ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
МЭДЭЭЛЭЛ, ХОЛБООНЫ ТЕХНОЛОГИЙН СУРГУУЛЬ

F.CS202
ОБЪЕКТ ХАНДЛАГАТ ПРОГРАМЧЛАЛ

Лекц №8

Алдаа боловсруулах, файлтай ажиллах

док., дэд проф. Б.Батзолбоо
маг. Б.Мөнхбуян

2021 он

Агуулга

- Алдаа барих хэлбэр
- Олон алдаа барих, шатлал
- Алдааны төрлүүд
- Finally блок ашиглах
- Файлтай ажиллах

Өмнөх лекцээр

- Массивтай ажиллах
- Жагсаалт/List
- Буулгалт/Map
- Dictionary

Алдаа/exception

- Програмын энгийн ажиллагааг тасалдуулагч event
- Програмд алдаа гарахад програмын ажиллагааг зогсооно
- Системийн алдааны мэдээллийг харуулна
- Алдааг барьснаар програмын ажиллагаа зогсохоос сэргийлж, алдааны мэдээллийг хэрэглэгчид ойлгомжтой байдлаар харуулах боломжтой болно.

Жишээ – Тоог тэгд хуваах

```
class DivByZero
{ public static void main(String args[])
  {  System.out.println("Hello");
     int a=0;
     int d=10/a;
     System.out.println(d);
     System.out.println("Pls. print me.");
  }
}
```

Үр дүн

```
Hello
  Exception in thread "main"
     java.lang.ArithmeticException: / by zero
          at DivByZero.main(DivByZero.java:5)
```

Алдаа гарахад алдааны мэдээллийг харуулж програмаас гарна.

Иймээс алдаа гарч болох кодуудад дараах кодын блокыг бичнэ.

Алдаа барих ерөнхий хэлбэр

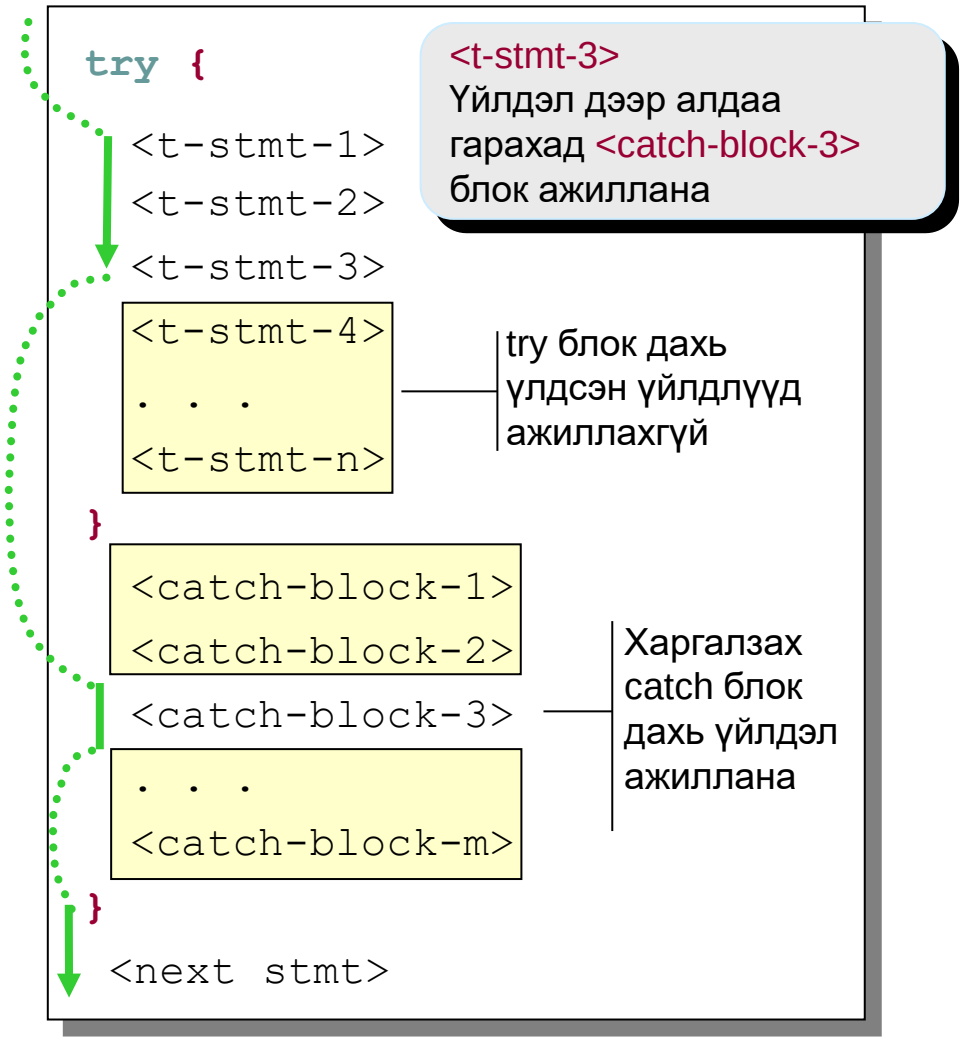
```
try
{
    // алдаа гарч болох програмын кодууд
}
catch (ExceptionType1 exOb)
{
    // алдааг удирдах хэсэг
}
```

