Java Advanced

Lezen en schrijven (I/O)

Hoofdstuk 5

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Mappen en bestanden

File separator OS

- Unix
  - /data/folder1/file1.txt
- Windows
  - C:\data\folder1\file1.txt

Mappen en bestanden

Relatief en absoluut path

- Relatief = t.o.v. een ander path
  - ../folder1/file1.txt
  - ./folder1/file1.txt
- Absoluut is het volledige path
  - C:\data\folder1\file1.txt

Mappen en bestanden

Goed om weten:

- < JDK 1.7: Toegang tot bestanden -> *File.class*
- > JDK 1.7: Toegang tot bestanden -> *Path.class*

Mappen en bestanden

Paths is een impl van het Factory Design Pattern

- Windows:
  - Path path = Paths.get("C:\\data\\folder1\\file1.txt");
  - Windows is niet hoofdlettergevoelig!
- Unix:
  - Path path = Paths.get("/data/folder1/file1.txt");

De klasse Paths

Bekijk de Java API doc voor Path en Paths.

Enkele voorbeelden van het gebruik:

```
Path p1 = Paths.get("C:\\data");
Path p2 = p1.resolve("folder1");
Path p3 = p2.resolve("file1.txt");
System.out.println(p3); // C:\data\folder\file1.txt

Path p4 = Paths.get("file2.txt");
System.out.println(p4.toAbsolutePath());
System.out.println(p4.toRealPath());
```

De klasse Paths

Relatieve path t.o.v. elkaar:

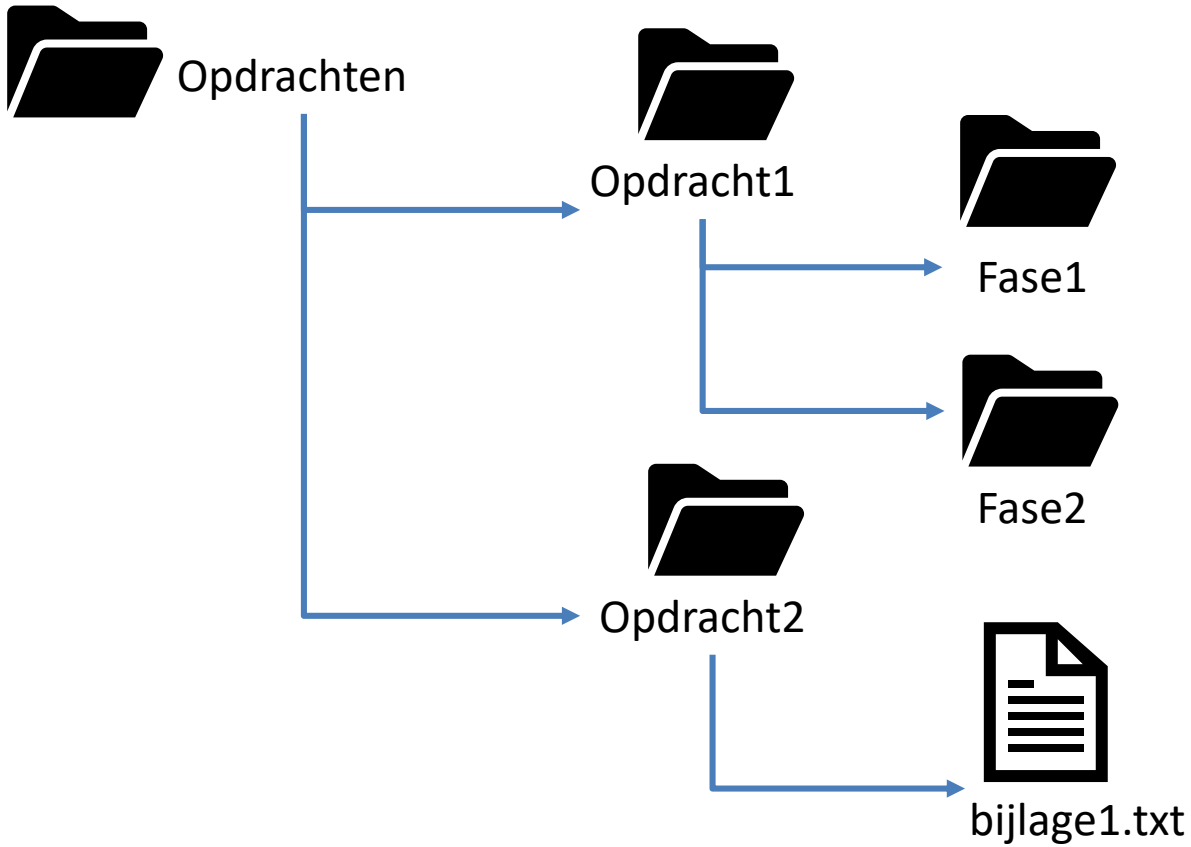
```
Path p5 = Paths.get("C:\\data\\subfolder1\\file1.txt");
Path p6 = Paths.get("C:\\data\\subfolder2\\file3.txt");
Path p7 = p5.relativize(p6);

System.out.println(p7); // ..\..\subfolder2\file3.txt
```

