Fájl IO

A&K AKADÉMIA

MARKOS ANDRÁS

java.io

A java.io csomag tartalmazza a ki- és beviteli műveletekhez szükséges osztályokat.

IO = Input-Output = be-kivitel

Mindig van egy forrás, ahonnan érkeznek adatok, és van egy cél, ahová szeretnénk eljuttatni.

A programunk az eszköz a forrástól a célig való eljuttatáshoz.

Többféleképpen csoportosíthatjuk az IO műveleteket.

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io

Az adat helye

- Memória
- Fájl
- Egy másik program
- Hálózat

Az adat típusa szerint

- Objektum
- Hang, …
- Karakteres adat
- …

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io

Az adatfolyam (stream) az IO műveletek egyik alapfogalma.

Ez egy absztrakt fogalom, a szekvenciális hozzáférésű csatornát jelöli, amely összeköti a forrást a nyelővel.

Adatbevitel általános menete:

- Megnyitjuk a folyamot (open)
- Olvasunk a folyamból (read)
- Bezárjuk a folyamot (close)

Adatkivitel általános menete:

- Megnyitjuk a folyamot (open)
- Írunk a folyamba (write)
- Bezárjuk a folyamot (close)

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io

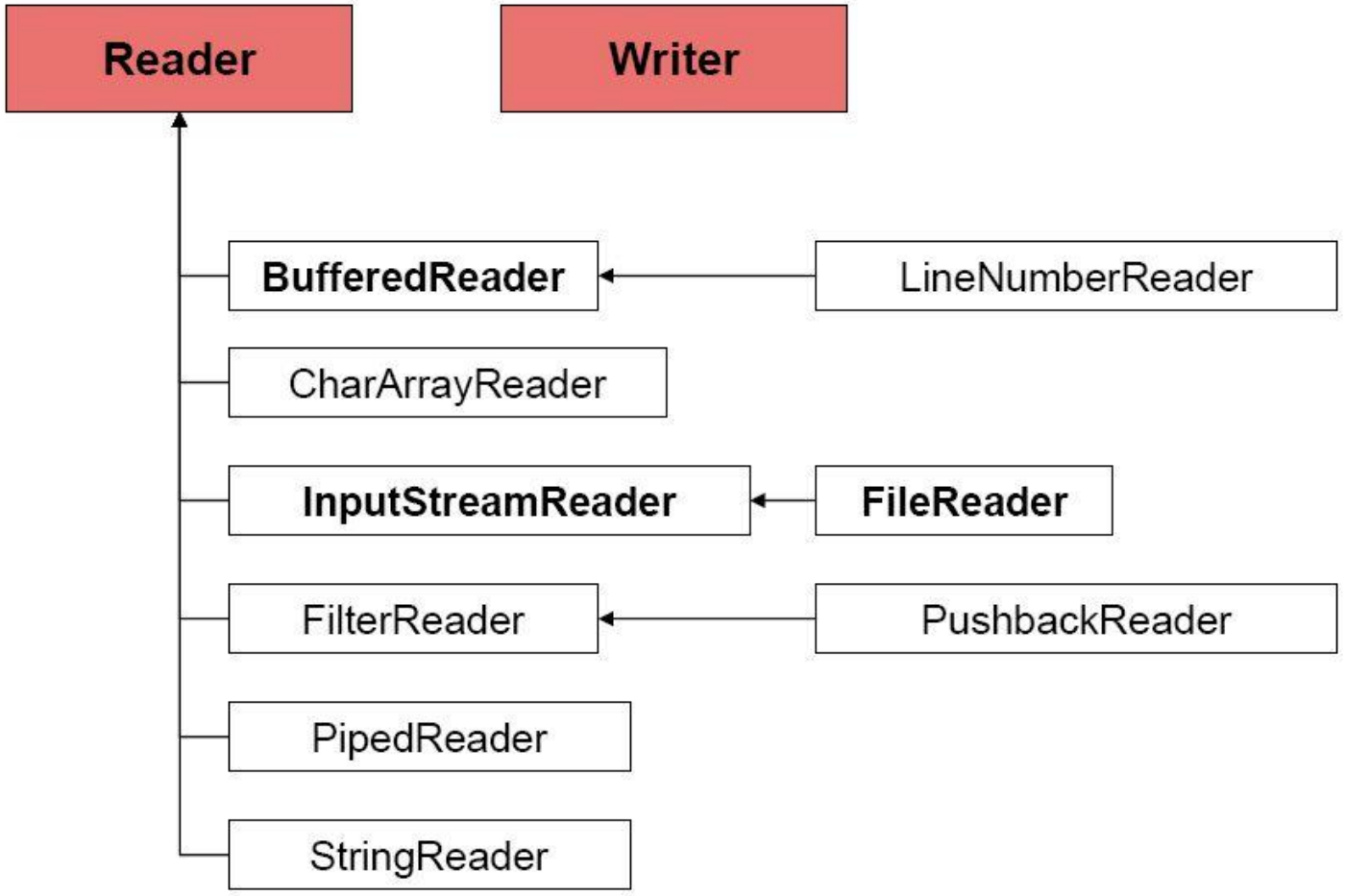
Kétfajta IO típus létezik

- Karakter folyam
  - 16 bites UNICODE karaktereket kezel
  - Írás, olvasás: `Writer`, `Reader`

- Bináris folyam
  - 8 bites, vagyis 1 bájtos adatokat kezel
  - Írás, olvasás: `InputStream`, `OutputStream`

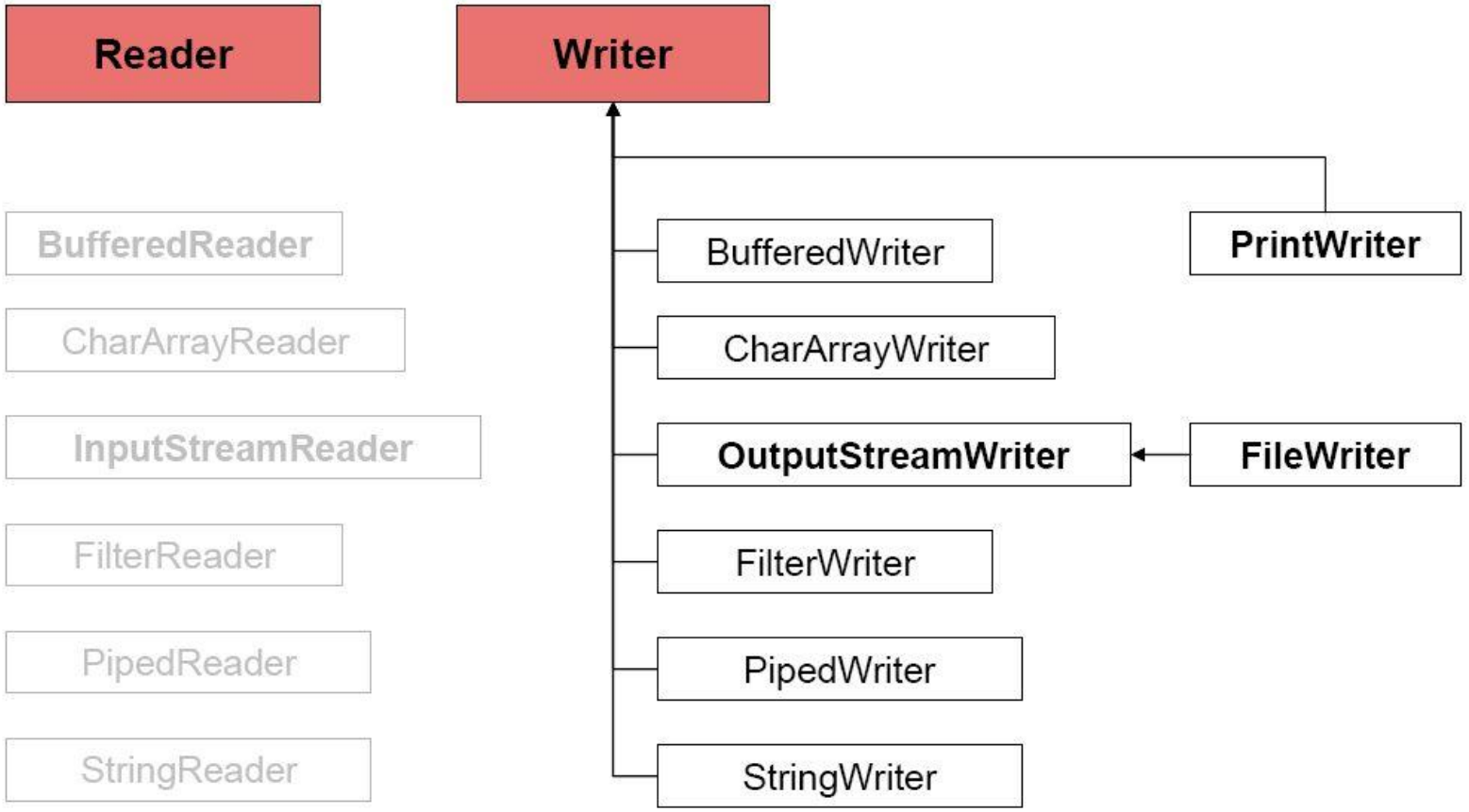
© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io