Олон алдаа барих

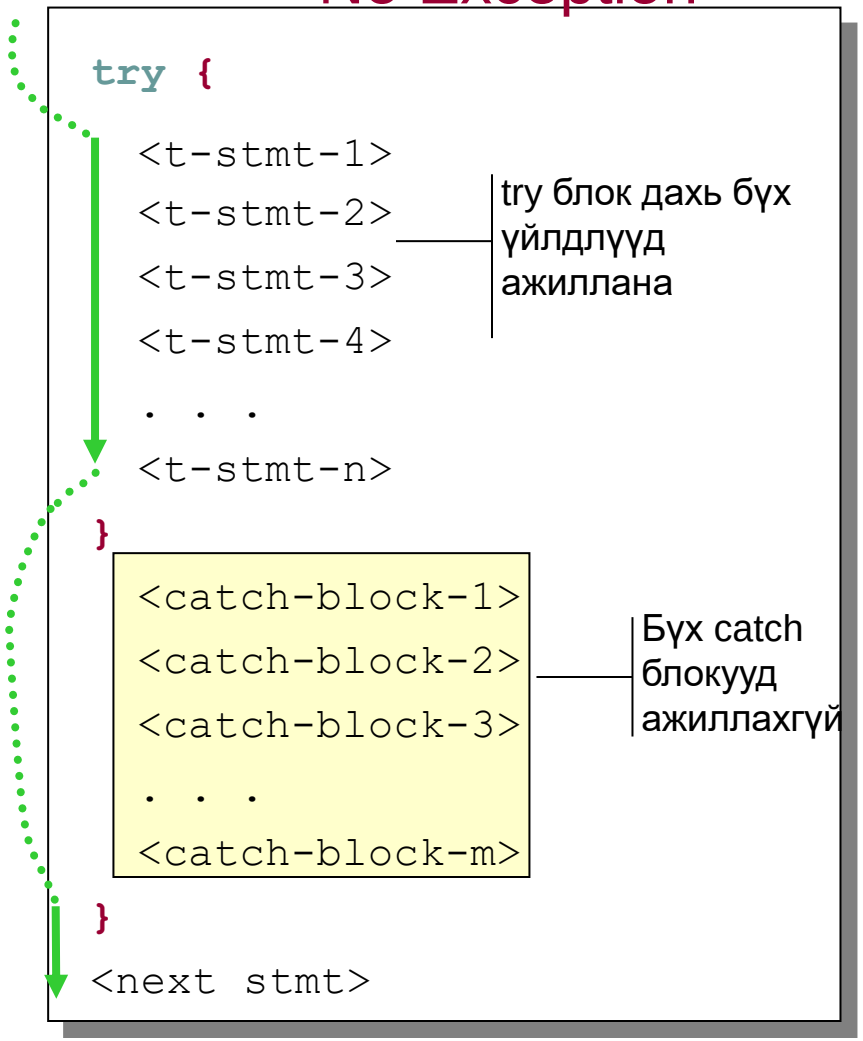
```
try
{
     // алдаа гарч болох програмын кодууд
}
catch (ExceptionType1 exOb)
{
     // ExceptionType1 төрлийн алдааг удирдах хэсэг
}
catch (ExceptionType2 exOb)
{
     // ExceptionType2 төрлийн алдааг удирдах хэсэг
}
```

Олон алдаа барих

Exception



No Exception



Алдаа

- Бичлэгийн алдаа /syntax error
- Утгын алдаа / semantic error - Буруу алгоритм
- Ажиллах үеийн алдаа / run time error - Програм ажиллах үед төхөөрөмжүүдийн нөхцөл байдлаас болж үүсдэг.
  - Тэгд хуваах x/z.
  - Массивын байхгүй индекст хандах a[i].
  - Буруу оролт.
  - Файлын зам буруу байхад нээх.
  - Хатуу диск эвдрэх, ...
- Алдаа гарах үед exception объект үүсдэг. Үүнд алдааны тухай мэдээллийг агуулдаг.

Алдаа барих шатлал

```
try
{      …….            }
catch(ArithmeticException e1)
{      //арифметик алдаа үүсэхэд      }
catch(NullPointerException e2)
{      ……..           }
catch(FileNotFoundException e3)
{      ……..    }
catch(IOException e4)
{      ……..    }
catch(RuntimeException e5)
{      ……..    }
catch(Exception e6)
{      //Бүх алдааг барьж чадна. Өмнөх блокуудаар алдаа барьж чадаагүй үед
}
```

Алдааг барих үед

1. Програм ажиллана.
2. Алдаа гарах эрсдэл үүснэ. Жишээлбэл тэгд хуваах
3. Java virtual machine exception объект үүсгэнэ.
4. Exception объектод алдааны тухай мэдээллийг оруулна.
5. Алдаа гарсан мөр лүү exception объектыг шиднэ. Хэрэв try блок дотор алдаа гарвал алдаа баригдана. Хэрэв try блок байхгүй бол алдааг мэдээлж програмаас гарна.