Om te starten…

Druk de waarde van systeemeigenschap user.home af met

System.getPropery(…)

In deze directory maken we de volgende folder-structuur aan



Paths: oefening 1

1. Vorm de systeemeigenschap user.home om tot een Path object
2. Welke concrete klasse heeft dit Path object?
3. Gebruik de methode resolve() van het Path object om vanuit de user.home directory het path “Opdrachten/Opdracht1/Fase2” te volgen

Paths: oefening 2

Wat is het resultaat wanneer je volgende methoden uitvoert op het laatst geconstrueerde Path?
toString()
getFileName()
getName(0)
getNameCount()
subpath(0,2)
getParent()
getRoot() EXTRA: wat is de root van een relatief path?

De klasse FileSystem

Stelt het bestandssysteem voor:

- String getSeparator()
- Iterable<Path> getRootDirectories()

De klasse Files

Bekijk de Java API doc voor Files

- *Path* verwijst naar een pad
- *Files* verwijst naar een bestand
- De klasse *Files* is een utility klasse

Files: oefening 1

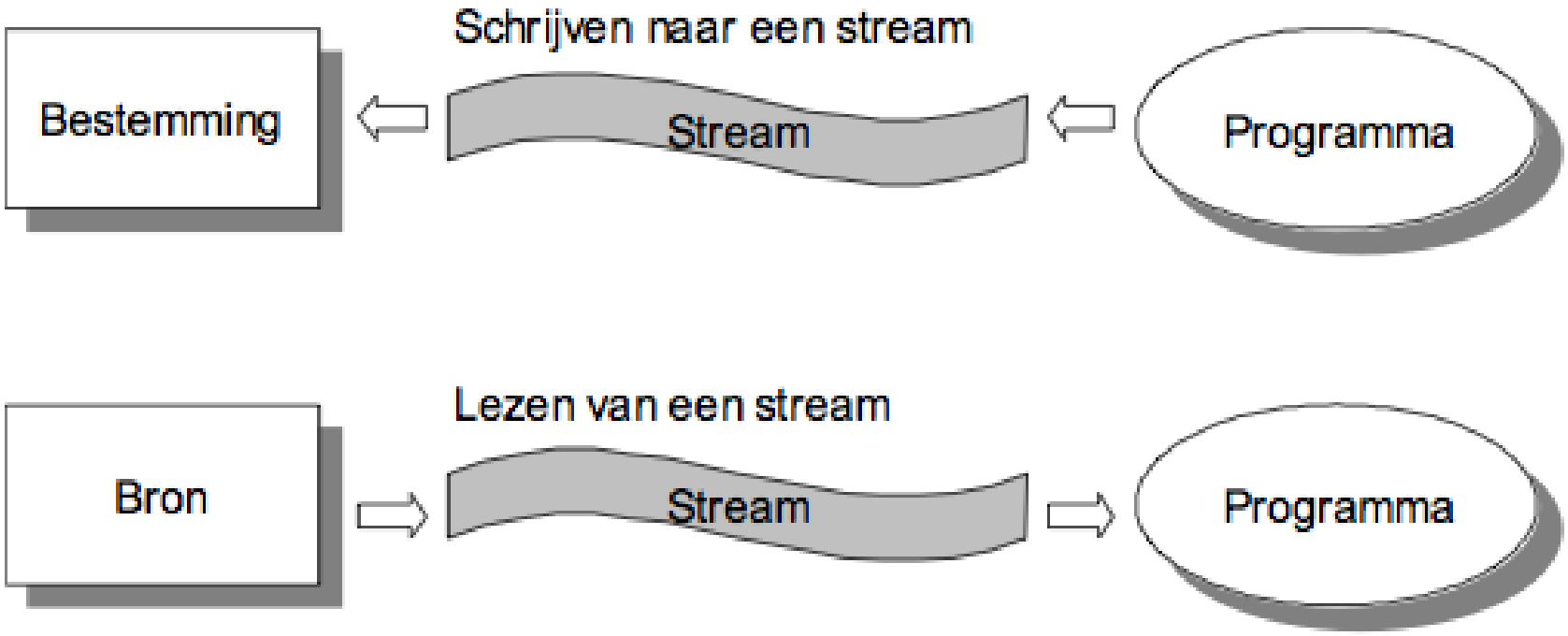
1. Lees het bestand bijlage1.txt in door gebruik te maken van een methode uit de utility klasse Files. Iedere regel van het bestand bevat 1 woord, sommige woorden kunnnen meerdere keren voorkomen.
2. Kies een gegevensstructuur om de woorden in op te slaan, zodat je ze makkelijk alfabetisch kan sorteren en dubbels worden verwijderd.
3. Schrijf nu alle waarden in de gegevensstructuur naar het bestand *output.txt*. Indien dit bestand reeds bestaat, verwijder je het eerst.

Files: oefening 2

Uitbreiding:

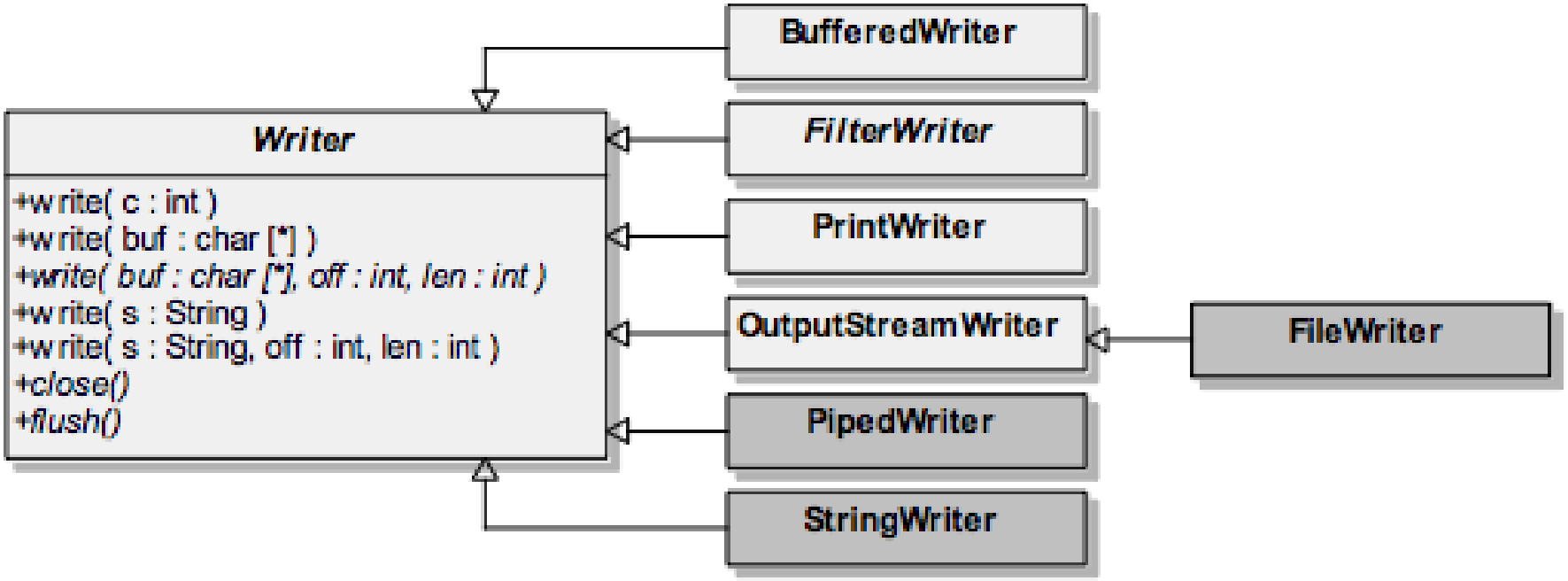
1. Tijdens het inlezen van de woorden, ga je tellen hoeveel keer een woord voorkomt.
2. Nadat de woorden alfabetisch gesorteerd zijn weggeschreven, schrijf je de woorden nog een tweede keer weg in hetzelfde bestand, maar dan gesorteerd op hun aantal voorkomens. Het woord dat het vaakst voorkomt staat eerst (met het aantal voorkomens), vervolgens het tweede meest voorkomende, …

