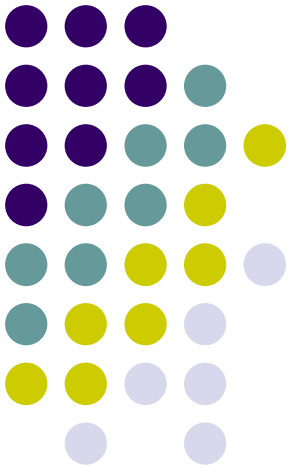
第１１回　独習JAVAゼミ

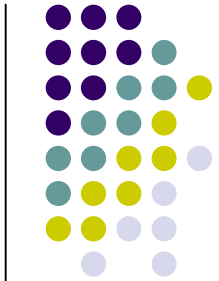
11.4　PrintWriterクラス

11.5　バイトストリーム

発表者：鈴木朋央



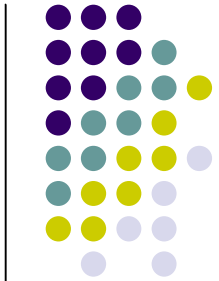
PrintWriterクラスの概要



- Writerを拡張したクラス
- Int, float, charなどの基本データ型およびオブジェクトと等価の文字列を表示
- 各種のデータ型を出力できる共通インターフェイスを提供
- Objectを含むすべての型についてprint()メソッドとprintln()メソッドをサポート

  → 引数が基本データ型でない場合、オブジェクトのtoString()メソッドを呼び出し、返された文字列を表示

PrintWriterコンストラクタ



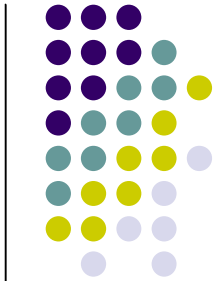
- PrintWriter(OutputStream outputStream)
- PrintWriter(OutputStream outputStream,
  boolean flushOnNewLine)
- PrintWriter(Writer writer)
- PrintWriter(Writer writer, boolean flushOnNewLine)

＊flushOnNewLine
→ 改行文字が出力されるたびに出力ストリームをフラッシュするか制御

真：自動的にフラッシュされる

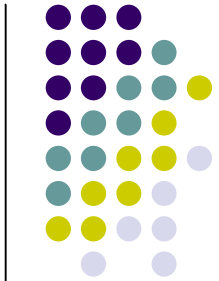
偽：自動的にフラッシュされない

PrintWriter使用例(1/2)



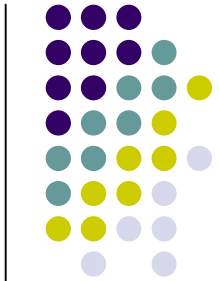
```
Import java.io.*;
Class PrintWirterDemo{
   public static void main(String args[]){
        try {
        PrintWriter pw = new PrintWriter(System.out);
        pw.println(true); //いくつかメソッドを実行
        pw.println('A');
        pw.println(500);
        pw.println(40000L);
        pw.println(45.67f);
```

PrintWriter使用例(2/2)



```
        pw.println(45.67);
        pw.println("Hello");
        pw.println(new Integer("99"));
        pw.close();  //オブジェクトのクローズ
        }
    catch (Exception e){
        System.out.println("Exception: " + e);
        }
    }
}
```

出力結果



pw.println(true); → true

pw.println('A'); → A

pw.println(500); → 500

pw.println(40000L); → 40000

pw.println(45.67f); → 45.67

pw.println(45.67); → 45.67

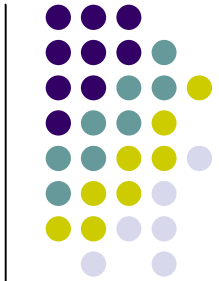
pw.println("Hello"); → Hello

pw.println(new Integer("99")); → 99

バイトストリーム

入力	出力
InputStream	OutputStream
FileInputStream	FileOutputStream
FilterInputStream	FilterOutputStream
BufferedInputStream	BufferedOutputStream
DataInputStream	DataOutputStream
	PrintStream