Жишээ ArithmeticException

```
public class Test
{     public static void main()
      {     Scanner in=new Scanner(System.in);
            int a=50, b, c;
            System.out.print("b=");
            b=in.nextInt();
            try
            {     c=a/b;
                  System.out.println(c);
            }
            catch(ArithmeticException obj)
            {     System.out.println("Aldaa");  }
      }
}
```

b-д 0 гэж өгвөл алдаа гарна.

Жишээ NullPointerException

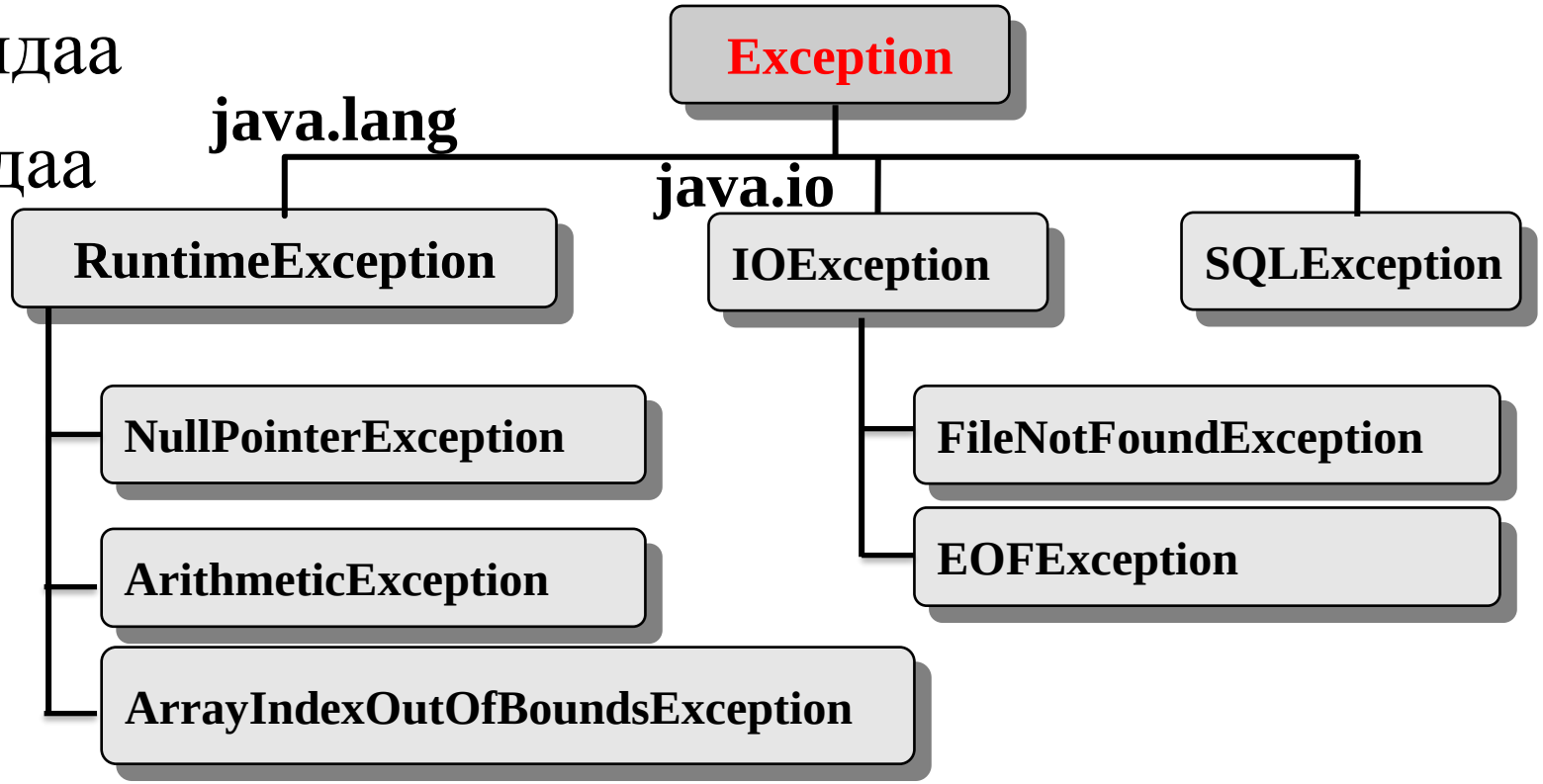
```
public class Program
{   public static void main()
    {
        Teacher t[] = new Teacher[1];
        t[0].toString();
    }
}
```

Жишээ ArrayIndexOutOfBoundsException

```
public class Program
{     public static void)
      {     int []d={1, 2, 3, 4};
            int s=0;
            try
            {     for(int i=0;i<=d.length;i++)
                  {     System.out.print(d[i]+" ");
                        s=s+d[i];
                  }
            }
            catch(ArrayIndexOutOfBoundsException ob)
            {     System.out.println("\nindex error:"+ob);
            }
            System.out.println("Sum="+s);
      }
}
```

