



Java Advanced

File I/O

DE HOGESCHOOL MET HET NETWERK

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



Inhoud

- Mappen, bestanden, paden
- De klasse *Files*
- IO-Streams
- Programma attributen
- Object serialization



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
(bv. huidige directory of project folder)
 - ../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*
 - Package: `java.io`
- **> JDK 1.7:**
 - Toegang tot bestanden -> *Path.class*
 - *Package: java.nio.file*



Mappen en bestanden

Path.of() gebruikt achterliggend de methode *getPath()* van de klasse *FileSystem*.

FileSystem is een implementatie van het Factory Design Pattern.

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



De interface Path

Bekijk de Java API doc voor Path.

Enkele voorbeelden van het gebruik:

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

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



De interface Path

Relatieve path t.o.v. elkaar:

```
Path p5 = Path.of("C:/data/subfolder1/file1.txt");
```

```
Path p6 = Path.of("C:/data/subfolder2/file3.txt");
```

```
// Schrijf het pad naar p6 relatief t.o.v. p5
```

```
Path p7 = p5.relativeTo(p6);
```

```
System.out.println(p7);
```

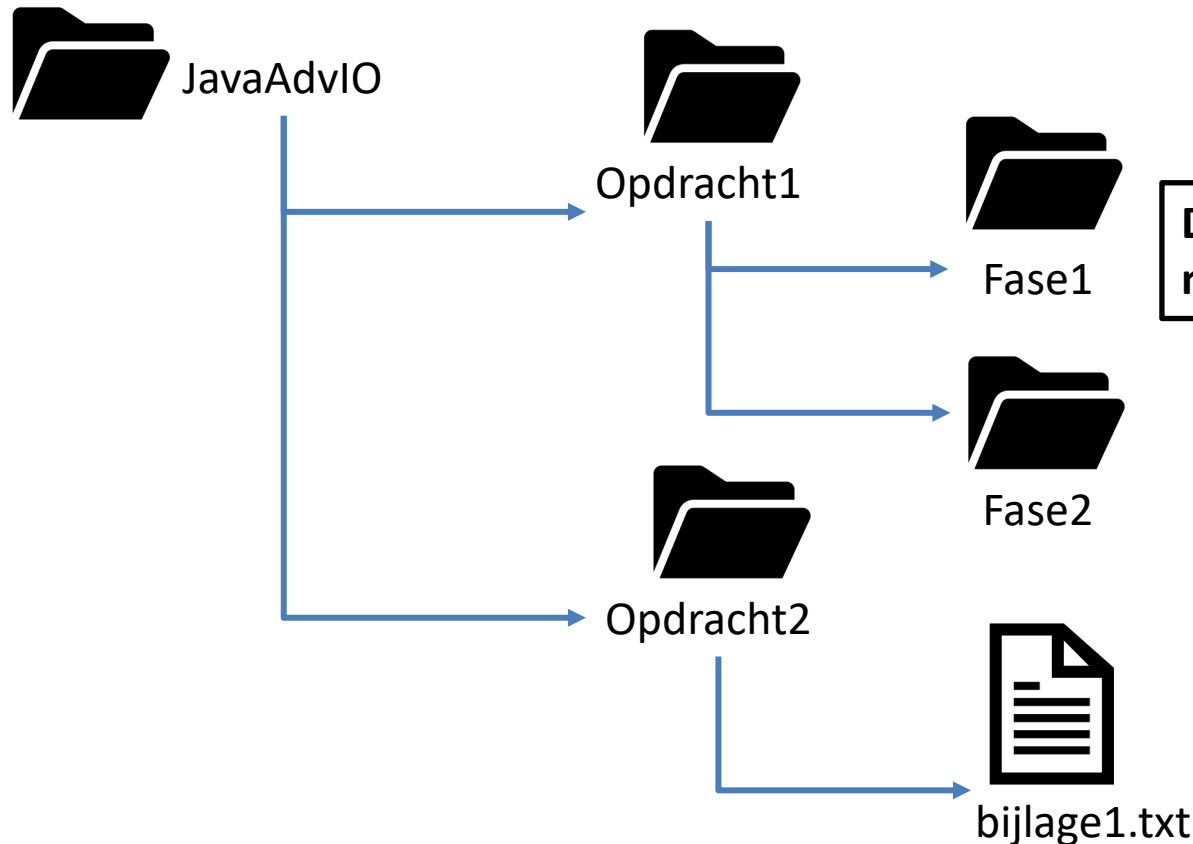
```
// ../../subfolder2/file3.txt
```



Om te starten...

