream
  - schreibt Inhalt der Datei als bytes
- FileReader
  - liest Inhalt der Datei als Zeichen
- FileWriter
  - schreibt Inhalt der Datei als Zeichen

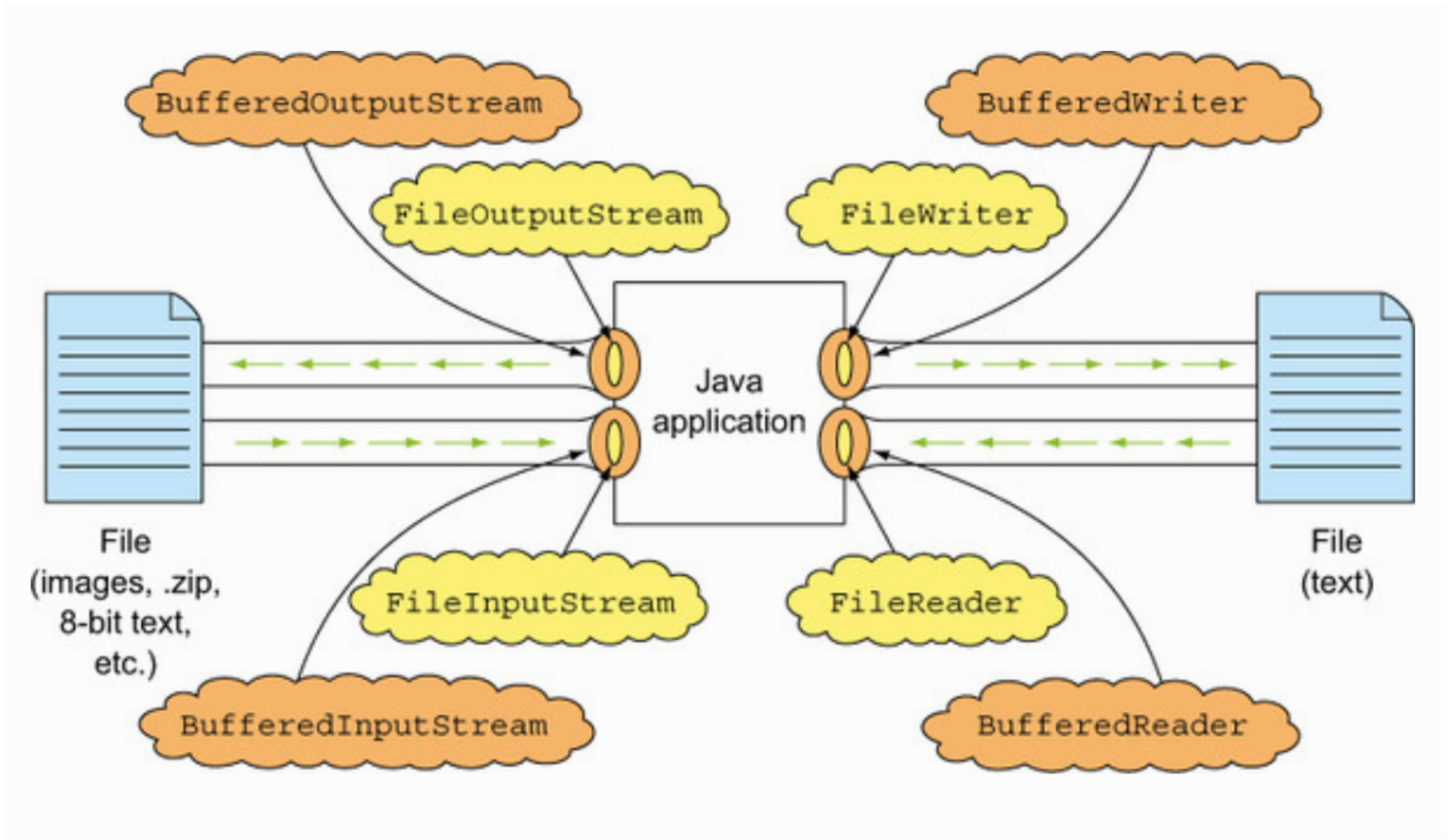
Streams


(Write bytes to file)
(Write Unicode chars to file)
FileOutputStream
FileWriter
Java application
File (images, .zip, 8-bit text)
FileInputStream
FileReader
File (text)
(Read bytes from file)
(Read Unicode chars from file)