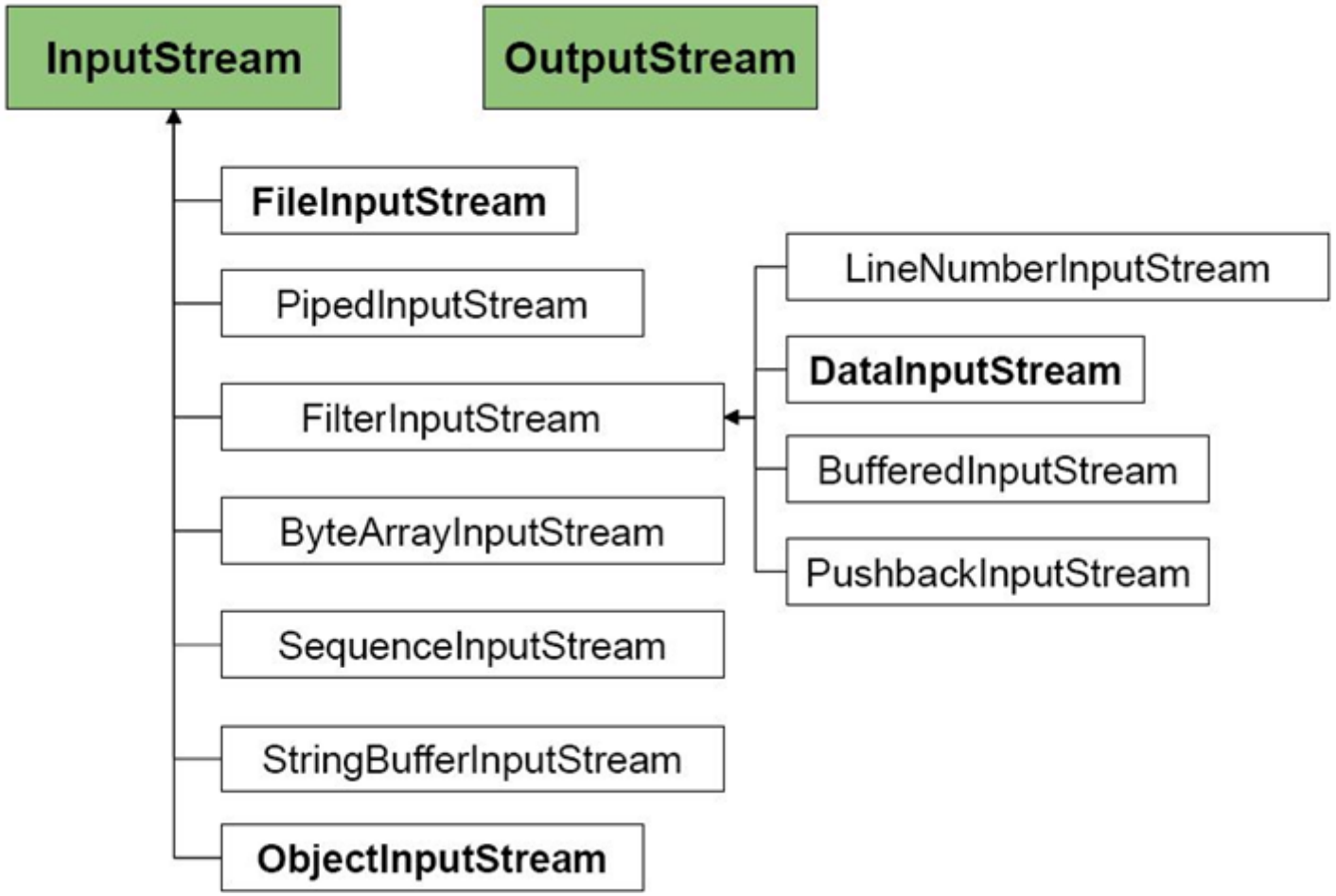
© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io