Druk de waarde van systeemeigenschap “*user.home*” af met *System.getProperty(...)*

In deze directory maken we de volgende folder-structuur aan:



Doe dit in Explorer, niet met code

Path: 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 “`JavaAdvIO/Opdracht1/Fase2`” te volgen



Path: 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()

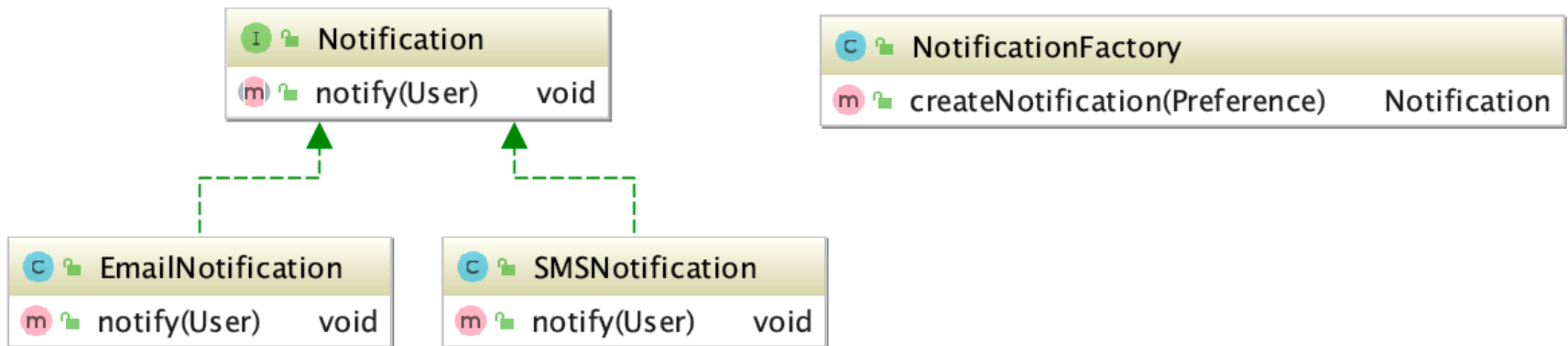
De klasse `FileSystem`

Stelt het bestandssysteem voor:

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



Factory design pattern (demo)



De klasse Files

Bekijk de Java API doc voor Files

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



Files utility class

Voorbeeld:

File aanmaken indien deze niet bestaat

```
if (!Files.exists(path)) {  
    try {  
        Files.createDirectories(path.getParent());  
        Files.createFile(path);  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```




Files utility class

Voorbeeld:

File aanmaken indien deze niet bestaat

```
try {  
    Files.createDirectories(path.getParent());  
    Files.createFile(path);  
} catch (IOException e) {  
    e.printStackTrace();  
}
```



Files utility class

Voorbeeld:

Alle folders in directory printen

```
Path scan = Path.of(System.getProperty("user.home"))
                .resolve("Opdrachten");

try (DirectoryStream<Path> subfolders = Files.newDirectoryStream(scan)) {
    for (Path folder : subfolders) {
        System.out.println(folder.getFileName());
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

