FACULTEIT ECONOMIE
EN BEDRIJFSKUNDE

ALGORITMEN EN DATASTRUCTUREN

UNIVERSITEIT
GENT

INHOUD

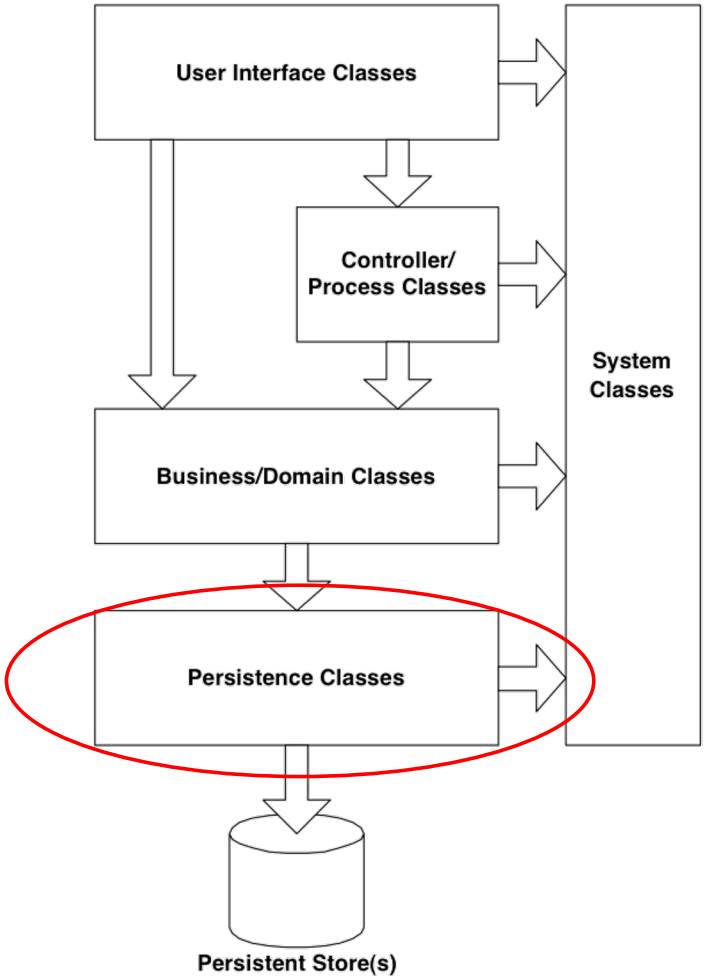
- Programmaontwerp
- Foutafhandeling
- **Bestanden input/output + databanken**
- Recursie
- Collections en Datastructuren
- Programma analyse

FACULTEIT ECONOMIE
EN BEDRIJFSKUNDE

BESTANDEN INPUT/OUTPUT

UNIVERSITEIT
GENT

ENTERPRISE SYSTEM LAYERS



PERSISTENCE LAYER

- Zorgt voor een scheiding tussen de business objects en permanente data
- Door middel van data access klassen
- Aanpassing aan de persistence layer hebben geen invloed op de business/domein objecten en logica
- 3 manieren:
  - Txt bestanden
  - Binaire bestanden
  - Databank

WAAROM FILES GEBRUIKEN

- Keyboard input en screen output worden gebruikt voor tijdelijke data
  - Wanneer programma stopt, is data weg
- Data in een bestand blijft bestaan na stoppen programma, dus kan gebruikt worden
  - in volgende uitvoering van hetzelfde programma
  - door een ander programma

