Laboration 5 (Datakom F3), Strömmar

Avsikten med laborationen är att du ska träna på att använda strömmar samt att lagra och hämta data på en hårddisk.

På kurssidan hittar du följande filer: *Person.java, WriteToFile.java, heltal.dat, medlemmar.txt*

Placera java-filerna i paketet ***laboration3*** och de övriga filerna i katalogen ***files*** vilken ska vara i projektet.

Uppgift 1 – Hur stor är summan?

Lägg filen **heltal.dat** i katalogen *files* och skriv sedan ett program som beräknar summan av heltalen i filen. Filen är formaterad så här:

int 1: Antal int som följer (nedan *n* st)
int 2, int 3, int 4, …, **int n**

Exempel: 7 19 2 -13 5 3 5 8 7:an anger att det följer 7 st int (19, 2, ..., 8)

Lämpliga klasser att använda i lösningen är **DataInputStream**, **FileInputStream** och eventuellt **BufferedInputStream**.

Uppgift 2 – Hur många är medlemmarna? Hur många är kvinnor?

Lägg filen **medlemmar.txt** i katalogen *files*. Filen består av ett antal rader med text. Varje rad representerar en medlem och består av ett namn och avslutas med K (kvinna) eller M (man):

”Anders M”

”Siv K”

Skriv ett program som skriver ut samtliga medlemmars namn, antalet medlemmar och antalet kvinnliga medlemmar.

Aktuella klasser att använda är **BufferedReader, InputStreamReader** och **FileInputStream**. Teckenkodningen är "ISO-8859-1".

Uppgift 3 – Skriva till hårddisk

Komplettera nedanstående metoder i klassen **WriteToFile** som du hittar på kurssidan.

- **public void writePrimes(String filenamn) throws IOException**
  writePrimes ska skriva primtalen i arrayen *primes* till filen ”files/primes.dat”. Först ska antalet primtal skrivas som en int och sedan följer primtalen:
  int, long, long, long, …

  När du testkör programmet WriteToFile och väljer ”Skriv primtal” så anropas metoden writePrimes. Sedan kan du kontrollera innehållet i filen du skapat genom att välja ”Läs primtal”.

- **public void writeInvited(String filename) throws IOException**
  writeInvited ska skriva strängarna i **invited**-listan till filen ”files/invited.txt”. Teckenkodningen i filen ska vara ”UTF-8”. Strängarna ska skrivas som strängar med **write**-metoden i **BufferedWriter**. Glöm inte att anropa **newLine** efter varje sträng.

  Kontrollera ditt resultat genom att välja ”Läs bjudna”.

- **public void writePersons(String filename) throws IOException**
  writePersons ska skriva **Person**-objekten i ***persons*** till filen ”files/persons.dat”. Först ska antalet objekt på filen skrivas som en int och sedan ska objekten skrivas:
  int, Person, Person, Person, …
  Lämpliga klassera att använda i lösningen är **ObjectOutputStream** och **FileOutputStream**.

  Kontrollera ditt resultat genom att välja ”Läs personer”.

Uppgift 4 – Hur gamla är medlemmarna?

Läs objekten i filen persons.dat (som du skrev i uppgift 3) och beräkna Person-objektens genomsnittliga ålder. Beräkna dessutom kvinnornas genomsnittliga ålder.

Lämpliga klasser att använda i lösningen är **ObjectInputStream** och **FileInputStream**. Tänk på att undantaget **ClassNotFoundException** måste fångas när man läser ett objekt via en objektström.

Uppgift 5 – Skriva till hårddisk

Skriv metoderna **writePrimes2**, **readPrimes2**, **writeInvited2**, **readInvited2**, **writePersons2** och **readPersons2** i WriteToFile.java så att objektsamlingarna *primes*, *invited* respektive *persons* skrivs/läses som objekt. Det innebär t.ex. att hela arrayen *primes* skrivs som ett objekt i *writeprimes2* och hela arrayen läses som ett objekt i *readPrimes2*.