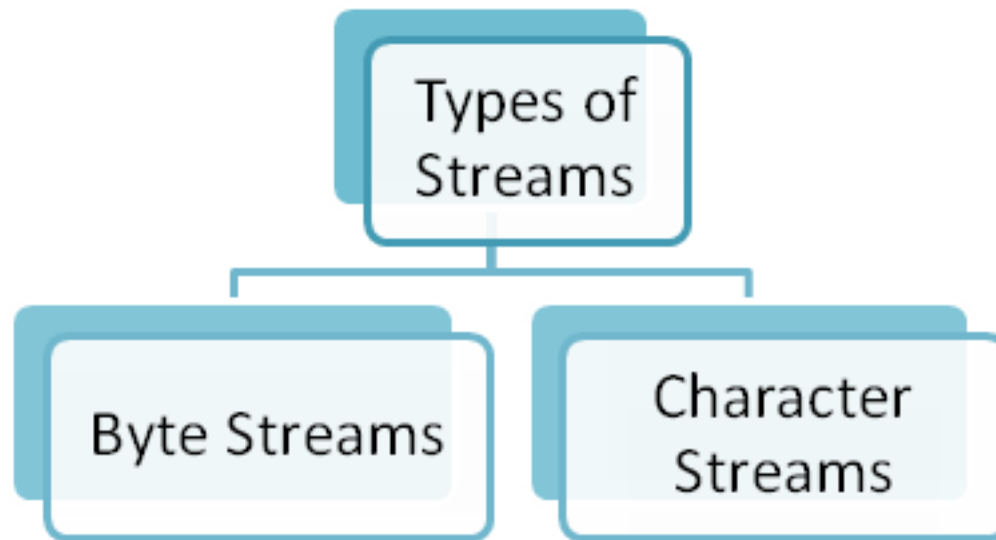


Files: oefening 1

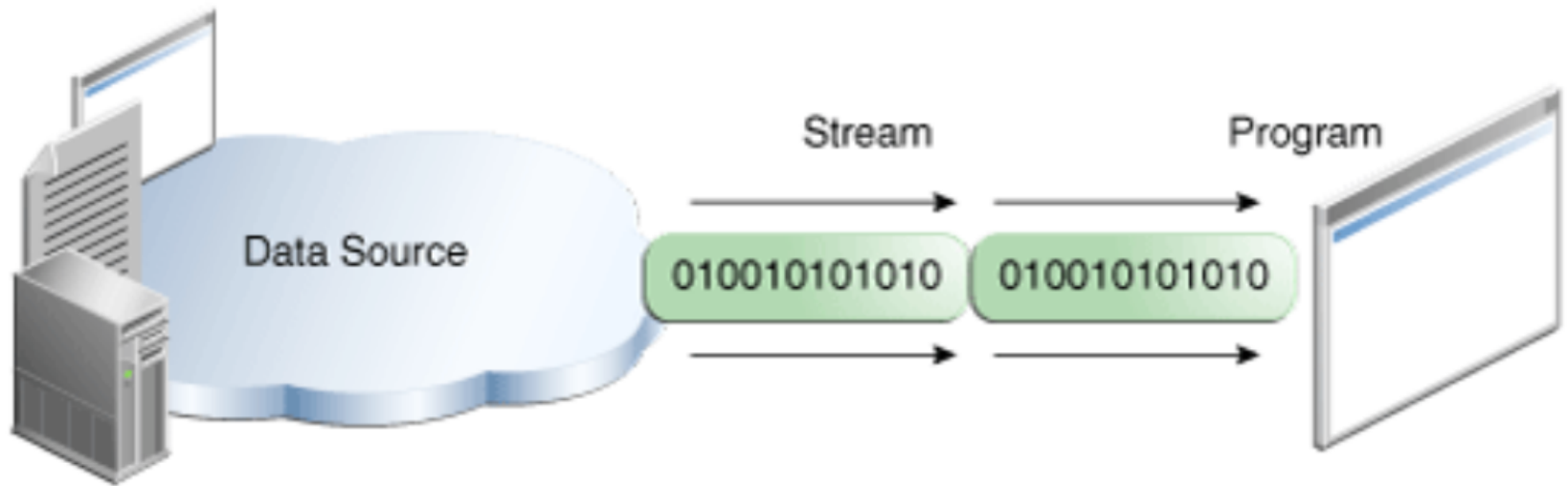
1. Lees het bestand *bijlage1.txt* in door gebruik te maken van een methode uit de utility klasse Files. Iedere regel van het bestand bevat 1 woord, sommige woorden kunnen meerdere keren voorkomen.
2. Kies een Collection om de woorden in op te slaan, zodat je ze makkelijk alfabetisch kan sorteren en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