Buffering

- In vielen Fällen wird nach einem write nicht sofort geschrieben sondern es wird gewartet, bis sich eine gewisse Menge von Daten in Puffer (engl. Buffer) angesammelt hat.
- Mit der flush-Methode kann man das Schreiben erzwingen
- public void flush()
- Schreiben aller noch ausständigen Daten, alles was sich durch Aufrufe von write angesammelt hat, aber noch nicht geschrieben worden ist.

Buffered Streams



Streams (High-Level)

- BufferedReader
- BufferedWriter
- ObjectInputStream
  - Deserialisierung von Java-Objekten
- ObjectOutputStream
  - Serialisierung von Java-Objekten

Streams (High-Level)

- InputStreamReader
  - Liest Zeichen aus einem InputStream
- OutputStreamWriter
  - Schreibt Zeichen in ein OutputStream
- PrintStream
  - Schreibt formatierte Representationen von Java-Objekten in ein Binary-Stream
- PrintWriter
  - Schreibt formatierte Representationen von Java-Objekten in ein Zeichenbasierten-Output-Stream

Buffered Streams

- Decorator Klassen
  - BufferedInputStream
  - BufferedOutputStream

Data Streams

- Lesen und Schreiben primitiver Datentypen
- Verwendet FileInputStream/FileOutputStream
  - DataInputStream
  - DataOutputStream

Object Streams

- Lesen und Schreiben von Objekten
- Verwendet FileInputStream/FileOutputStream
- Objekte müssen serialisierbar sein
  - ObjectInputStream
  - ObjectOutputStream

Reader

- implementiert AutoCloseable
  - FileReader
  - BufferedReader

Writer

- implementiert AutoCloseable
- FileWriter
- BufferedWriter

PrintWriter

- Ermöglich ein formatiertes Schreiben
  - write
  - print
  - println
  - format
  - printf

Console

- Ermöglicht das Lesen und Schreiben von und auf der Console

- Kein public Konstructor, wird über System.console() geholt

Console

- Console format(String fmt, Object... args)
    - Schreibt einen formatierten String in die Console
- Console printf(String format, Object... args)
    - Übliche Methode für formatierte Ausgabe
- String readLine()
    - Liest eine Zeile von der Console
- String readLine(String fmt, Object... args)
    - Ein formatierter Prompt. Liest eine Zeile von der Console
- char[] readPassword()
    - Liest ein Password (es gibt keine Ausgabe)
- char[] readPassword(String fmt, Object... args)
    - Ein formatierter Passwort Prompt (es gibt keine Ausgabe)

File (io) Problem

- Methoden von File werfen keine Exceptions
- Größere Verzeichnis-Auflistungen können das System ausbremsen
- Abfrage von Metadaten ist rudimentär
- Arbeit mit symbolischen Links ist nicht möglich

Path Interface

- representer ein Pfad zu einer Datei oder Verzeichnis in einem Filesystem
- Kann aus mehreren Verzeichnissen, einen Dateinamen, oder beiden bestehen
- Ist nicht an eine existierende Datei oder ein Verzeichnis gebunden
- system-dependent
- ist immutable
- erweitert Comparable, Iterable, und Watchable
- unterstützt SymLinks

Path erzeugen

- java.nio.file.**Paths**
  - public static Path get(String first, String... more)
  - public static Path get(URI uri)
- java.nio.file.FileSystem
  - public abstract Path getPath(String first, String... more)
- java.io.File
  - public Path toPath()

Optionen für Methoden

- ATOMIC_MOVE
  - Atomares verhalten, Exception, wenn nicht unterstützt
- COPY_ATTRIBUTES
  - Attribute werden mitkopiert
- REPLACE_EXISTING
  - ersetzt Datei, Exception, wenn es nicht möglich ist
- NOFOLLOW_LINKS
  - SymLinks nicht folgen
- FOLLOW_LINKS
  - SymLinks folgen