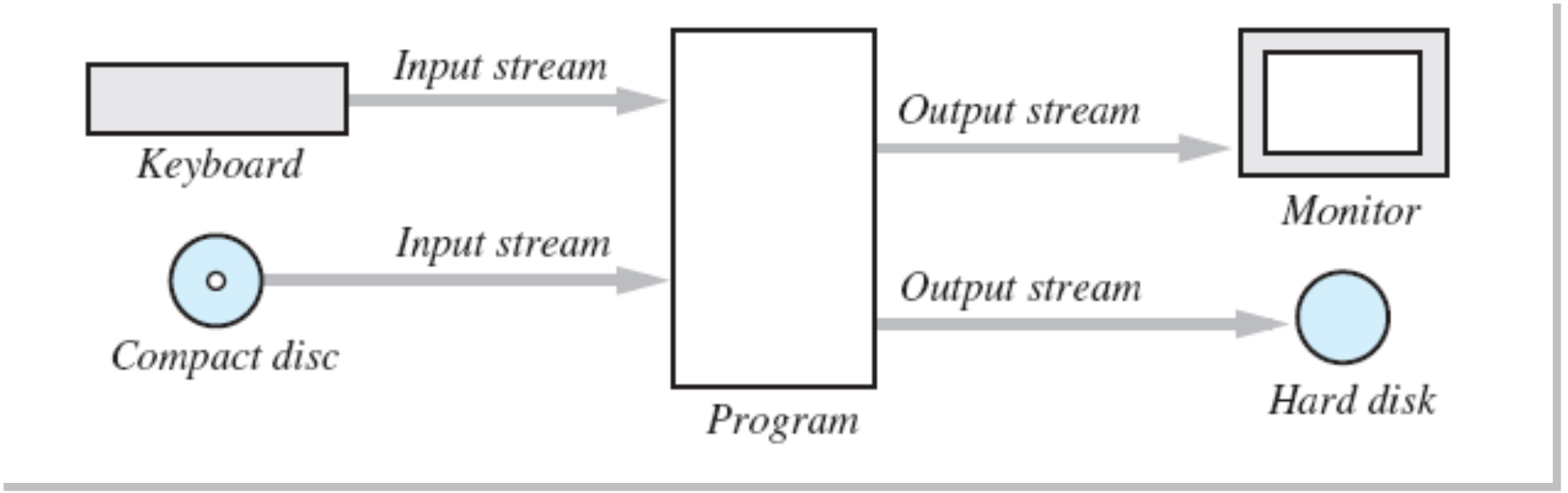
OUTLINE

- Streams + File I/O
- I/O Txt bestanden
- Werken met File Klasse
- Een csv-bestand lezen
- I/O Binaire bestanden
- GUI: Bestandsmanagement Demo

STREAM CONCEPT

- Een stream is een stroom van input of output data
  - Karakters
  - Getallen
  - Bytes
- Streams zijn geïmplementeerd als objecten van speciale Stream klassen
  - Class `Scanner` is specialisatie van Input Stream
  - Object `System.out` is object van Output Stream

STREAM CONCEPT

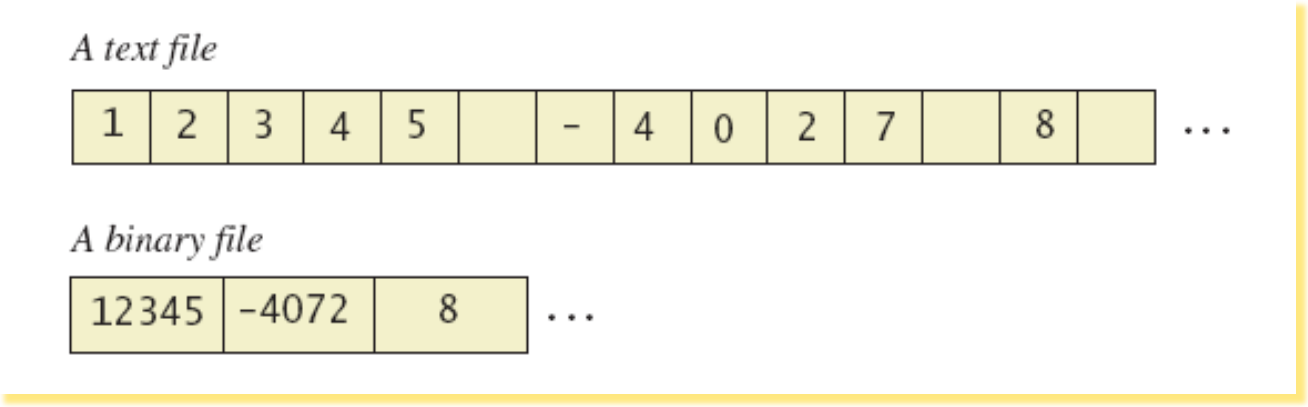


TXT VS BINAIRE FILES

- Alle data in bestanden worden opgeslagen als binaire digits
- Bestanden met sequentie van karakters, noemt men txt files:
  - Java programma source code
  - Kan men openen en aanpassen met een txt editor
- Alle andere bestanden zijn binaire bestanden
  - Movie, muziek
  - Vereist gespecialiseerd programma