Алдааны төрөл

- Санах ойн алдаа
- Үйлдлийн системийн алдаа
- Техник хангамжийн алдаа
- Java virtual machine алдаа
- Сүлжээний алдаа



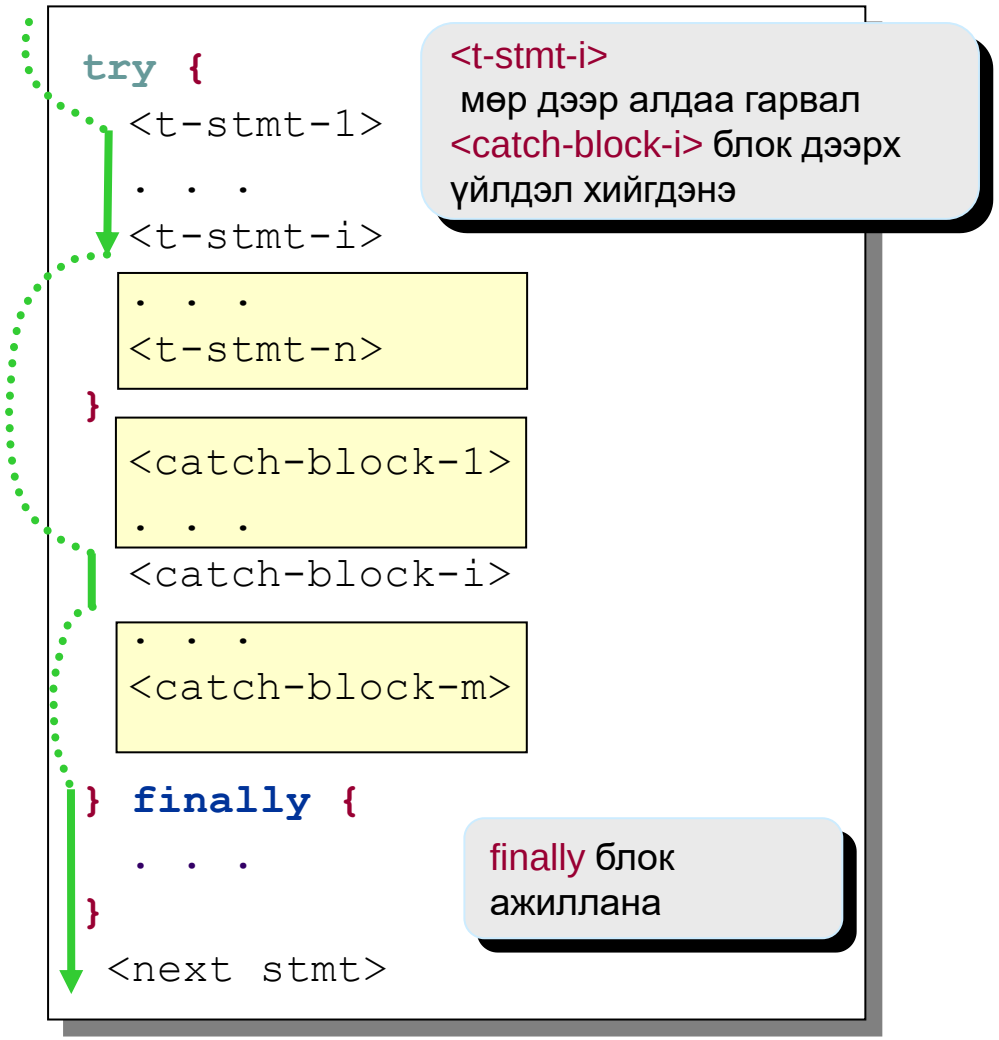
Finally блок

```
try
{
      //алдаа үүсгэх магадлалтай код
}
catch(ExceptionType1 e1)
{
      //алдаа үүссэн үед хийх үйлдэл
}
catch(ExceptionType2 e2)
{
      //алдаа үүссэн үед хийх үйлдэл
}
finally
{
      //үргэлж хамгийн сүүлд ажиллана.алдаа гарсан ч дараа нь ажиллана.
}
```