Uppgift 6a – Kopiera fil

Skriv metoden **copyFile(String source**) i klassen Exercise6. Metoden tar namnet till en fil som input, läser innehållet från filen och skriver innehållet det till en ny fil.

Upplägg:

Skapa ett filnamn, **dest**, till den nya filen. Filnamnet kan vara samma som source men med ” copy” på slutet.
Skapa en InputStream till **source**. Den bör använda en BufferedInputStream. *bis* nedan.
Skapa en OutputStream till **dest**. Den bör använda en BufferedOutputStream. *bos* nedan.
Läs från **source** med *bis.read()*-metoden och skriv till **dest** med *bos.write(int)*-metoden. Så länge det finns mer data att överföra.

```
    int data = bis.read();
    while( data!=-1 ) {
        bos.write(data);
        date = bis.read();
    }
    bos.flush();
```

Du kan testa din metod med följande main-metod.

```
public static void main(String[] args) {
    Exercise6 e6 = new Exercise6();
    String filepath;
    JFileChooser fileChooser = new JFileChooser();
    if(fileChooser.showOpenDialog(null)==JFileChooser.APPROVE_OPTION) {
        filepath = fileChooser.getSelectedFile().getPath();
        e6.copyFile(filepath);
    }
}
```

Uppgift 6b – Kopiera fil (version 2)

Raderna

```
    int data = bis.read();
    while( data!=-1 ) {
        bos.write(data);
        date = bis.read();
    }
    os.flush();
```

är användbara I sammanhang då man ska läsa från en ström och skriva till en annan. Det kan handla om att kopiera en fil som ovan men även om man ska göra en zip-fil eller packa upp en zip-fil, kryptera en fil eller dekryptera en fil osv.

Skriv klassmetoden **public static void copy(InputStream is, OutputStream os)** i klassen Exercise6.

Ersätt ovanstående rader i metoden *copyFile* med anrop till copy-metoden:

```
    Exercise6.copy(bis,bos);
```

Uppgift 6c – skapa en zip-fil

Här ska du skriva en metod vilken tar en lista av filer och placerar dessa i en zip-fil. JFileChooser medger att man låter användaren markera många filer:

```
Exercise6 e6 = new Exercise6();
JFileChooser fileChooser = new JFileChooser();
fileChooser.setMultiSelectionEnabled(true);
if (fileChooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
    File[] files = fileChooser.getSelectedFiles();
    if(files.length>0) {
        e6.zip(files);
    }
}
```

Sedan får man de markerade filerna, som en array med **File**-objekt, genom anrop till metoden: *fileChooser.getSelectedFiles()*

Klassen **File** innehåller många metoder, bl.a.:

public String getPath() Returnerar hela sökvägen till filen
public String getName() Returnerar filens namn
public String getParent() Returnerar sökvägen utan filnamnet

Klasserna *FileInputStream* och *FileOutputStream* har konstruktorer som tar ett *File*-objekt som argument.

Skriv metoden **public void zip(File[] files)** i klassen *Exercise6*. Metoden ska ta en array med *File*-objekt som input och skapa en zip-fil som innehåller dessa filer. Zip-filen ska vara i samma katalog som de valda filerna och heta **archive.zip**.

Metoden ska fungera ungefär så här:

- Skapa arkivfilens namn genom att använda sökvägen för det första *File*-objektet i listan *files* och lägga till ”/archive.zip”, dvs. *files[0].getParent() + ...*
- Skapa en *ZipOutputStream* (*zos* nedan) kopplad till arkivfilens namn.
- För varje *File*-objekt i argumentet *files*:
    - Skapa en *InputStream* (*bis* nedan) till aktuellt *File*-objekt.
    - Lägg till en ny entry i zip-filen: Anropa *zos.putNextEntry (filens namn)*
    - Anrop *Exercise6.copy(bis,zos)*
    - Anrop *zos.closeEntry()*

Testa din lösning med exempelkoden i början av Uppgift 6c. Dubbelklicka en skapad archive.zip-fil och konstatera att zip-filen är korrekt skriven.