Streams



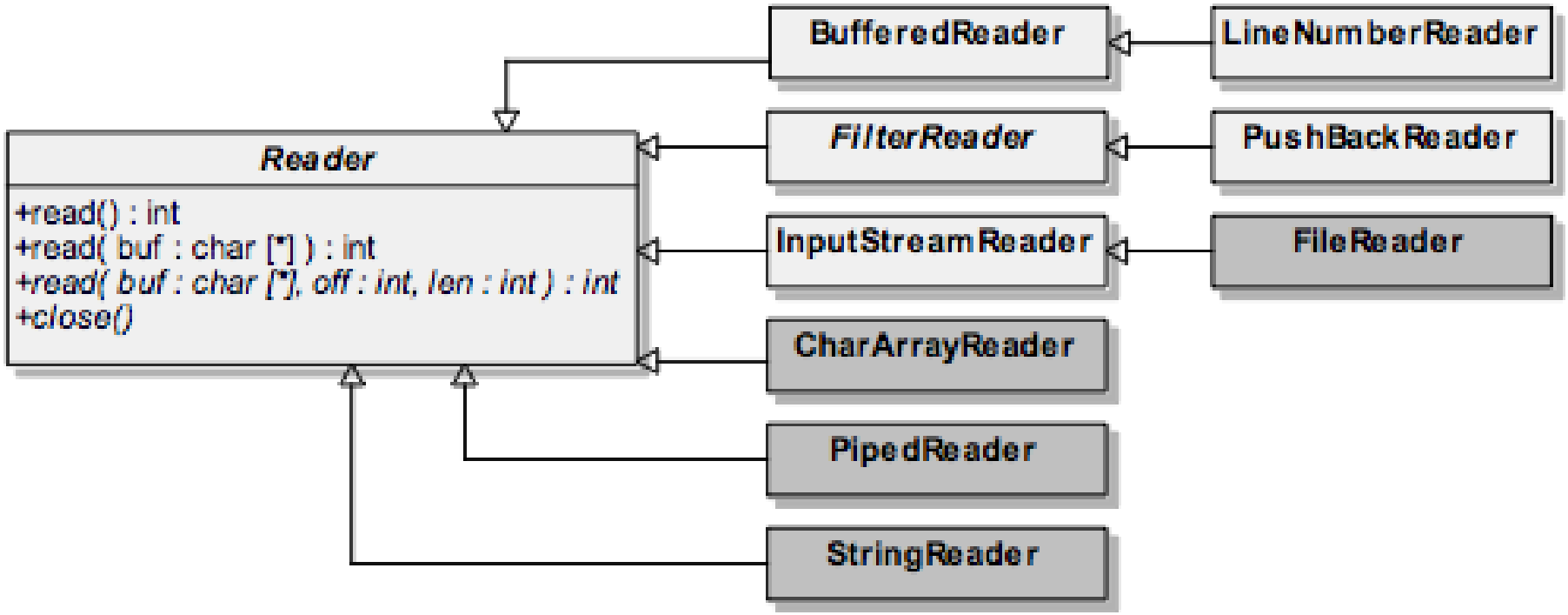
Afbeelding 21: Streams

Character streams



Afbeelding 23: Character streams: Writer

Character streams



Afbeelding 24: Character streams: Reader

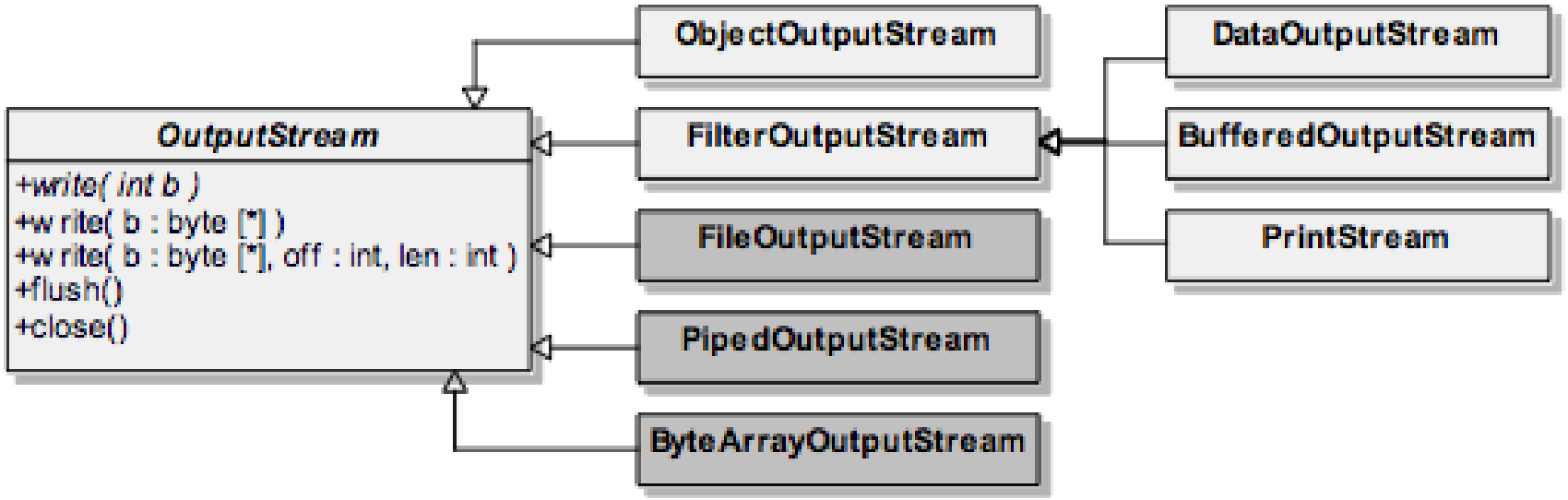
Character streams write example

```
import java.io.*;

public class WriteFileAutoClose {
   public static void main(String[] args) {
      try (FileWriter file = new FileWriter("MyFile.txt"))
      {
          file.write("Hello");
          file.write("World");
      } catch (IOException ex) {
          System.out.println("Ooops, something went wrong");
          System.out.println(ex.getMessage());
      }
    }
}
```