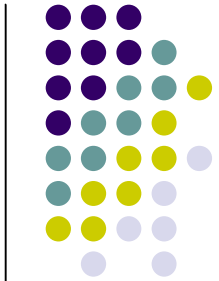
OutputStreamクラス



OutputStreamクラスに定義されているインスタンスメソッド

メソッド	説明
void close() throws IOException	出力ストリームをクローズする
void flush() throws IOException	出力ストリームをフラッシュする
void write(int i) throws IOException	ストリームにiの下位8ビットを書き込む
void write (byte buffer[ ]) throws IOException	ストリームにbufferを書き込む
void write(byte buffer[ ], int index, int size) throws IOException	位置indexを先頭としてbufferからsizeバイトをストリームに書き込む

FileOutputStreamクラス



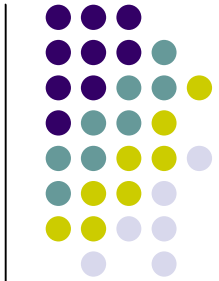
- FileOutputStream(String filepath) throws IOException
- FileOutputStream(String filepath, boolean append) throws IOException
- FileOutputStream(File fileobj) throws IOException

filepath:ファイルの完全パス名

fileObj:ファイルを表すFileオブジェクト

Append { 真：文字はファイルの終わりに付加
偽：ファイルの既存の内容を上書き

FilterOutputStreamクラス



- FilterOutputStream(OutputStream os)

  os：フィルタにかける出力ストリーム

- OutputStreamを拡張したクラスであり、出力をフィルタにかける場合に使用する
- 必要な機能を実装するには、FilterOutputStreamのインスタンスを直接作成するのではなく、サブクラスを作成しなければならない

BufferedOutputStreamクラス

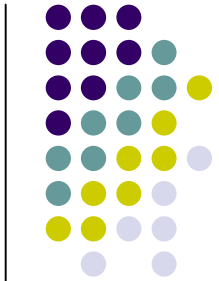
- BufferedOutputStream(OutputStream os)
- BufferedOutputStream(OutputStream os, int bufSize)

- FilterOutputStreamを拡張したクラス
- バイトストリームへの出力をバッファに入れる
- バッファのサイズ

  → 1行目：既定サイズのバッファを使用

  2行目：バッファのサイズをbufSizeで指定

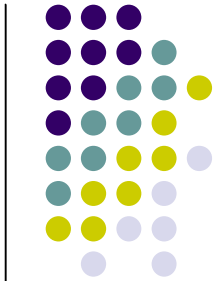
DataOutputStreamクラス



- DataOutputStream(OutputStream os)

- FilterOutputStreamを拡張したクラス
- DataOutputを実装していることにより、基本データ型をバイト出力ストリームに書き込むことが可能

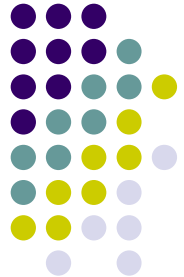
DataOutputインターフェイス(1/2)



- 定義されているインスタンスメソッド

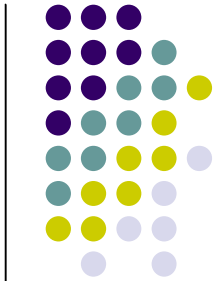
メソッド	説明
void flush() throws IOException	ストリームをフラッシュする
int size()	書き込み済みのバイト数を返す
void write(int i) throws IOException	ストリームにiを書き込む
void write(byte buffer[ ]) throws IOException	ストリームにbufferを書き込む
void write(byte buffer[ ], int index, int size) throws IOException	位置indexを先頭としてbufferからsizeバイトをストリームに書き込む
void writeBoolean(boolean b) throws IOException	ストリームにbを書き込む
void writeByte(int i) throws IOException	ストリームにiの下位8ビットを書き込む
void writeBytes(String s) throws IOException	ストリームにsを書き込む