Uppgift 6d – packa upp en zip-fil

I uppgift 6d ska du skriva en metod vilken tar en zip-fil som input (file / filename nedan), skapar en katalog i den katalog där zip-filen lagras, ger den nya katalogen samma namn som zip-filen (utan filtypen) och packar upp filerna i zip-filen i den nya katalogen.

Skriv metoden **public void unzip(File file)** eller **public void unzip(String filename)** i klassen Exercise6.

För att skapa en katalog använder du argumentet *file* eller *filename*:

- Ta bort filens typ från *file.getPath()* / *filename* och lagra resultatet i *directoryName*.
- Skapa ett *File*-objekt, *directory*, med *directoryName* som argument.
- Anrop metoden *mkdir()* med *directory*, dvs *directory.mkdir();*

Nu går du till väga som i 6c men ”tvärtom” (se även F3-WriteReadData2.java):

Medans input-strömmen inte är slut (inga nya Entry-objekt i zip-filen) så skapar vi en ny output-ström till det nya Entry-objektet, dvs directoryName + ”/” + …

Uppgift 1

Svar: 172907

```
public class Exercise1 {
    public void sum(String filename) throws IOException {
        try (DataInputStream dis = new DataInputStream(new
FileInputStream(filename))) {
            int sum=0;
            int n = dis.readInt();
            for(int i=0; i<n; i++) {
                sum += dis.readInt();
            }
            System.out.println(sum);
        }
    }

    public static void main(String[] args) throws IOException {
        Exercise1 ex1 = new Exercise1();
        ex1.sum("files/heltal.dat");
    }
}
```

Uppgift 2

```
public class Exercise2 {
    public void statistics(String filename) throws IOException {
        try (BufferedReader br = new BufferedReader( new InputStreamReader(
                new FileInputStream(filename),"ISO-8859-1"))) {
        int count=0, women=0;
        String name = br.readLine();
        while(name!=null) {
            count++;
            System.out.print(name);
            if(name.endsWith("K"))
                women++;
            name = br.readLine();
            if(name!=null)
                System.out.print(", ");
        }
        System.out.print("\nAntal medlemmar: "+count+" , därav "+
                        women+" kvinnor.\n");
        }
    }

    public static void main(String[] args) throws IOException{
        Exercise2 ex2 = new Exercise2();
        ex2.statistics("files/medlemmar.txt");
    }
}
```

Uppgift 3

```
public void writePrimes(String filename) throws IOException {
    try (DataOutputStream dos = new DataOutputStream(
             new BufferedOutputStream(new FileOutputStream(filename)))) {
        dos.writeInt(primes.length);
        for (int i = 0; i < primes.length; i++) {
            dos.writeLong(primes[i]);
        }
        dos.flush();
    }
}

public void writeInvited(String filename) throws IOException {
    try (BufferedWriter bos = new BufferedWriter(new OutputStreamWriter(
             new FileOutputStream(filename),"UTF-8"))) {
    for (int i = 0; i < invited.size(); i++) {
```

```
        bos.write(invited.get(i));
        bos.newLine();
    }
    bos.flush();
    }
}

public void writePersons(String filename) throws IOException {
    try (ObjectOutputStream oos = new ObjectOutputStream(
             new BufferedOutputStream(new FileOutputStream(filename)))) {
        oos.writeInt(persons.size());
        for (Person p : persons) {
            oos.writeObject(p);
        }
        oos.flush();
    }
}
```

Uppgift 4

```
public class Exercise4 {
    public void statistics(String filename) throws IOException {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(filename))) {
            int n, women=0, sum=0, sumWomen=0;
            n = ois.readInt();
            Person[] p = new Person[n];
            for(int i=0; i<n; i++) {
                try {
                    p[i] = (Person) ois.readObject();
                    sum += p[i].getAge();
                    if(p[i].getSex()=='K') {
                        sumWomen += p[i].getAge();
                        women++;
                    }
                }catch(ClassNotFoundException e) {
                    System.out.println(e);
                }
            }
            System.out.println("Genomsnittlig ålder är "+(double)sum/n);
            System.out.println("Kvinnornas genomsnittlig ålder är "+
                    (double)sumWomen/women);
        }
    }

    public static void main(String[] args) throws IOException {
        Exercise4 ex4 = new Exercise4();
        ex4.statistics("files/persons.dat");
    }
}
```