IO-streams

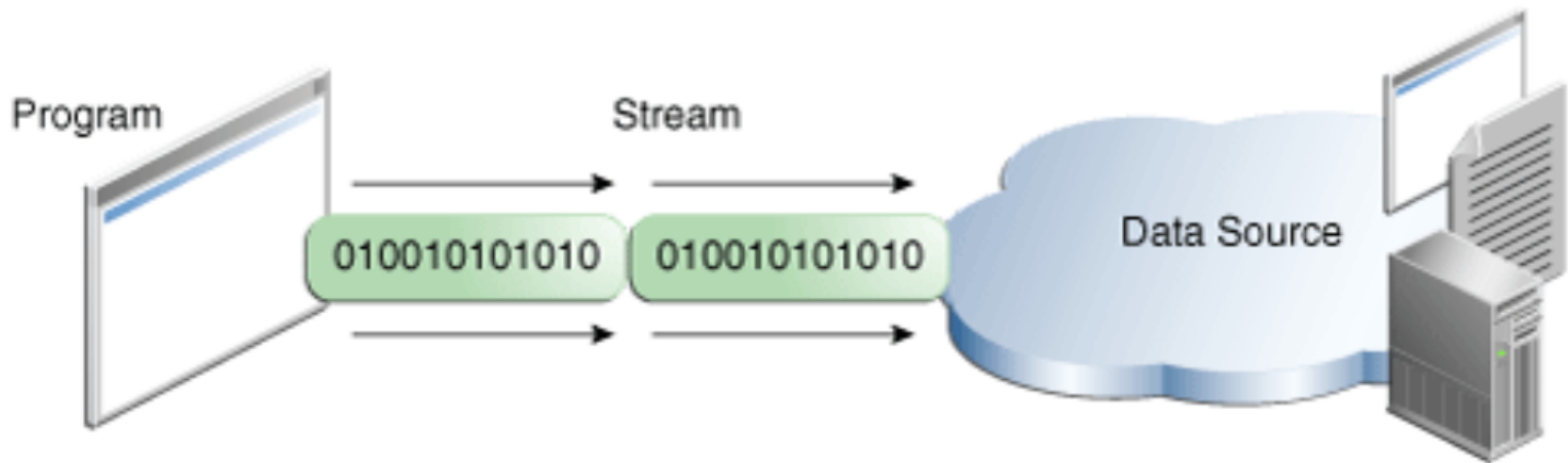


Byte streams



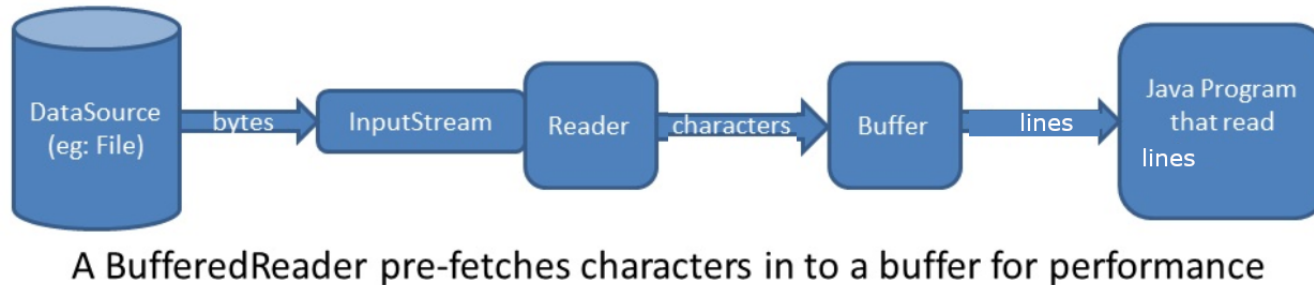
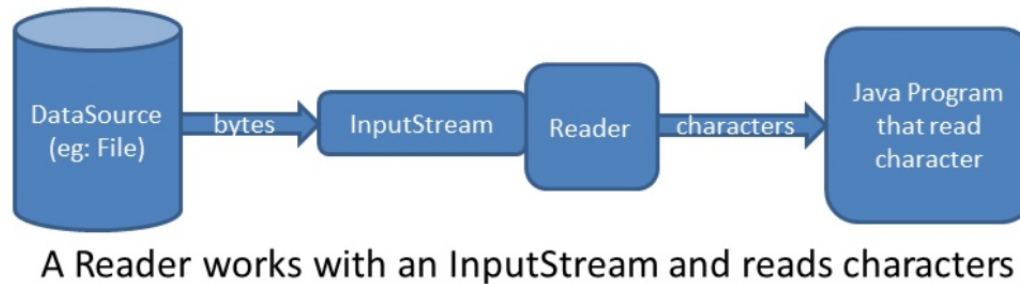
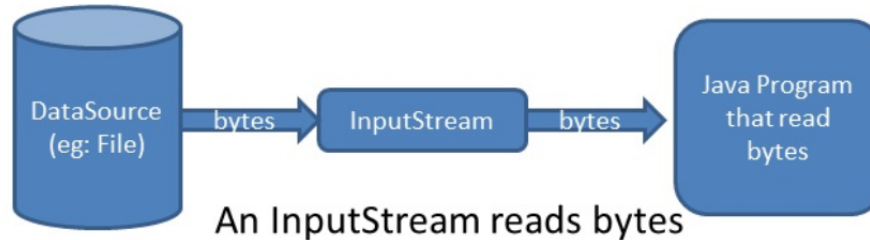
Reading information into a program.