DataOutputインターフェイス(2/2)



void writeChar(int i) throws IOException	ストリームにiの下位16ビットを書き込む
void writeChars(String s) throws IOException	ストリームにsを書き込む
void writeDouble(double d) throws IOException	ストリームにdを書き込む
void writeFloat(float f) throws IOException	ストリームにfを書き込む
void writeInt(int i) throws IOException	ストリームにiを書き込む
void writeLong(long l) throws IOException	ストリームにlを書き込む
void writeShort(short s) throws IOException	ストリームにsを書き込む
void writeUTF(String s) throws IOException	ストリームにsを書き込む。文字はunicodeからUTF-8エンコーディングに変換される

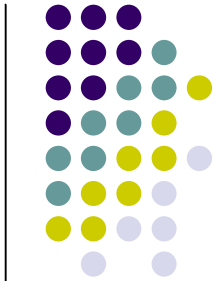
PrintStreamクラス



- PrintStream(OutputStream outputStream)
- PrintStream(OutputStream outputStream, boolean flushOnNewline)
- PrintStream(OutputStream outputStream, String encoding)

- FilterOutputStreamを拡張したクラス
- これまでSystem.outをとおして使用してきたすべての書式設定機能を提供する

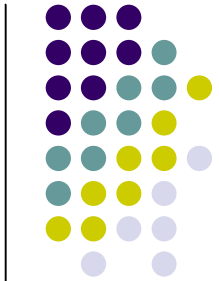
InputStreamクラス



- InputStreamクラスに定義されているインスタンスメソッド

メソッド	説明
int available() throws IOException	現時点で読み取りに利用できるバイト数を返す
void close() throws IOException	入力ストリームをクローズする
void mark(int numBytes)	入力ストリームの現在の位置にマークをつける。numBytesが読み取られるまで有効とする
boolean markSupported()	mark()/reset()がサポートされている場合は真を返し、サポートされていない場合は偽を返す。
int read() throws IOException	入力ストリームから1バイトを読み取る
int read(byte buffer[ ]) throws IOException	buffer, lengthバイトまでバッファに読み取ることを試み、正しく読み取った実際のバイト数を返す。
int read(byte buffer[ ], int offset, int numBytes) throws IOException	位置buffer[offset]を先頭としてバッファにnumBytesバイトまでを書き込むことを試みる。正しく読み取った実際のバイト数を返す。
void reset() throws IOException	入力ポインタを、事前に設定されているマークにリセットする。
int skip(long numBytes) throws IOException	入力のうちのnumBytesバイトを飛ばす。実際に飛ばしたバイト数を返す。

FileInputStreamクラス



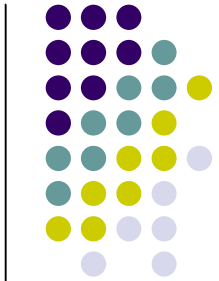
- FileInputStream(String filepath) throws FileNotFoundException
- FileInputStream(File fileObj) throws FileNotFoundException

  filepath:ファイルの完全パス名

  fileObj:ファイルを表すFileオブジェクト

- InputStreamを拡張したクラス
- ファイルからバイナリデータを読み取ることが可能

FilterInputStreamクラス

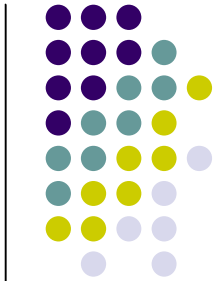


- FilterInputStream(InputStream is)

  is:フィルタにかける入力ストリーム

- InputStreamを拡張したクラス
- FilterInputStreamのインスタンスを直接作成するのではなく、必要な機能を実装するサブクラスを作成しなければならない

BufferedInputStreamクラス



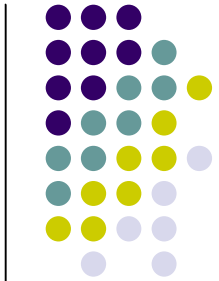
- BufferedInputStream(InputStream is)
- BufferedInputStream(InputStream is, int bufsize)