Path Methoden

- Viele Path-Methoden liefern ein Path zurück und können daher verkettet werden

- Methoden mit Positionsangaben können eine IllegalArgumentException werfen

- getName, getNameCount, und subpath beinhalten kein root Verzeichnis

Path Methoden

- getFileName
  - liefert den Namen der Datei oder Verzeichnisses zurück
- getParent
  - Liefert den Eltern-Path oder null zurück
- getRoot
  - Liefert den Root als Path oder null zurück

Path Methoden

- toAbsolutePath
    - liefert den absoluten Pfad
- normalize
    - eliminiert redundant Namens-Elemente im Pfad
- toRealPath
    - löst eine Ausnahme aus, wenn die Auflösung eines Pfades auf eine Datei zeigt, die nicht existiert
    - Unterstützt als einzige Methode die NOFOLLOW_LINKS Option
    - verwendet implizit normalize
    - kann eine IOException werfen

Path Methoden

- relativize
  - Erzeugt einen Pfad zwischen zwei Path Objekten
  - Relativ und absolut darf nicht gemischt werden sonst wird eine IllegalArgumentException geworfen
  - Unter Win muss der Root gleich sein, sonst wird ebenfalls eine IllegalArgumentException geworfen

Path Methoden

- resolve(String), resolve(Path)
- resolveSibling(String), resolveSibling(Path)
  - Verknüpft Pfade
  - überprüft nicht die Richtigkeit auf dem Filesystem
  - Das Ergebnis ist nicht normalisiert

Path Methoden

- lexikographisch compareTo(Path)
- startsWith(String), startsWith(Path)
- endsWith(String), endsWith(Path)
- konvertieren den String-Parameter vor dem Vergleich in ein Path
- Kein Stringvergleich sondern Pfadvergleich

Files

- statische Methoden für die Manipulation von Dateien und Verzeichnissen
- arbeitet mit echte Dateien und Verzeichnisse auf dem Filesystem
- createFile(Path, FileAttribute…)
  - prüft die Existenz der Datei
  - FileAlreadyExistsException wenn Datei bereits existiert
- createDirectory, createDirectories