Byte streams



Writing information from a program.

IO-streams



Character streams read example

```
public class ReadFile {  
    public static void main(String[] args) {  
        Path path = Path.of("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());  
        }  
    }  
}
```



Character streams read example

```
public class ReadFile {  
    public static void main(String[] args) {  
        Path path = Path.of("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());  
        }  
    }  
}
```

Try-with-resources

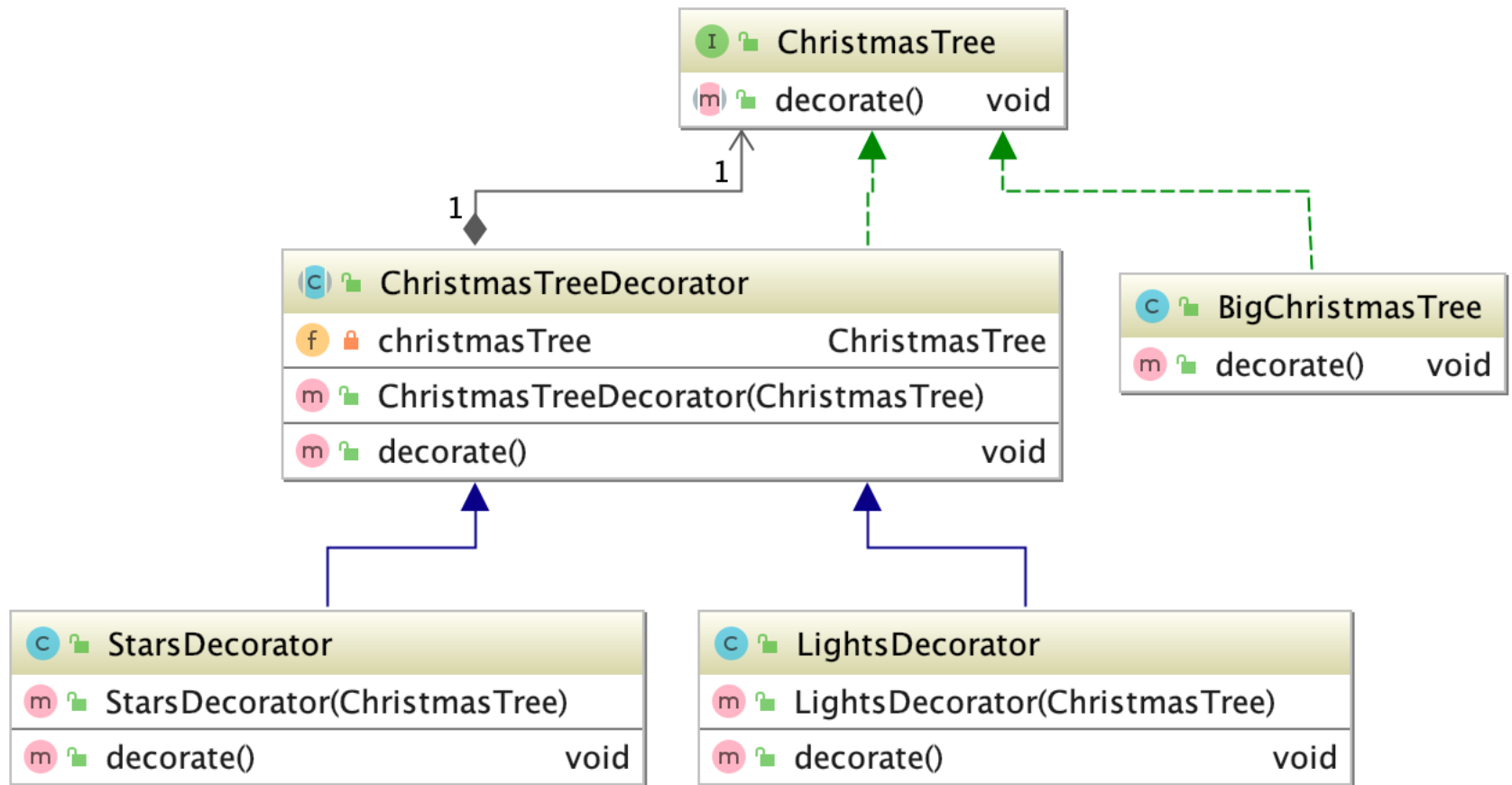


Character streams write example

```
public class ReadFile {  
    public static void main(String[] args) {  
        Path path = Path.of("MyFile.txt");  
        try(BufferedWriter writer = Files.newBufferedWriter(path)) {  
            String line = null;  
            writer.write("Hallo");  
            writer.write(System.lineSeparator());  
            writer.write("Tweede regel");  
        } catch (IOException ex) {  
            System.out.println("Oops, something went wrong!");  
            System.out.println(ex.getMessage());  
        }  
    }  
}
```



Decorator design pattern (demo)



Streams Encoding en Karaktersets

```
FileOutputStream out = new  
    FileOutputStream("MyEncodedFile.txt");  
  
OutputStreamWriter writer = new OutputStreamWriter(out, "UTF-  
8");
```



IO-streams: oefening 1

Gegeven het bestand ***code.code***

In het bestand staan ongeveer 25000 **strings** die bestaan uit 2 tot 9 karakters, telkens gescheiden door een spatie of een new line.

Lees de file in en filter de strings er uit die **enkel uit hoofdletters** bestaan. Plaats deze strings achter elkaar om de code te vormen.



IO-streams: oefening 1

Gegeven het bestand ***code.code***

In het bestand staan ongeveer 25000 strings die bestaan uit 2 tot 9 karakters, telkens gescheiden door een spatie of een new line.

Lees de file in en filter de strings er uit die **enkel uit hoofdletters** bestaan. Plaats deze strings achter elkaar om de code te vormen.

De eerste student die de code kent, wint!



Programma attributen

Opslaan en inladen van specifieke waarden/attributen voor programma (settings, configuratie, ...)