- FilterInputStreamを拡張したクラス
- バイトストリームからの入力をバッファに入れる
- バッファのサイズ

  → 1行目：既定サイズのバッファを使用

  2行目：バッファのサイズをbufSizeで指定

DataInputStreamクラス



- DataInputStream(InputStream is)

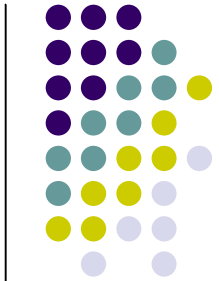
- FilterInputStreamを拡張したクラス
- DataInputを実装していることにより、バイト入力ストリームから単純なJava型を読み取ることが可能

DataInputインターフェイス(1/2)

- 定義されているインスタンスメソッド

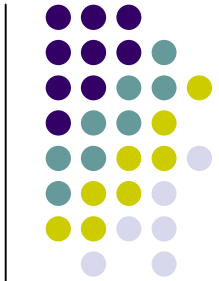
メソッド	説明
int read(byte buffer[ ]) throws IOException	buffer, lengthバイトまでバッファに読み取ることを試み、正しく読み取った実際のバイト数を返す。
int read(byte buffer[ ], int offset, int numBytes) throws IOException	位置buffer[offset]を先頭としてバッファにnumBytesバイトまでを書き込むことを試みる。正しく読み取った実際のバイト数を返す。
boolean readBoolean() throws IOException	ストリームからboolean型を読み取って返す
byte readByte() throws IOException	ストリームからbyte型を読み取って返す
char readChar() throws IOException	ストリームからchar型を読み取って返す
double readDouble() throws IOException	ストリームからdouble型を読み取って返す。
float readFloat() throws IOException	ストリームからfloat型を読み取って返す。
void readFully(byte buffer[ ]) throws IOException	バイトを読み取り、bufferに入れる

DataInputインターフェイス(2/2)



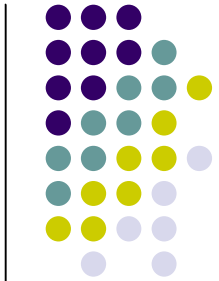
void readFully(byte buffer[ ], int index, int size) throws IOException	sizeバイトを読み取り、位置indexを先頭としてバッファに入れる。
int readInt() throws IOException	ストリームからint型を読み取って返す。
long readLong() throws IOException	ストリームからlong型を読み取って返す。
short readShort() throws IOException	ストリームからshort型を読み取って返す。
int readUnsignedByte() throws IOException	ストリームから符号なしbyte型を読み取って返す。
int readUnsignedShort() throws IOException	ストリームから符号なしshort型を読み取って返す。
String readUTF() throws IOException	入力から文字列を読み取る。文字はUTF-8形式からUnicodeに変換される。メソッドからは文字列が返される。
int skipBytes(int n) throws IOException	ストリーム内の先頭からnバイトを飛ばす。

FileOutputStream使用例



```
Import java.io.*;
Class FileOutputStreamDemo{
    public static void main(String args[]){
        try{
        FileOutputStream fos = new FileOutputStream(args[]);
                for(int i=0; i<12; i++){  //ファイルに１２バイト書き込む
                        fos.write(i);
                }
        fos.close();
        }
        catch (Exception e){
        System.out.println(“Exception: “ + e);}}}
```

FileInputStream使用例



```
try{
FileInputStream fis = new FileInputStream(args[]);
int i;
while((i = fis.read()) != -1){ //データを読み取って表示
    System.out.println(i);
}
fis.close();
catch (Exception e){
    System.out.println("Exception: " + e);
}
```

BufferedOutputStream使用例

```
try{
FileOutputStream fos = new FileOutputStream(args[0]);
BufferedOutputStream bos = new BufferedOutputStream(fos);
for(int i=0; i<12; i++){  //ファイルに１２バイト書き込む
    bos.write(i);
}
bos.close;
catch{
    System.out.println(“Exception: “ + e);
}
```