Files

- isSameFile
  - Prüft die Gleichheit von zwei Path Objekten

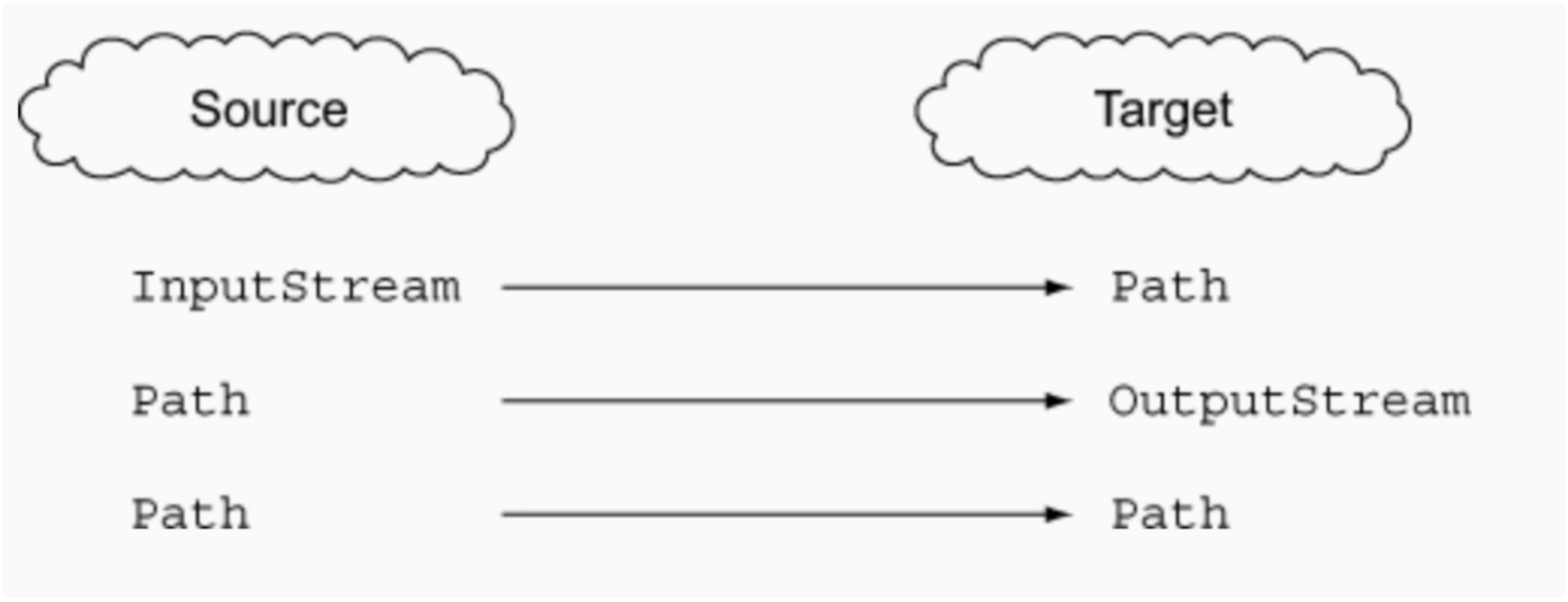
Files

- exists(Path, LinkOption...)
- notExists(Path, LinkOption…)
  - liefern beide false, wenn die Datei nicht überprüft werden kann
  - Default: folgen den symbolischen Links
  - LinkOption.NOFOLLOW_LINKS

Files

- Files.copy()
  - kopiert Dateien oder Verzeichnisse
  - Inhalt eines Verzeichnisses wird nicht kopiert
- StandardCopyOption
  - ATOMIC_MOVE
    Kopiert atomar
  - COPY_ATTRIBUTES
    Versucht alle Attribute zu kopieren
  - REPLACE_EXISTING
    ersetzt Dateien mit gleichen Namen

copy (Streams)



Files

- move
  - nicht leere Verzeichnisse können nur innerhalb des gleichen Laufwerks bewegt werden
  - Verzeichnis kann umbenannt werden
  - Datei oder Verzeichnis kann nicht in ein nicht existierendes Verzeichnis verschoben werden
  - Wirft eine IOException, wenn Datei oder Verzeichnis existiert (Kann mit StandardCopyOptions geändert werden)

Files delete

- delete, deleteIfExists
  - löscht Dateien und leere Verzeichnisse (DirectoryNotEmptyException) löschen
  - delete wirft eine Exception, wenn Datei nicht vorhanden.
  - deleteIfExists liefert ein boolean zurück

Files

- newBufferedReader(Path, Charset), newBufferedWriter(Path, Charset)

Files

- readAllLines(Path, [Charset])
  - Liefert die Textzeilen als eine geordnete List von Strings
  - Die gesamte Datei wird in den Speicher geladen

Attribute

- isDirectory()
  isRegularFile()
  isSymbolicLink()

- kann auf Dateien, Verzeichnissen und Symlinks angewandt werden

- werfen keine Exceptions, wenn der Path nicht existiert

Attribute

- isHidden()
  size() // in bytes
  - kann eine IOException werfen

- isReadable()
  isExecutable()
  - werfen keine Exceptions

Attribute

- getLastModifiedTime()
  setLastModifiedTime()
- arbeitet mit FileTime
- wirft eine IOException

Attribute

- getOwner()
setOwner()
- arbeitet mit UserPrincipal
- wirft eine IOException

Views

- BasicFileAttributes
  DosFileAttributes
  PosixFileAttributes

- Wenn ein View nicht unterstützt wird wirft Files.getFileAttributeView() und Files.readAttributes(Path,Class<A>) eine IOException