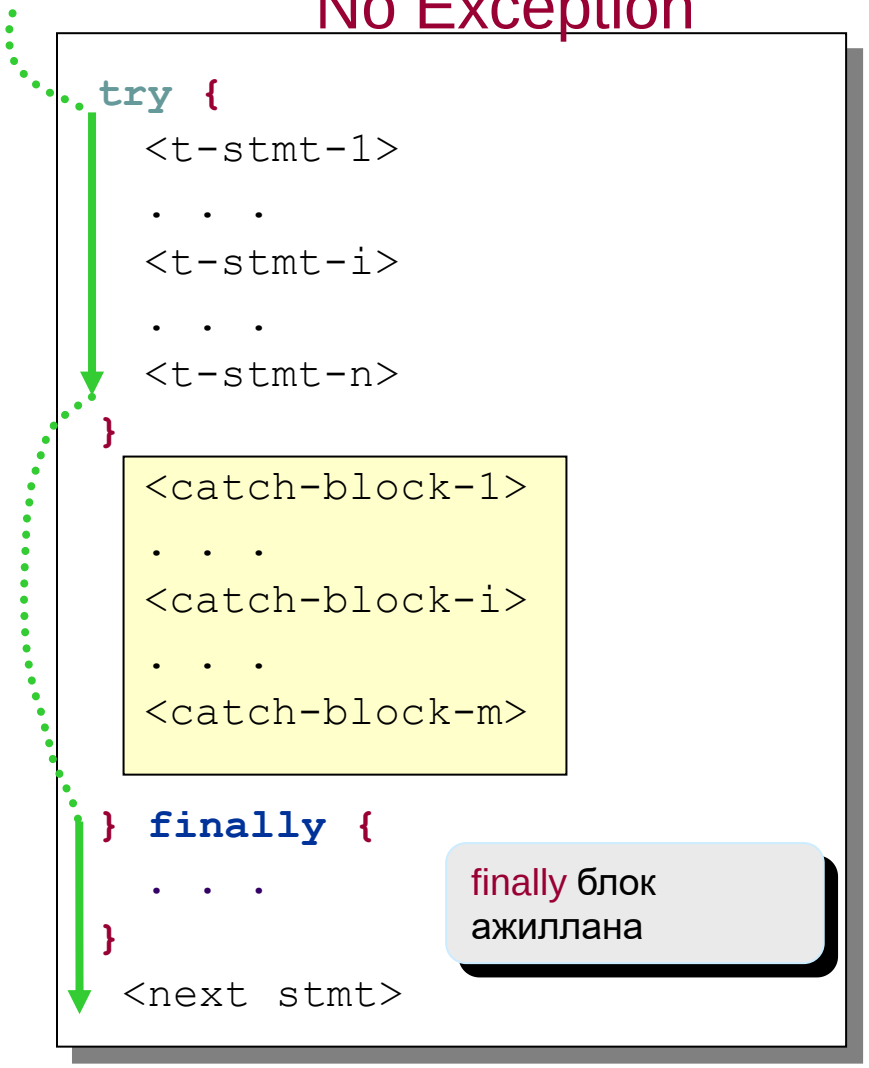
объект

try-catch-finally - ажиллагаа

Exception



No Exception



Файлтай ажиллах

```
import java.io.*;

public class aa
{
      public static void main() {
            File inFile = new File("sample.data");

            if (inFile.exists( )) {
                // файл байна.
            } else {
                // файл байхгүй.
      ...
      }
}
```

Файлын нэр зам

- Windows

  “C:\JavaPrograms”

- Unix

  "/JavaPrograms“

- Mac

  “/MacHD/JavaPrograms“

- идэвхтэй директортой харьцангуй зам
  - "../Ch12"

File методууд

```
if ( inFile.exists( ) ) {
```

inFile нь байгаа эсэх

```
if ( inFile.isFile() ) {
```

inFile нь файл мөн эсэх. Худал утга буцвал энэ нь хавтас байна.

```
File directory = new
        File("C:/JavaPrograms/Ch12");

String filename[] = directory.list();

for (int i = 0; i < filename.length; i++) {
    System.out.println(filename[i]);
}
```

C:\JavaProjects\Ch12 хавтаст байрлаж буй файлуудын жагсаалтыг харах

Low-Level File I/O

- Файлаас өгөгдөл унших, бичихийн тулд Java stream объект үүсгэж, түүндээ файлаа холбоно.
- *stream* гэдэг нь өгөгдлийн нэгжүүдийн дараалал бөгөөд ихэвчлэн 8-bit байтууд байна.
- Жава нь *input stream* , *output stream* гэсэн 2 stream-тай.
- *input stream* нь өгөгдлийн ирсэн эх сурвалж, *output stream* нь өгөгдлийг илгээх байрлалын сурвалжийг хадгална.

Low-Level File I/O

- **FileOutputStream** нь байтуудын дарааллыг гаралтанд гаргах
- **FileInputStream** нь байтуудын дарааллыг унших

Жишээ: Low-Level Файлын гаралт

```
//файл болон stream-г үүсгэх
File  outFile   = new File("sample1.data");

FileOutputStream
      outStream = new FileOutputStream( outFile );

//хадгалах өгөгдөл
byte[] byteArray = {10, 20, 30, 40,
                    50, 60, 70, 80};

//stream-д өгөгдлийг бичих
outStream.write( byteArray );

//гаралт дуусахад stream-г хаана
outStream.close();
```

Жишээ: Low-Level Файлын гаралт

```
//файл болон stream-г үүсгэх
File            inFile   = new File("sample1.data");
FileInputStream inStream = new FileInputStream(inFile);

//өгөгдлийг уншиж хадгалах хүснэгтийг үүсгэх
int    fileSize  = (int)inFile.length();
byte[] byteArray = new byte[fileSize];

//өгөгдлийг уншиж дэлгэцэнд харуулах
inStream.read(byteArray);
for (int i = 0; i < fileSize; i++) {
        System.out.println(byteArray[i]);
}

//оролт дуусахад stream-г хаах
inStream.close();
```