Character streams read example

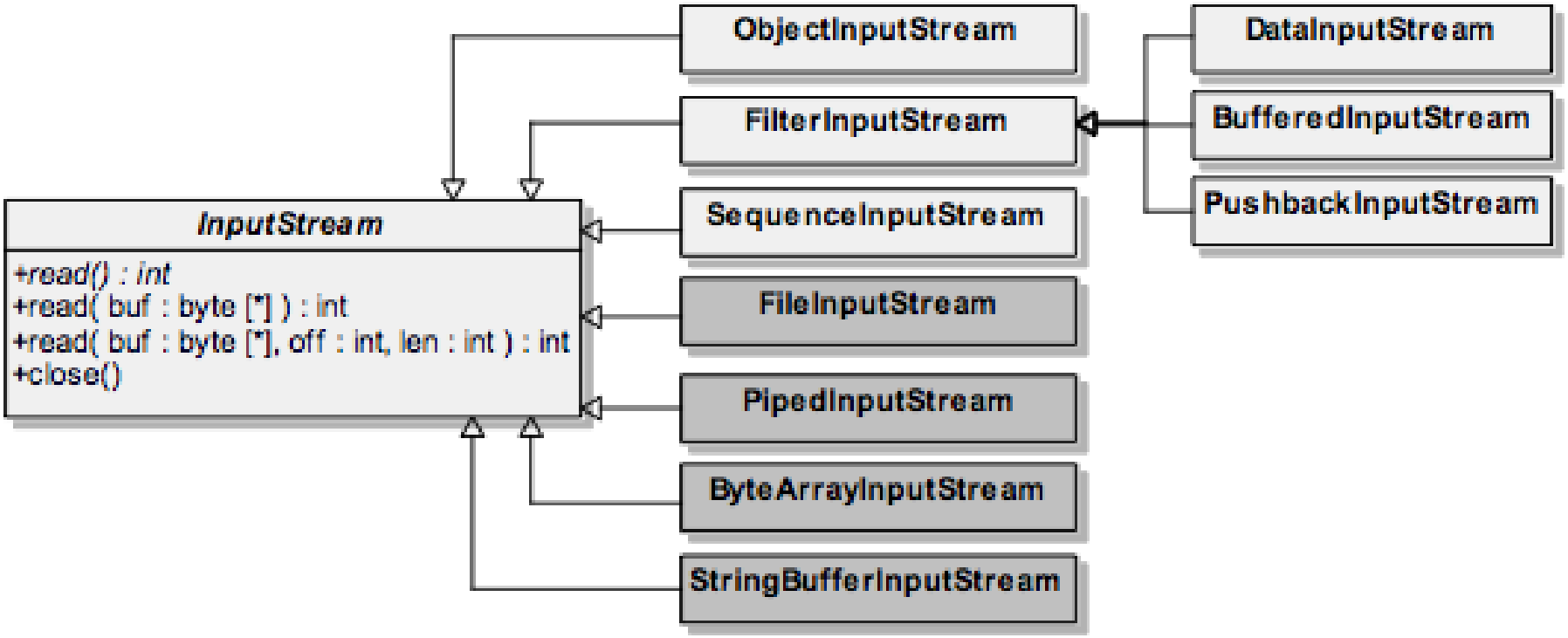
```
public class ReadFile2 {
  public static void main(String[] args) {
  Path path = Paths.get("MyFile.txt");
  try(BufferedReader reader = Files.newBufferedReader(path)) {
       String line = null;
       while ((line = reader.readLine()) != null) {
         System.out.println(line);
       }
    } catch (IOException ex) {
       System.out.println("Oops, something went wrong!");
       System.out.println(ex.getMessage());
    }
  }
}
```

Byte streams



Afbeelding 27: Byte streams: OutputStream

Byte streams



Afbeelding 28: Byte streams: InputStream

Streams Encodering en Karaktersets

```
FileOutputStream out = new
    FileOutputStream("MyEncodedFile.txt");

OutputStreamWriter writer = new OutputStreamWriter(out, "UTF-8");
```

Streams: oefening 1

Schrijf een programma om telefoonnummers bij te houden.