Views

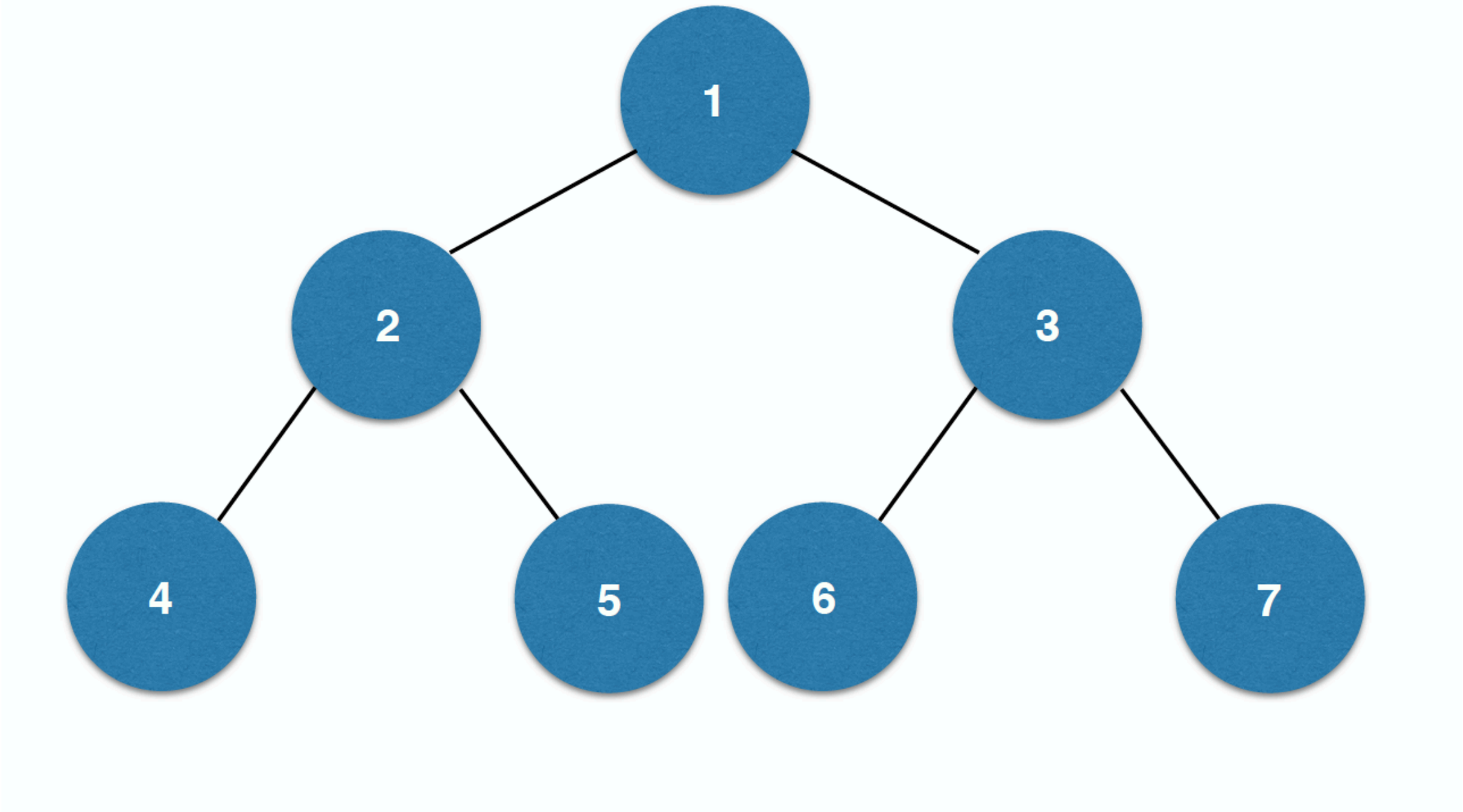
- isOther()
- lastAccessTime()
- creationTime()
- fileKey()