Uppgift 5

```
public void writePrimes2(String filename) throws IOException {
    try (ObjectOutputStream dos = new ObjectOutputStream(
                new FileOutputStream(filename))) {
            dos.writeObject(primes);
            dos.flush();
    }
}

public void readPrimes2(String filename) throws IOException {
    long[] primes=null;
    try (ObjectInputStream ois = new ObjectInputStream(
                new FileInputStream(filename))) {
            primes = (long[])ois.readObject();
    }catch(ClassNotFoundException e) {}
}
```

```
public void writeInvited2(String filename) throws IOException {
    try (ObjectOutputStream oos = new ObjectOutputStream(
                new FileOutputStream(filename))) {
            oos.writeObject(invited);
            oos.flush();
    }
}

public void readInvited2(String filename) throws IOException {
    try (ObjectInputStream ois = new ObjectInputStream(
                new FileInputStream(filename))) {
            invited = (LinkedList<String>) ois.readObject();
    }catch(ClassNotFoundException e) {}
}

public void writePersons2(String filename) throws IOException {
    try (ObjectOutputStream oos = new ObjectOutputStream(
                new FileOutputStream(filename))) {
            oos.writeObject(persons);
            oos.flush();
    }
}

public void readPersons2(String filename) throws IOException {
    try (ObjectInputStream ois = new ObjectInputStream(
                new FileInputStream(filename))) {
            persons = (ArrayList<Person>) ois.readObject();
    }catch(ClassNotFoundException e) {}
}
```

Uppgift 6

```
public class Exercise6 {
    public void copyFile(String source) {
        String dest = source+" copy";
        try ( BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(source));
              BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(dest))) {
            int data = bis.read();
            while(data!=-1) {
                bos.write(data);
                data = bis.read();
            }
            bos.flush();
        } catch(IOException e) {
            System.err.println(e.toString());
        }
    }

    public void copyFile2(String source) {
        String dest = source+" copy";
        try ( BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(source));
              BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(dest))) {
            Exercise6.copy(bis,bos);
        } catch(IOException e) {
            System.err.println(e.toString());
        }
    }

    public static void copy(InputStream is, OutputStream os) throws IOException {
        int data = is.read();
        while(data!=-1) {
            os.write(data);
            data = is.read();
        }
        os.flush();
```

```
    }


    public void zip(File[] files) {
        String filename;
        String directories = files[0].getParent();
        String archivename = directories+"/achive.zip";
        try (ZipOutputStream zos = new ZipOutputStream(new BufferedOutputStream(new
FileOutputStream(archivename)))) {
            for(File file : files) {
                try (BufferedInputStream bis = new BufferedInputStream(new
FileInputStream(file))) {
                    filename = file.getName();
                    zos.putNextEntry(new ZipEntry(filename));
                    Exercise6.copy(bis,zos);
                    zos.closeEntry();
                    zos.flush();
                } catch(IOException e) {
                    System.err.println(e);
                }
            }
        } catch(IOException e) {
            System.err.println(e);
        }
    }

    public void unzip(File file) {
        String directoryName = withoutSuffix(file.getPath());
        File directory = new File(directoryName);
        directory.mkdir();
        ZipEntry zipEntry;
        try ( ZipInputStream zis = new ZipInputStream(new BufferedInputStream(new
FileInputStream(file)))) {
            while((zipEntry = zis.getNextEntry())!=null) {
                try (BufferedOutputStream bos = new BufferedOutputStream(new
FileOutputStream(new File(directory,zipEntry.getName())))) {
                    Exercise6.copy(zis,bos);
                } catch(IOException e) {
                    System.err.println(e);
                }
            }
        } catch(IOException e) {
            System.err.println(e);
        }
    }

    private String withoutSuffix(String filename) {
        int dotIndex = filename.indexOf(".");
        if(dotIndex==-1)
            return filename;
        return filename.substring(0,dotIndex);
    }
}
```