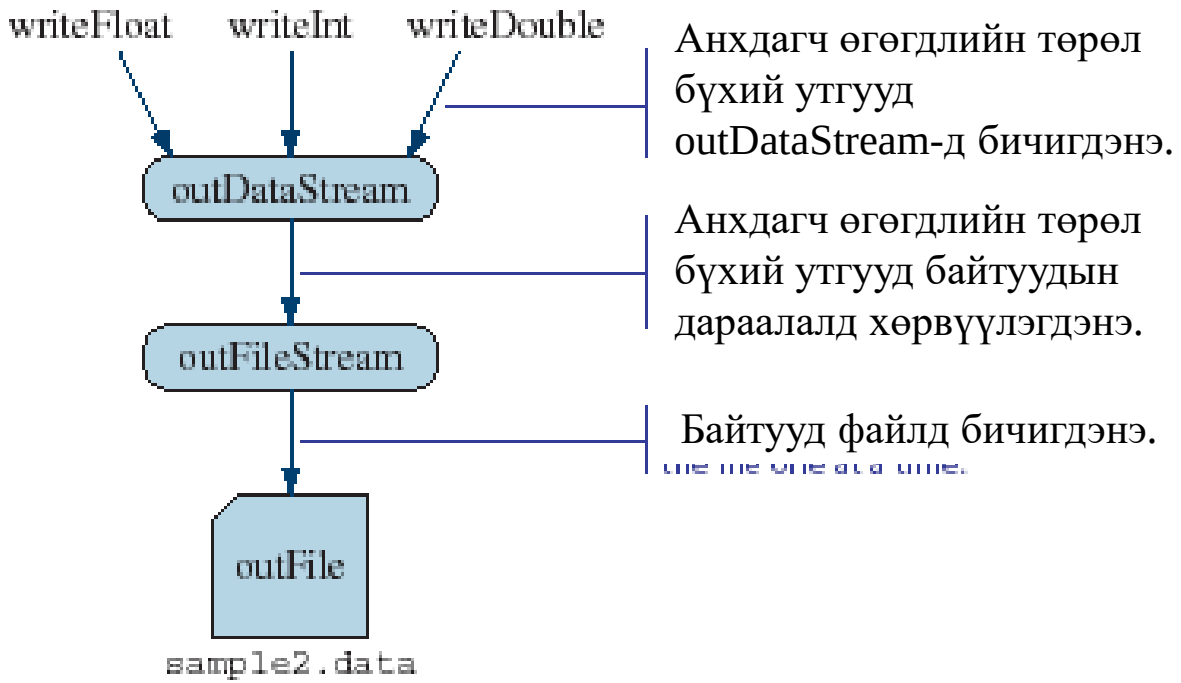
High-Level File I/O-д ашиглагдах stream-үүд

- FileOutputStream, DataOutputStream нь анхдагч өгөгдлийн төрлүүд бүхий утгуудыг гаралтад гаргахад ашиглагдана.
- FileInputStream, DataInputStream анхдагч өгөгдлийн төрлүүд бүхий утгуудыг оруулахад ашиглагдана.
- Өгөгдлийг буцааж зөв уншихын тулд өгөгдлийн хадгалагдсан дараалал болон өгөгдлийн төрлүүдийг мэдэж байх шаардлагатай.

DataOutputStream

- Стандарт DataOutputStream объект үүсгэх

```
File             outFile       = new File( "sample2.data" );
FileOutputStream outFileStream = new FileOutputStream(outFile);
DataOutputStream outDataStream = new DataOutputSteam(outFileStream);
```



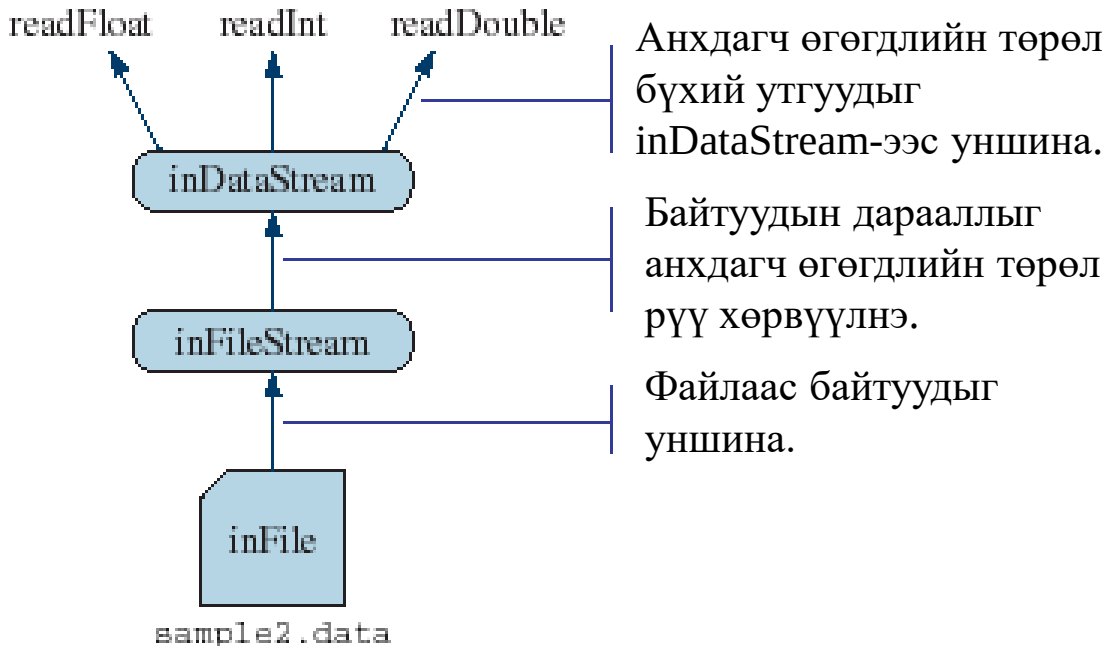
Жишээ - Output

```
import java.io.*;
class Ch12TestDataOutputStream {
    public static void main (String[] args) throws IOException {

        . . . //outDataStream үүсгэх

        //stream-д анхдагч өгөгдлийн төрөл бүхий өгөгдлийг бичих
        outDataStream.writeInt(987654321);
        outDataStream.writeLong(11111111L);
        outDataStream.writeFloat(22222222F);
        outDataStream.writeDouble(3333333D);
        outDataStream.writeChar('A');
        outDataStream.writeBoolean(true);

        //stream-г хаах
        outDataStream.close();
    }
}
```