TXT VS BINAIRE FILES

– Een txt file en een binaire file met dezelfde inhoud:



OUTLINE

- Java Streams en File I/O
- I/O Txt bestanden
- Werken met File Klasse
- Een csv-bestand lezen
- I/O Binaire bestanden
- GUI:

AANMAKEN EN SCHRIJVEN NAAR TXT FILE

- Klasse **`PrintWriter`** definieert methodes die nodig zijn voor het aanmaken van en schrijven naar een txt file
  - Import package **`java.io`** noodzakelijk
- Openen txt file
  - Declareer *stream variabele* om stroom te kunnen aanspreken
  - Roep **`PrintWriter`** constructor op met bestandsnaam als argument
  - **`try`** en **`catch`** blok is noodzakelijk

AANMAKEN EN SCHRIJVEN NAAR TXT FILE

- File is oorspronkelijk leeg
  - Men kan nu tekst toevoegen door gebruik te maken van methode **`println`**
- Data gaat eerst naar memory buffer.
  - Wanneer buffer vol is, wordt naar bestand geschreven
- Sluiten file maakt buffer leeg en disconnecteert de stream

AANMAKEN EN SCHRIJVEN NAAR TXT FILE

- Uitleg code:
  - Een file heeft twee namen in een programma
    - Bestandsnaam gebruikt door het besturingssysteem
    - De naam van de stream variabele
  - Openen en schrijven naar een file, overschrijft de inhoud van reeds bestaande file in de directory

TEKST TOEVOEGEN AAN TXT FILE

- In sommige gevallen wens je data toe te voegen aan een txt-file
- Code

```
outputStream =
    new PrintWriter(
    new FileOutputstream(fileName, true));
```

- Methode `println` zal data toevoegen aan het einde van de file

TEKST LEZEN VAN EEN TXT-FILE

- Statement voor het openen van het bestand

  Scanner stream_naam =

    new Scanner(new File([bestandsnaam])

- Boolean statement voor het lezen van tekst en beëindigen van de “lees” loop

```
while(inputStream.hasNextLine()){
  String lijn = inputStream.nextLine();
  System.out.println(lijn);
}
```

TEKST LEZEN VAN EEN TXT-FILE

- Bijkomende methodes van de klasse **`Scanner`**
  - **`Scanner_object.hasNext()`**
  - **`Scanner_object.hasNextDouble()`**
  - **`Scanner_object.hasNextInt()`**
  - **`Scanner_object.hasNextLine()`**

OUTLINE

- Streams + File I/O
- I/O Txt bestanden
- Werken met File Klasse
- Een csv-bestand lezen
- I/O Binaire bestanden
- GUI: Bestandsmanagement Demo

DE `FILE` KLASSE

- Klasse maakt het mogelijk om bestandsnamen voor te stellen op een algemene manier
  - Een `File` object is een systeemonafhankelijke abstractie van de bestandsnaam
- Het object

  ```
  new File ("treasure.txt")
  ```

  is geen eenvoudige string
  - Het is een object dat weet dat het moet overeenkomen met de naam van een bestand

GEBRUIK VAN PADNAMEN

- Tot nu werden alle bestanden opgeslagen in de map van waaruit programma ook wordt uitgevoerd
- Het is natuurlijk ook mogelijk om gebruik te maken van bestandspaden
  - Volledige padnaam
  - Relatieve padnaam
- Opgelet met verschillen tussen padnaam stijlen tussen verschillende besturingssystemen.

METHODES VAN DE FILE KLASSE

- `File` klasse heeft methodes die men kan gebruiken voor het ophalen informatie over het pad en het bestand
  - public boolean canRead()
  - public boolean canWrite()
  - public boolean delete()
  - public boolean exists()
  - public String getName()
  - public String getPath()
  - public long length()

METHODE VOOR OPENEN STREAM

- Methode heeft een `String` als parameter
  - De bestandsnaam
- Methode zal het stream object retourneren
- Methode zal exception gooien indien
  - Het bestand niet gevonden is
  - Bij andere I/O problemen
- Moet worden opgeroepen binnen een `try` blok en heeft aangepast `catch` blok

METHODE VOOR OPENEN STREAM

- code

```
public static PrintWriter openOutputTextFile(String fileName)
                    throws FileNotFoundException, IOException
{
    PrintWriter toFile = new PrintWriter(fileName);
    return toFile;
}
```

