Programozási nyelvek Java

9. gyakorlat

Fájlkezelés

A fájlkezelés Java-ban különböző osztályok összekapcsolásával történik. Minden egyes osztály valamilyen minimális szolgáltatást tesz hozzá a többihez. Az osztályok a **java.io** csomagban találhatók.

BufferedReader

Megvalósítja az alapvető olvasási funkciókat.

- Egy Reader típusú (absztrakt) objektumot vár konstruktorparaméterként
- **boolean ready()**: megadja, hogy a stream-ből tudunk-e olvasni
- **int read()**: beolvas egy karaktert
- **String readLine()**: a sor végéig olvas
- **void close()**: lezárja a stream-et.
- Műveletei **IOException** kivételt válthat ki

FileReader

- Megadható egy BufferedReader objektum konstruktorában várt paraméternek
- Konstruktorban megadható egy fájlnév vagy egy File típusú objektum is
- Kiválthat **FileNotFoundException** kivételt

BufferedWriter

Megvalósítja az alapvető írási funkciókat.

- Megadható konstruktorparaméterként egy Writer típusú (absztrakt) objektumot, egy fájlnév és egy File típusú objektum is.
- **void write(char[] c)**: kiírja a stream-be a megadott paramétert
- **void newLine()**: új sort emel
- **void flush()**: kitörli a stream tartalmát
- **void close()**: bezárja a stream-et, és felszabadítja az erőforrásokat

FileWriter

- Megadható egy PrintWriter objektum konstruktorában várt paraméternek
- Konstruktorban megadható egy fájlnév vagy egy File típusú objektum is
- Kiválthat **IOException** kivételt

PrintWriter

Megvalósítja az alapvető írási funkciókat.

- Megadható konstruktorparaméterként egy Writer típusú (absztrakt) objektumot, egy fájlnév és egy File típusú objektum is.
- Mindegyik esetben megadható még egy boolean típusú paraméter is, amely azt mondja meg, hogy megnyitáskor törölje-e a fájl tartalmát
- **void print(object o)**: kiírja a stream-be a megadott paramétert (több túlterhelése létezik)
- **void println(object o)**: kiírja a stream-be a megadott paramétert, majd új sort emel (több túlterhelése létezik)
- **void flush()**: kitörli a stream tartalmát
- **void close()**: bezárja a stream-et, és felszabadítja az erőforrásokat

Példák

Fájlolvasás

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

import java.io.IOException;
import java.io.FileNotFoundException;

public class Main {

    public static void main(String [] args) {

        BufferedReader br = null;
        try {
            //Read
            br = new BufferedReader(new FileReader("input.txt"));
            String str = br.readLine();
            while(null != str) {
                System.out.println(str);
                str = br.readLine();
            }
        }
        catch(FileNotFoundException ex) {
            System.out.println("The input.txt does not exist!");
        }
        catch(IOException ex) {
            ex.printStackTrace();
        }
        finally {
            //closing section
            try {
                if(br != null) br.close();
            }
            catch(IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

Fájlírás1

```
import java.io.BufferedWriter;
import java.io.FileWriter;

import java.io.IOException;
import java.io.FileNotFoundException;

public class Main {

    public static void main(String [] args) {

        BufferedWriter bw = null;
        try {
            bw = new BufferedWriter(new FileWriter("output2.txt", false));

            bw.write("I never saw a purple cow,");
            bw.newLine();
            bw.write("I never hope to see one;");
            bw.newLine();
            bw.write("But I can tell you, anyhow,");
            bw.newLine();
            bw.write("I'd rather see than be one!");
            bw.newLine();
        }
        catch(IOException ex) {
            ex.printStackTrace();
        }
        finally {
            //closing section
            try {
                if(bw != null) bw.close();
            }
            catch(IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

Fájlírás2

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

import java.io.IOException;
import java.io.FileNotFoundException;

public class Main {

    public static void main(String [] args) {

        PrintWriter pw = null;
        try {
            pw = new PrintWriter(new FileWriter("output1.txt", false));

            pw.println("I never saw a purple cow,");
            pw.println("I never hope to see one;");
            pw.println("But I can tell you, anyhow,");
            pw.println("I'd rather see than be one!");

        }
        catch(IOException ex) {
            ex.printStackTrace();
        }
        finally {
            pw.close();
        }
    }
}
```

Feladatok

1. Készíts egy olyan programot, ami az első paraméterben megadott nevű fájlban megszámolja hány sornyi szöveg található, majd kiírja a képernyőre az eredményt.

2. Készíts egy olyan programot, ami az első paraméterben (argumentumban) megadott nevű fájlban található (soronként 1 darab) egész számokat beolvassa, majd a második argumentumban megadott nevű fájlba visszaírja csak az ellentettjüket, végül pedig a fájl végére kiírja az eredeti számok összegét és annak ellentettjét is.

3. Készíts egy programot, ami kiírja egy parancssori argumentumként megadott nevű fájl páratlan sorait a képernyőre, páros sorait pedig egy „even.txt” nevű fájlba teszi bele. Ezután az „even.txt” fájl tartalmával írjátok felül az eredeti fájlt.

4. Készítsünk egy egyszerű *l33t5p34k* (leetspeak) generátort! A program 1 parancssori argumentumot kapjon: egy input fájl elérési utat. A program olvassa be az input fájlt, és minden szón végezze el a következő módosításokat, majd írja ki a képernyőre a módosított szöveget:

    - Ha a szó vége:
        - "s" helyett "z"
        - "ck" helyett "x"
    - "a" helyett "@"
    - "e" helyett "3"
    - "i" helyett `"1"
    - "o" helyett "0"
    - "u" helyett "v"
    - "f" helyett "ph"
    - "s" helyett "$"
    - "g" helyett `"9"
    - "y" helyett "j"
    - "t" helyett "+"
    - "!" helyett "1"
    - Ezen kívül minden 2. karakter esetén a kisbetűből csináljon nagyot, a nagybetűből kicsit!