DataInputStream үүсгэх

- DataInputStream объект үүсгэх стандарт дараалал:

```
File            inFile       = new File( "sample2.data" );
FileOutputStream inFileStream = new FileOutputStream(inFile);
DataOutputStream inDataStream = new DataOutputSteam(inFileStream);
```

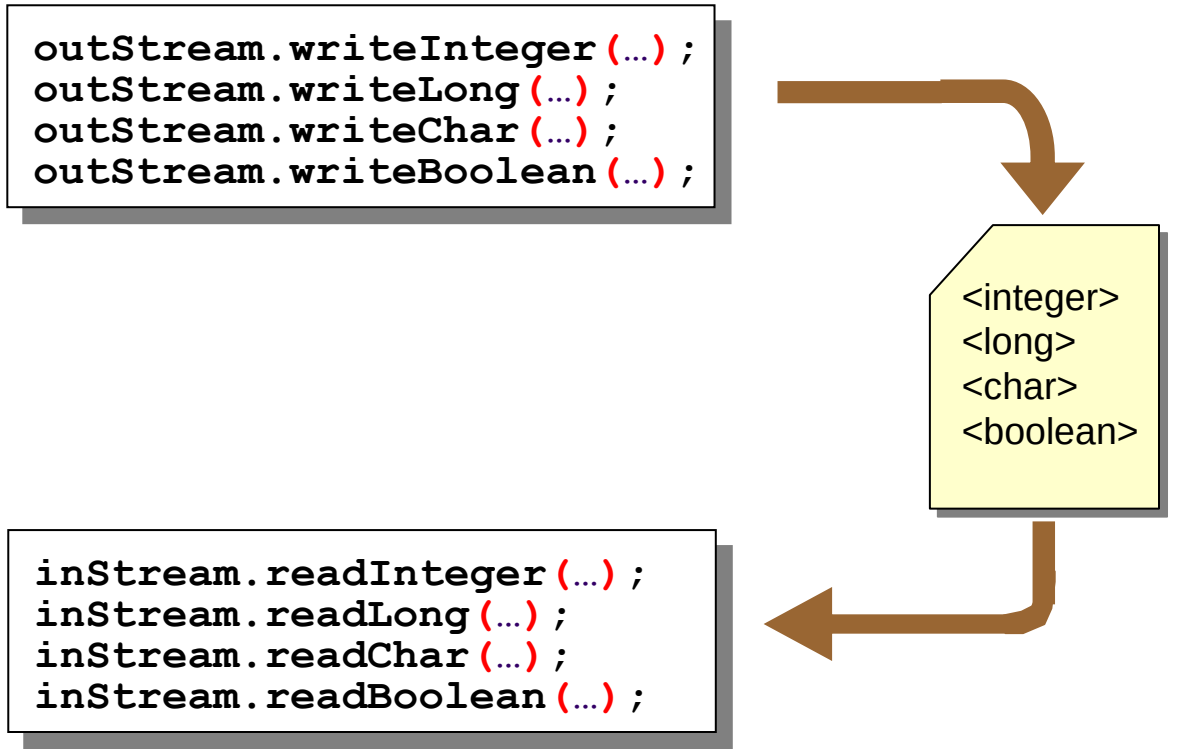


Жишээ - Input

```
import java.io.*;
class Ch12TestDataInputStream {
    public static void main (String[] args) throws IOException {

        . . . //inDataStream үүсгэнэ

        //stream-ээс өгөгдлүүдийг уншиж дэлгэцэнд харуулах
        System.out.println(inDataStream.readInt());
        System.out.println(inDataStream.readLong());
        System.out.println(inDataStream.readFloat());
        System.out.println(inDataStream.readDouble());
        System.out.println(inDataStream.readChar());
        System.out.println(inDataStream.readBoolean());

        //stream-г хаах
        inDataStream.close();
    }
}
```

Өгөгдлийг зөв дарааллаар унших

- Бичих, унших дараалал дахь өгөгдлийн төрлүүдийн дараалал тохирч байх ёстой.