- Oproep

```
PrintWriter outputStream = null;
try
{
    outputStream = openOutputTextFile("data.txt");
}
< appropriate catch block(s) >
```

UNIVERSITEIT GENT

OUTLINE

- Streams + File I/O
- I/O Txt bestanden
- Werken met File Klasse
- Een csv-bestand lezen
- I/O Binaire bestanden
- GUI: Bestandsmanagement Demo

CASE STUDIE: EEN CSV BESTAND

- Een comma-separated values of CSV bestand is een eenvoudig tekst formaat dat wordt gebruikt voor het opslaan van records
- Voorbeeld: transacties van kassa van een bepaalde dag

```
SKU,Quantity,Price,Description
4039,50,0.99,SODA
9100,5,9.50,T-SHIRT
1949,30,110.00,JAVA PROGRAMMING TEXTBOOK
5199,25,1.50,COOKIE
```

CASE STUDIE: EEN CSV BESTAND

- Uitleg code: String methode split

  lijn = inputSream.nextLine();

  String Transactie = lijn.split(";")

OUTLINE

- Streams + File I/O
- I/O Txt bestanden
- Werken met File Klasse
- Een csv-bestand lezen
- I/O Binaire bestanden
- GUI: Bestandsmanagement Demo

I/O BINAIRE BESTANDEN

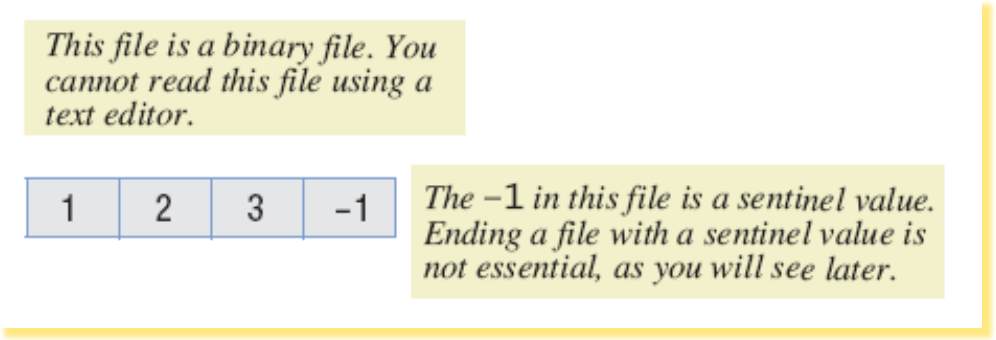
- Aanmaken binair bestand
- Schrijven van primitieve waarden naar binaire bestanden
- Schrijven van Strings naar binaire bestanden
- Lezen van binair bestand
- De klasse **`EOFException`**
- Case: Verwerken van bestand met binaire data
- Schrijven/lezen van objecten
- Schrijven/lezen van array objecten

AANMAKEN BINAIR BESTAND

- Stream klasse **`ObjectOutputStream`** maakt bestand dat men kan gebruiken voor opslaan
  - Primitieve waarden
  - Strings
  - Objecten

SCHRIJVEN VAN PRIMITIEVE WAARDEN

- Method println niet beschikbaar
  - gebruik writeInt method
- Binaire bestanden slaan nummers op in hun binaire vorm
  - Een sequentie van bytes
  - Onmiddellijk na elkaar



OPENEN EN SLUITEN BINAIR BESTAND VOOR OUTPUT

- `public ObjectOutputStream(`
  `new FileOutputStream([Bestandsnaam]))`
- `public ObjectOutputStream(`
  `new FileOutputStream(new File([Bestandsnaam])))`
- `public void close()throws IOException`

SCHRIJVEN NAAR BINAIR BESTAND

- `public void writeInt(int n) throws IOException`
- `public void writeLong(long n)throws IOException`
- `public void writeDouble(double x)throws IOException`
- `public void writeFloat(float x)throws IOException`
- `public void writeChar(int c)throws IOException`
- `public void writeBoolean(boolean b)throws IOException`
- `public void writeUTF(String aString)throws IOException`
- `public void writeObject(Object anObject)`
  `throws IOException, NotSerializableException, InvalidClassException`