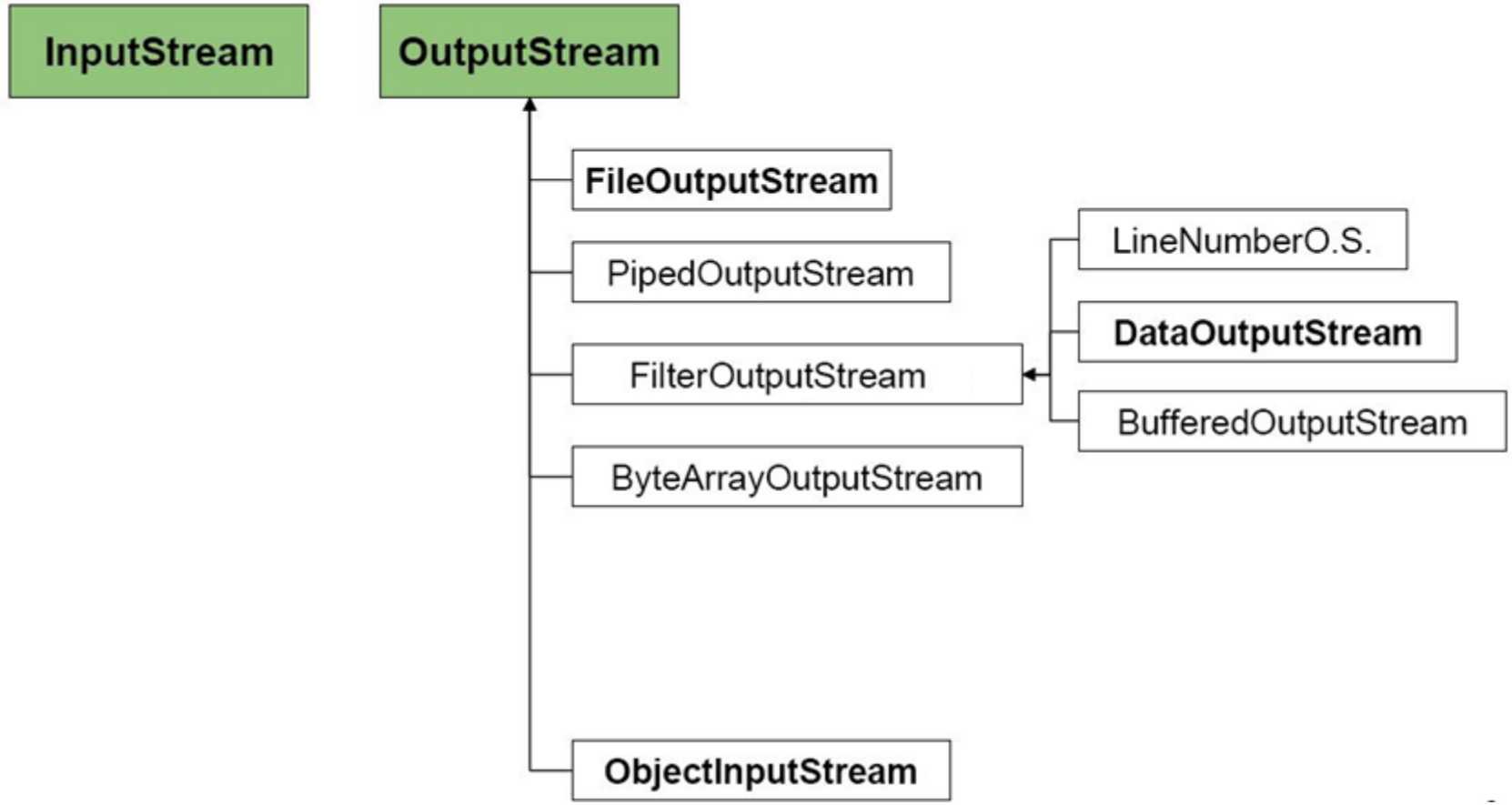
© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io



© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io



© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

java.io

Mind a karakteres, mind a bináris IO-nak az API-ja `read()` és `write()` metódusokból áll.

A karakteres folyamnál ezek a metódusok `char` primitív típussal operálnak.

A bináris folyamnál `byte` primitív típussal.

Az előbb bemutatott számos osztály mind IO kezelést végez, mindig a megfelelőt kell kiválasztanunk.

Most végignézzük a fontosabbakat.

Egyes változatok speciális visszatérési értékkel jelzik a folyam végét, mások kivételt dobnak.

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Karakteres fájlkezelés

Ehhez a `FileReader` és `FileWriter` osztályokat használhatjuk.

Másoljunk le egy fájlt karakterről karakterre!

Példa:

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class TextFileCopyDemo {
    public static void main(String[] args) throws IOException {
        FileReader in = new FileReader("copy-from.txt");
        FileWriter out = new FileWriter("copy-to.txt");

        in.close();
        out.close();
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Karakteres fájlkezelés

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class TextFileCopyDemo {
    public static void main(String[] args) throws IOException {
        FileReader in = new FileReader("copy-from.txt");
        FileWriter out = new FileWriter("copy-to.txt");

        while (true) {
            int character = in.read();
            if (character == -1) {
                break;
            }
            out.write(character);
        }

        in.close();
        out.close();
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Karakteres fájlkezelés

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class TextFileCopyDemo {
    public static void main(String[] args) {
        try (FileReader in = new FileReader("copy-from.txt");
             FileWriter out = new FileWriter("copy-to.txt")) {
            while (true) {
                int character = in.read();
                if (character == -1) {
                    break;
                }
                out.write(character);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Bináris fájlkezelés

Ehhez az `FileInputStream` és `FileOutputStream` osztályokat használhatjuk.

Másoljunk le egy fájlt bájtról bájtra!