Wanneer je het programma opstart wordt het bestand “phonedirectory.txt” ingelezen. Ieder lijn in het bestand heeft het volgende formaat: <naam>;<tel1>;<tel2>;…

1. Lees alle lijnen in zodat je aan de hand van een naam de bijhorende telefoonnummers kan zoeken.
2. Zorg ook voor functionaliteit om telefoongegevens toe te voegen. Als je telefoonnummers voor een reeds bestaande persoon toevoegt, gooit het programma een zelfgemaakte exception.
3. Wanneer je ervoor kiest om het programma af te sluiten, worden alle gegevens eerst weer weggeschreven in “phonedirectory.txt” zodat geen gegevens verloren gaan.
4. **Uitbreiding:** onderzoek het gebruik van programma-attributen om de locatie en naam van het bestand in te stellen

Programma attributen

- naam=waarde
- store() en load() methoden
- Xml support

Programma attributen write

```
import java.io.*;
import java.util.*;
public class WriteProperties {
   public static void main(String[] args) {
      try (FileOutputStream out = new
                  FileOutputStream("Application.properties");) {
         Properties atts = new Properties();
         atts.setProperty("Attribute1", "Value1");
         atts.setProperty("Attribute2", "Value2");
         atts.setProperty("Attribute3", "Value3");
         atts.store(out, "Application properties");
      } catch (Exception ex) {
         System.out.println(ex.getMessage());
      }
}
```

Programma attributen read

```
import java.io.*;
import java.util.*;
public class ReadProperties {
  public static void main(String[] args) {
    try (FileInputStream in =
        new FileInputStream("Application.properties");) {
      Properties atts = new Properties();
      atts.load(in);
      atts.list(System.out);
    }
    catch(Exception ex) {
      System.out.println(ex.getMessage());
    }
  }
}
```

Programma attributen file

#Application Properties

#Thu Jun 27 09:10:19 CEST 2013

Attribute3=Value3

Attribute2=Value2

Attribute1=Value1

Programma attributen file xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM
"http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>Application Properties</comment>
  <entry key="Attribute3">Value3</entry>
  <entry key="Attribute2">Value2</entry>
  <entry key="Attribute1">Value1</entry>
</properties>
```

Object Serialization write

```
import java.io.*;
import java.time.*;
public class WriteObjectApp {
  public static void main(String[] args) {
    String text = new String("This is some text");
    LocalDateTime date = LocalDateTime.now();
    try (FileOutputStream file = new
                    FileOutputStream("MyFile.ser");
       ObjectOutputStream out = new ObjectOutputStream(file);){
      out.writeObject(text);
      out.writeObject(date);
    }
    catch(IOException ex) {
       System.out.println(ex.getMessage());
    }
  }
}
```

Object Serialization read

```
import java.io.*;
import java.time.*;
public class ReadObjectApp {
  public static void main(String[] args) {
    try (FileInputStream file = new
                    FileInputStream("MyFile.ser");
        ObjectInputStream in = new ObjectInputStream(file);) {
            String text = (String) in.readObject();
            LocalDateTime date = (LocalDateTime) in.readObject();
            System.out.println(text); System.out.format("%td/%<tm/%<tY %<tH:%<tM:%<tS%n",date);
    } catch (Exception ex) {
      System.out.println(ex.getMessage());
    }
  }
}
```

Eigen objecten serializeren

- Import java.io.*
- … implements Serializable
- **transient** property is not serialized !
- Versienummering (serialVersionUID)

Serialization: oefening

Schrijf een applicatie voor het beheren van spaarrekeningen.

Voorzie volgende functionaliteit:

- Maak een klasse Spaarrekening met enkele member variabelen (bv. saldo, naam, id, …)
- Zorg dat instanties van deze klasse als object naar een file geschreven kunnen worden (Serializable)
- Schrijf in een main methode de code om een Spaarrekening object naar een file weg te schrijven
- Voeg code toe om een spaarrekening object terug uit te lezen uit deze file
- Controleer of de member variabelen hun waarde hebben behouden na uitlezen