UNIVERSITEIT GENT

SCHRIJVEN STRINGS NAAR BINAIR BESTAND

- Gebruik methode **`writeUTF`**
- Voorbeeld

```
outputStream.writeUTF("Hi Mom");
```

- UTF staat voor *Unicode Text Format*
- Gebruikt variabel aantal bytes voor het opslaan van de string
  - Hangt af van de lengte van de string
  - In contrast met **`writeInt`** die altijd zelfde aantal bytes wegschrijft

LEZEN VAN BINAIR BESTAND

- Bestand moet geopend zijn als **`ObjectInputStream`**
- Lezen met methodes die overeenkomen met schrijf methodes
  - Integer weggeschreven met **`writeInt`**, zal voor lezen gebruik maken van **`readInt`**
- Lees altijd zelfde type dan type dat weggeschreven is

OPENEN EN SLUITEN BINAIR BESTAND VOOR INPUT

- `public ObjectOutputStream(`
  `new FileInputStream([Bestandsnaam]))`
  `public ObjectInputStream(`
  `new FileInputStream(new  File([Bestandsnaam])))`
- `public void close()`
  `throws IOException`

LEZEN VAN BINAIR BESTAND

- `public int readInt() throws EOFException, IOException`
- `public int readLong() throws EOFException, IOException`
- `public int readDouble() throws EOFException, IOException`
- `public int readFloat() throws EOFException, IOException`
- `public int readChar() throws EOFException, IOException`
- `public int readBoolean() throws EOFException, IOException`
- `public int readUTF() throws IOException, UTFDataFormatException`

UNIVERSITEIT GENT

LEZEN VAN BINAIR BESTAND

- `public Object readObject() throws ClassNotFoundException, InvalidClassException, OptionalDataException, IOException`

DE KLASSE `EOFEXCEPTION`

- De verschillende methodes die lezen van een binair bestand, gooien wanneer men niets meer kan lezen een `EOFException`
  - Kan me gebruiken voor het bepalen of men aan het einde van een bestand zit
  - Stopt dus de lees-lus

DOUBLER CASE

- Verwerken van een bestand met binaire data
  - Vraag aan gebruiker twee bestandsnamen
  - Lees de getallen van het input bestand
  - Verdubbel deze getallen
  - Schrijf de getallen naar het output bestand

SCHRIJVEN EN LEZEN VAN OBJECTEN

- Soms nodig om niet-String objecten weg te schrijven of te lezen
  - Men kan instantie variabelen één voor één weg schrijven
  - En vervolgens object opnieuw reconstrueren
- Een betere aanpak is beschikbaar in Java
  - Object serialisatie – een object representeren als een sequentie van bytes die men kan schrijven en lezen
  - Mogelijk voor elke klasse door het implementeren Serializable interface

SCHRIJVEN EN LEZEN VAN OBJECTEN

- Interface Serializable is een lege interface
- Men moet geen methodes implementeren
- Zorgt ervoor dat de klasse serialiseerbaar is
  - objecten van deze klassen wegschrijven naar een binair bestand met de methode writeObject
  - objecten lezen met de methode readObject()
    - Object moet wel nog gecast worden naar juiste type

SCHRIJVEN EN LEZEN VAN OBJECTEN

- Wanneer is klasse serialiseerbaar:
  - Implementatie interface Serializable
  - All instantie variabelen van klasse zijn ook objecten van een serialiseerbare klasse
  - De directe superklassen van de klasse zijn serialiseerbaar of definieren een default constructor

SCHRIJVEN EN LEZEN VAN OBJECTEN

- Gevolgen van het serialiseerbaar maken van een klasse
  - Heeft invloed op hoe java de I/O methodes uitvoert van objecten
  - Java geeft een serienummer aan elk object dat men wegschrijft naar de `ObjectOutputStream`
  - Indien hetzelfde objecte meermaals wordt weggeschreven, zal men enkel serienummer nog wegschrijven na de eerste keer

SCHRIJVEN EN LEZEN VAN ARRAYS

- Aangezien een array een object is, kan men gebruik maken van **`writeObject`** voor het wegschrijven van Array
- Annaloog lezen van volledige array via **`readObject`**

”

OUTLINE

- Streams + File I/O
- I/O Txt bestanden
- Werken met File Klasse
- Een csv-bestand lezen
- I/O Binaire bestanden
- GUI: Bestandsmanagement Demo