Példa:

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class BinaryFileCopyDemo {
    public static void main(String[] args) throws IOException {
        FileInputStream in = new FileInputStream("13. File IO.pdf");
        FileOutputStream out = new FileOutputStream("13. File IO copy.pdf");

        in.close();
        out.close();
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Bináris fájlkezelés

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class BinaryFileCopyDemo {
    public static void main(String[] args) throws IOException {
        FileInputStream in = new FileInputStream("13. File IO.pdf");
        FileOutputStream out = new FileOutputStream("13. File IO copy.pdf");

        while (true) {
            int oneByte = in.read();
            if (oneByte == -1) {
                break;
            }
            out.write(oneByte);
        }

        in.close();
        out.close();
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Bináris fájlkezelés

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class BinaryFileCopyDemo {
    public static void main(String[] args) {
        try (FileInputStream in = new FileInputStream("13. File IO.pdf");
             FileOutputStream out = new FileOutputStream("13. File IO copy.pdf")) {
            while (true) {
                int oneByte = in.read();
                if (oneByte == -1) {
                    break;
                }
                out.write(oneByte);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Fájlkezelés

A fájlkezeléshez a konstruktorban több mindent megadhatunk, nem csak a fájl nevét `String`-ként, hanem

- `File` objektumot
- `FileDescriptor`-t

Ha nem található a fájl, akkor `FileNotFoundException` dobódik.

`SecurityException` is dobódhat ha nincs elég jogosultságunk egy fájlhoz való hozzáférésnél.

A `File` osztály egy absztrakt reprezentációja egy fájlnak vagy könyvtárnak.

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Memória IO

Az IO műveletek irányulhatnak nem csak a háttértárra, hanem a memóriára is.

Ezt kezelik ezek az osztályok:

- `StringReader`
- `StringWriter`
- `CharArrayReader`
- `CharArrayWriter`
- `ByteArrayReader`
- `ByteArrayWriter`

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Memória IO

Példa:

```
import java.io.CharArrayReader;
import java.io.IOException;

public class CharArrayReaderDemo {
    public static void main(String[] args) throws IOException {
        char[] charArray = { 'h', 'e', 'l', 'l', 'o' };
        CharArrayReader in = new CharArrayReader(charArray);
        charArray[4] = '!';
        System.out.println((char) in.read()); // h
        System.out.println((char) in.read()); // e
        System.out.println((char) in.read()); // l
        System.out.println((char) in.read()); // l
        System.out.println((char) in.read()); // !
    }
}
```

- A `CharArrayReader` nem másolja le a `char` tömböt.
- A `ByteArrayInputStream` és a `StringReader` immutábilisak, vagyis megnyitásuk után nem módosíthatók.

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Memória IO

`StringReader` példa:

```
import java.io.IOException;
import java.io.StringReader;

public class StringReaderDemo {
    public static void main(String[] args) {
        String data = "First line\nSecond line\nThird line";
        try (StringReader stringReader = new StringReader(data)) {
            for (int character = stringReader.read(); character != -1; character = stringReader.read()) {
                System.out.print((char) character);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Memória IO

Memóriába írás.

A memóriába író osztályok egy tömböt használnak a háttérben, amely növekszik.

Megadható a puffer kezdeti mérete és a növekedés mértéke.

Alapértelmezés szerint, ha betelik, akkor megkétszerezi saját méretét.

A `close()` metódus ezen az osztályon (pl. `CharArrayWriter`) nem működik, és bezárás után is kivétel dobása nélkül meg lehet hívni a metódusait.

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Konverzió karakter- és bájtfolyam között

Ha van egy `byte` csatornánk, akkor ehhez egy karakteres felületet biztosít az `InputStreamReader` és az `OutputStreamWriter`.

A konverzió alapértelmezés szerint Unicode-ra történik.

Billentyűzetről beolvasás adott karakterkódolással:

```
InputStreamReader in = new InputStreamReader(System.in, "ISO-8859-2");
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Szűrők

Az **eddig**i osztályoknál mindig az olvasás **hely**ét kellett megadnunk.

Most az **olvasás egység**ét nézzük meg, ehhez vannak külön osztályok, ezeket szűrőknek nevezzük.