Текст файл бичих, унших

- Анхдагч өгөгдлийн төрөл бүхий өгөгдлүүдийг файлд хоёртын өгөгдөл хэлбэрээр хадгалахаас гадна, тэдгээрийг тэмдэгт өгөгдөл болгон хөрвүүлж хадгалах боломжтой.
  - Дурын текст засварлагч ашиглан файлын агуулгыг харах боломжтой болно.
- Өгөгдлийг файл руу текст хэлбэрээ хадгалахын тулд PrintWriter объектыг ашиглана.
- Текст файлаас өгөгдөл оруулахын тулд, FileReader болон BufferedReader классуудыг ашиглана
  - Java 5.0 (SDK 1.5), хувилбараас эхлэн текст файлын оролтонд Scanner классыг ашиглах болсон.

Текст файлын гаралт

```
import java.io.*;
class Ch12TestPrintWriter {
    public static void main (String[] args) throws IOException {

        //файл болон stream-ээ үүсгэх
        File outFile = new File("sample3.data");
        FileOutputStream outFileStream
                        = new FileOutputStream(outFile);
        PrintWriter outStream = new PrintWriter(outFileStream);

        // stream-д үндсэн өгөгдлийн төрөл бүхий өгөгдөл бичих
        outStream.println(987654321);
        outStream.println("Hello, world.");
        outStream.println(true);

        //stream-г хаах
        outStream.close();
    }
}
```

Текст файлын оролт

```
import java.io.*;
class Ch12TestBufferedReader {

    public static void main (String[] args) throws IOException {

        //файл болон stream-г үүсгэнэ
        File inFile = new File("sample3.data");
        FileReader fileReader = new FileReader(inFile);
        BufferedReader bufReader = new BufferedReader(fileReader);
        String str;

        str = bufReader.readLine();
        int i = Integer.parseInt(str);

        //бусад өгөгдлийн төрөл дээр ижил үйлдэл хийгдэнэ

        bufReader.close();
    }
}
```

Scanner ашиглан текст файл оруулах

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

class Ch12TestScanner {

    public static void main (String[] args) throws IOException {

        //Scanner-аа нээх
        Scanner scanner = new Scanner(new File("sample3.data"));

        //бүхэл тоон утга авах
        int i = scanner.nextInt();

        // бусад өгөгдлийн төрөл дээр ижил үйлдэл хийгдэнэ

        scanner.close();
    }
}
```

Object File I/O

- Объектыг хадгалах нь үндсэн өгөгдлийн төрөл бүхий өгөгдөл хадгалахтай ижил хялбархан.
- Файлд объект бичих, уншихад ObjectOutputStream, ObjectInputStream –г ашиглана.
- Тухайн классын объектыг хадгалахын тулд класс зарлалтанд implements Serializable кодыг бичсэн байх ёстой. Жишээ нь:

```
class Person implements Serializable {
  . . .
}
```

Объект хадгалах

```
File                  outFile
                          = new File("objects.data");
FileOutputStream      outFileStream
                          = new FileOutputStream(outFile);
ObjectOutputStream outObjectStream
                          = new ObjectOutputStream(outFileStream);
```

```
Person person = new Person("Mr. Espresso", 20, 'M');

outObjectStream.writeObject( person );
```

```
account1    = new Account();
bank1       = new Bank();

outObjectStream.writeObject( account1 );
outObjectStream.writeObject( bank1    );
```

Өөр өөр классуудын объектыг хадгалж болно.

Объектыг унших

```
File                  inFile
                          = new File("objects.data");
FileInputStream       inFileStream
                          = new FileInputStream(inFile);
ObjectInputStream inObjectStream
                          = new ObjectInputStream(inFileStream);
```

```
Person person
      = (Person) inObjectStream.readObject( );
```

Зөв объектын төрлөөр уншихын тулд хувиргалт хийнэ.

```
Account account1
       = (Account) inObjectStream.readObject( );
Bank bank1
       = (Bank) inObjectStream.readObject( );
```

Зөв дарааллаар уншина

Массив хадгалах, дуудах

- Массивын элементүүдийг нэг нэгээр нь хандаж ажиллахаас гадна массивыг бүхлээр нь хадгалж , унших боломжтой.

```
Person[] people = new Person[ N ];
            //N-л утга олгогдсон гэж үзье.

//people хүснэгтэнд утгуудыг оруулна
. . .
//хүснэгтийг хадгална
outObjectStream.writeObject ( people );
```

```
//хүснэгтийг унших
Person[ ] people = (Person[]) inObjectStream.readObject( );
```

ДҮГНЭЛТ

- Програмын хэвийн ажиллагааг тасалдуулахгүйн тулд алдааг барина.
- Алдаа барих блок бичихдээ алдааны түвшингийн дарааллыг баримтална.
- Finally блок нь try блокт алдаа гарсан эсэхээс үл хамааран ажиллана
- Байт код, тэмдэгт, объект, объект массив хэлбэрээр өгөгдлийг файлд бичиж, унших боломжтой
- Файлаас унших үед бичигдсэн дарааллаар нь уншина.