- 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



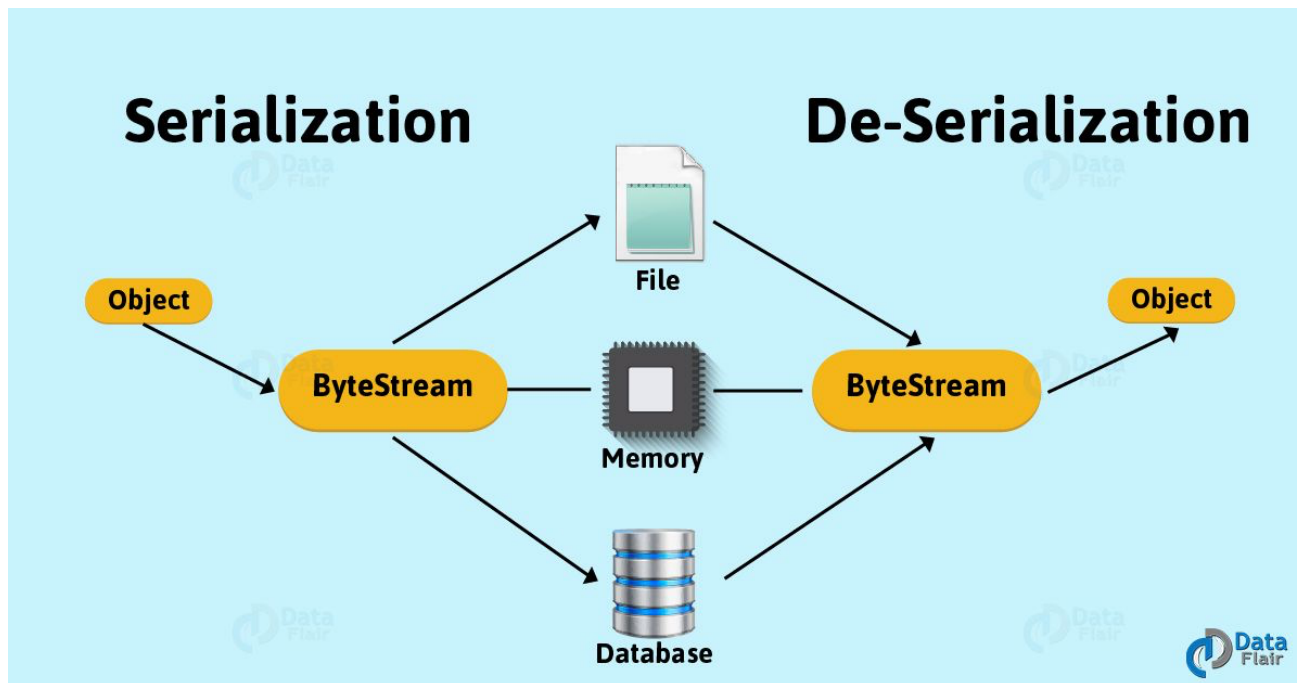
Programma attributen file xml

```
<?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

- Volledig object binair wegschrijven en inladen
- Geen aparte waarden opslaan
- Volledige instantie (eigenschappen) opgeslagen



Object Serialization write

```
import java.io.*;
import java.math.BigDecimal;
import java.util.*;

public class SerializationDemo {

    public static void main(String[] args) {
        BigDecimal value = BigDecimal.valueOf(250);
        List<String> animals = Arrays.asList("elephant", "zebra", "donkey");
        try (FileOutputStream file = new FileOutputStream("resources/demo.ser");
            ObjectOutputStream out = new ObjectOutputStream(file)) {
            out.writeObject(value);
            out.writeObject(animals);
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```



Object Serialization read

```
import java.io.*;
import java.math.BigDecimal;
import java.util.List;

public class Deserialization {

    public static void main(String[] args) {
        try (FileInputStream file = new FileInputStream("resources/demo.ser");
             ObjectInputStream in = new ObjectInputStream(file)) {
            BigDecimal value = (BigDecimal) in.readObject();
            System.out.println(value);
            List<?> animals = (List<?>) in.readObject();
            System.out.println(animals.size());
            System.out.println(animals.get(0));
        } catch (IOException ex) {
            System.out.println(ex.getMessage());
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```



Eigen objecten serializeren

```
import java.io.Serializable;  
import java.time.LocalDate;
```

```
public class Student implements Serializable {
```

```
    private String name;  
    private LocalDate graduationDate;  
    private transient LocalDate dateOfBirth;
```

```
    public Student(String name, LocalDate dateOfBirth) {  
        this.name = name;  
        this.dateOfBirth = dateOfBirth;  
    }
```

```
    // getters en setters  
}
```



Eigen objecten serializeren

- Import java.io.*
- ... implements Serializable
- Subklassen ook serializeerbaar
- **transient** property is not serialized !



Oefeningen

- Blackboard: File I/O – Oefeningen
- Gebruik try with resources !