BufferedInputStream使用例

```
try{
FileInputStream fis = new FileInputStream(args[0]);
BufferedInputStream bis = new BufferedInputStream(fis);
int i;
while((i  = bis.read()) != -1){
      System.out.println(i);
}
fis.close();
catch (Exception e){
System.out.Println(“Exception: “ + e);
}
```

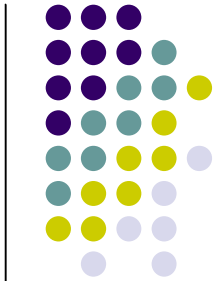
DataOutputStream使用例(1/2)

```
try{
FileOutputStream fos = new FileOutputStream(args[0]);
DataOutputStream dos = new DataOutputStream(fos);
dos.writeBoolean(false);
dos.writeByte(Byte.MAX_VALUE);
dos.writeChar('A');
dos.writeDouble(Double.MAX_VALUE);
dos.writeFloat(Float.MAX_VALUE);
dos.writeInt(Integer.MAX_VALUE);
dos.writeLong(Long.MAX_VALUE);
```

DataOutputStream使用例(2/2)

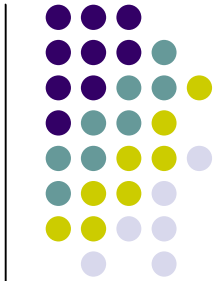
```
    dos.writeShort(Short.MAX_VALUE);
    fos.close();
}
catch (Exception e){
    System.out.println("Exception: " + e);
}
```

DataInputStream使用例(1/2)



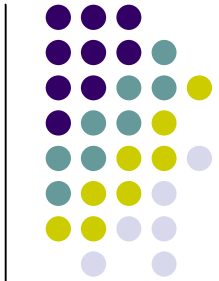
```
try{
FileInputStream fis = new FileInputStream(args[0]);
DataInputStream dis = new DataInputStream(fis);
System.out.println(dis.readBoolean());
System.out.println(dis.readByte());
System.out.println(dis.readChar());
System.out.println(dis.readDouble());
System.out.println(dis.readFloat());
System.out.println(dis.readInt());
System.out.println(dis.readLong());
```

DataInputStream使用例(2/2)



```
    System.out.println(dis.readShort());
    fis.close();
}
catch (Exception e){
    System.out.println("Exception: " + e);
}
```

出力結果(1/2)



System.out.println(dis.readBoolean()); → false

System.out.println(dis.readByte()); → 127

System.out.println(dis.readChar()); → A

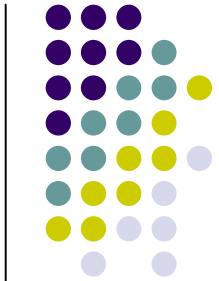
System.out.println(dis.readDouble());

→ 1.79769313486231570E308

System.out.println(dis.readFloat());

→ 3.4028235E38

出力結果(2/2)



System.out.println(dis.readInt());

→ 2147483647

System.out.println(dis.readLong());

→ 9223372936854775807

System.out.println(dis.readShort());

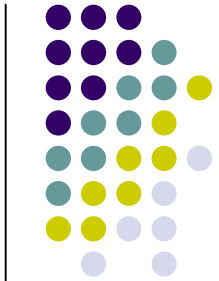
→ 32676

練習問題1

- 任意のファイルに、1から指定された数までの自然数を1行ずつ書き込むプログラムを作成せよ。
- プログラムの実行時に指定する引数は2つ。1つは出力ファイル名。もう1つは自然数。

＊必ずPrintWriterクラスを用いること

練習問題2



- 任意のテキストファイルを読み込み、内容をそのまま別のファイルに書き出しなさい。その際、バッファリングを行う場合と行わない場合の両方を作成しなさい。

- バッファリングを行うプログラムを作成する際に、ファイルのクローズをせずにプログラムを走らせるとどのような結果になるか、検討しなさい。