Ezek a szűrők becsomagolják (wrap) az eddig tárgyalt olvasókat / írókat, amelyek már tisztában vannak a forrás / cél helyével.

```
FileInputStream fileInputStream = new FileInputStream("13. File IO.pdf");
BufferedInputStream in = new BufferedInputStream(fileInputStream);
```

általánosságban:

```
SZŰRŐ in = new SZŰRŐ(fileInputStream);
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Szűrők

A legfontosabb szűrők:

- `DataInputStream / DataOutputStream`
- `BufferedReader / PrintWriter`
- `ObjectInputStream / ObjectOutputStream`

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

DataInputStream / DataOutputStream

A Java primitív adattípusokat tudjuk vele hatékonyan beolvasni, illetve kiírni (`DataOutputStream`).

Primitív adattípusok fájlba írására példa:

```
import java.io.DataOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class DataOutputStreamDemo {
    public static void main(String[] args) throws IOException {
        try (DataOutputStream out = new DataOutputStream(new FileOutputStream("myBinaryFile.bin"))) {
            for (int i = 0; i < 100; i++) {
                out.writeDouble(i + 3.1415);
                out.writeChar('a');
                out.writeInt(i - 50);
                out.writeUTF("hello");
            }
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

DataInputStream / DataOutputStream

Primitív adattípusok fájlból olvasására példa:

```
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;

public class DataInputStreamDemo {
    public static void main(String[] args) throws IOException {
        try (DataInputStream in = new DataInputStream(new FileInputStream("myBinaryFile.bin"))) {
            for (int i = 0; i < 100; i++) {
                double doubleValue = in.readDouble();
                char character = in.readChar();
                int wholeNumber = in.readInt();
                String text = in.readUTF();
                System.out.println(doubleValue);
                System.out.println(character);
                System.out.println(wholeNumber);
                System.out.println(text);
            }
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Pufferelt olvasás / írás

Sokszor jó, hogyha nem karakterenként vagy bájtonként kell beolvasnunk egy folyamból, hanem egyszerre nagyobb egységekben érjük el az adatokat.

Jó, mert kényelmes.

Jó, mert gyorsabb tud lenni.

Nem feltétlen kell tudnunk előre, hogy mennyit fogunk beolvasni, kijelölhetünk valami végjelet, ami az olvasás végét jelöli.

Ide tartozó osztályok:

- `BufferedReader, PrintWriter`
- `BufferedInputStream, BufferedOutputStream / PrintStream`

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

BufferedReader

Tartalma egy hívással elérhető.

Ha túlcsordulna, akkor megduplázódik.

Megadható a kezdeti puffer méret.

Példa:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class BufferedReaderDemo {
    public static void main(String[] args) throws IOException {
        try (BufferedReader in = new BufferedReader(new FileReader("input.txt"))) {
            String line = in.readLine();
            System.out.println(line);
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

PrintWriter

A kiírandó objektumnak meghívja a `toString()` metódusát, hogy előállítsa a szöveges változatát.

Automatikus puffer kiírás (`autoflush`) megadható.

Példa:

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class PrintWriterDemo {
    public static void main(String[] args) throws IOException {
        try (PrintWriter out = new PrintWriter(new FileWriter("output.txt"))) {
            out.println("test text");
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Karakterkódolás megadása szükség esetén

Karakterkódolás csak az `InputStreamReader`-nek és az `OutputStreamWriter`-nek adható meg, a leszármazottjainak nem.

Példa fájlból való olvasásra karakterkódolás megadásával:

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;

public class BufferedReaderWithSpecifiedCharacterEncodingDemo {
    public static void main(String[] args) throws IOException {
        try (BufferedReader in = new BufferedReader(
                                 new InputStreamReader(
                                 new FileInputStream("input.txt"), "ISO-8859-2"))) {
            String line = in.readLine();
            System.out.println(line);
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

IO Objektumokkal

Erre használható osztályok:

- `ObjectInputStream`
- `ObjectOutputStream`

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

IO Objektumokkal

Példa objektumok fájlba történő kiírására:

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class ObjectOutputStreamDemo {
    public static void main(String[] args) throws IOException {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("objects.bin"))) {
            for (int i = 0; i < 10; i++) {
                out.writeObject(new String("some text " + i));
                out.writeObject(new Auto(i));
            }
        }
    }
}
```

```
import java.io.Serializable;

public class Auto implements Serializable {

    private static final long serialVersionUID = 7060879287674253903L;

    private int speed;

    public Auto(int speed) {
        this.speed = speed;
    }

    public int getSpeed() {
        return speed;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

    @Override
    public String toString() {
        return "Auto [speed=" + speed + "]";
    }

}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

IO Objektumokkal

Objektum beolvasása fájlból példa:

```java
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;

public class ObjectInputStreamDemo {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream("objects.bin"))) {
            for (int i = 0; i < 10; i++) {
                String string = (String) in.readObject();
                Auto auto = (Auto) in.readObject();
                System.out.println(string);
                System.out.println(auto);
            }
        }
    }
}
```

```java
import java.io.Serializable;

public class Auto implements Serializable {

    private static final long serialVersionUID = 7060879287674253903L;

    private int speed;

    public Auto(int speed) {
        this.speed = speed;
    }

    public int getSpeed() {
        return speed;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

    @Override
    public String toString() {
        return "Auto [speed=" + speed + "]";
    }

}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Szerializáció

A szerializáció az a folyamat, amikor egy memóriában létező objektumot átalakítunk bájtsorozattá.

A transzformáció a két forma között egyértelmű.

Csak a `Serializable` interfészt megvalósító osztály szerializálható.

A `Serializable` interfész így néz ki:

```
package java.io;

public interface Serializable {
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

Automatikus szerializáció

```
ObjectOutputStream.defaultWriteObject() throws IOException;

ObjectInputStream.defaultReadObject() throws IOException, ClassNotFoundException;
```

Az automatikus szerializációkor kiíródik:

- Az objektum osztálya, az osztály szignatúrája.
- Az objektum nem-statikus és nem-tranziens mezőinek értéke.
- Az objektum referenciáinak tranzitív lezártja.

Tranziens mező: **`transient`** módosítószó, nem íródik ki a szerializációkor.

A szerializáció veszélyes, új projektekben kerülendő, helyette hasonló funkcionalitást megvalósító külső library-k használata javasolt.

Tervezik a helyettesítését a Java Core library-ban is, ekkor a jelenlegi szerializációs folyamat deprecated lesz.

© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

StreamTokenizer

Olyan, mint a `StringTokenizer`, de folyamokra.

Megadható, hogy milyen elválasztó karakter mentén daraboljon.

Karakteres csatorna felett működik.

Példa:

```
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.io.StreamTokenizer;

public class StreamTokenizerDemo {
    public static void main(String[] args) throws IOException {
        try (Reader fileReader = new FileReader("input.txt")) {
            StreamTokenizer tokenizer = new StreamTokenizer(fileReader);
            while (tokenizer.nextToken() != StreamTokenizer.TT_EOF) {
                System.out.println(tokenizer.sval);
            }
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA

RandomAccessFile

Tetszőleges hozzáférést biztosít egy fájlhoz, amiből olvashatunk és amibe írhatunk.

Példa:

```
import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;

public class RandomAccessFileDemo {
    public static void main(String[] args) throws IOException {
        try (RandomAccessFile randomAccessFile = new RandomAccessFile(new File("input.txt"), "rw")) {
            String line = randomAccessFile.readLine();
            System.out.println(line);
            randomAccessFile.seek(new File("input.txt").length());
            randomAccessFile.write("some more text".getBytes());
        }
    }
}
```


© 2020 A&K AKADÉMIA – MINDEN JOG FENNTARTVA