Verzeichnisse

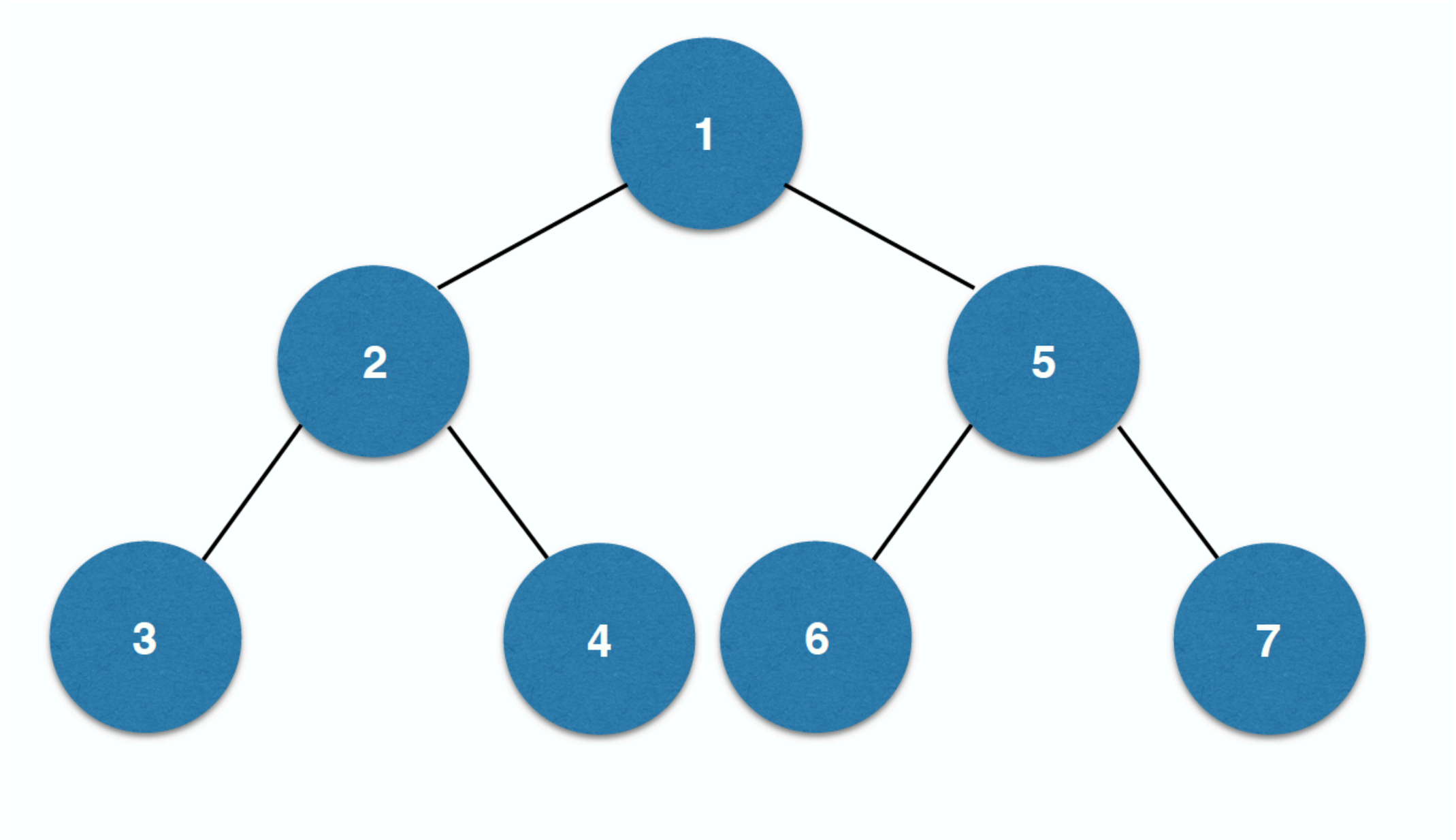
- Files.walk(Path)
  .walk(Path, int)

  - liefert ein Stream<Path>
  - depth-first, lazy
  - wirft eine IOException
  - default Tiefe Integer.MAX_VALUE
  - folgt per default nicht SymLinks

breadth-first



depth-first



Suche

- Files.find(Path,int,BiPredicate)
- liefert ein Stream<Path> zurück
- wirft eine IOException

Auflisten

- Files.list(Path)
  - liefert ein Stream<Path> zurück
  - wirft eine IOException
  - liest nur 1 Level aus

Inhalt ausgeben

- Files.lines(Path)
  - liefert ein Stream<String> zurück
  - wirft eine IOException