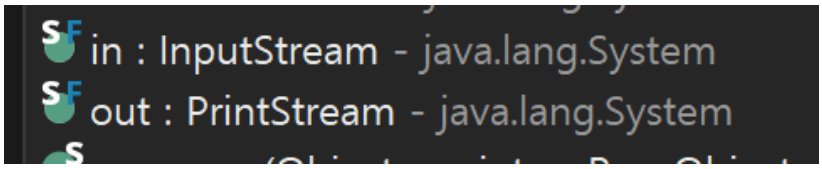
8_04_2022__java_Thread, InputOutput

Step15_thread

~Step16_InputOutput

new Thread(()- >{}); runnable interface

InputOutput



InputStream

- 키보드와 연결된 InputStream type 의 참조값을 kbd라는 지역변수에 담기
- InputStream 객체는 1byte 단위 처리 스트림이다.
- 영문자 대소문자, 숫자, 특수문자만 처리할 수 있다.
- 한글 처리 불가.

InputStreamReader

- 2byte 단위 처리 스트림
- 한글 처리 가능
- Reader type이기도 함

```
package test.main;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class MainClass02 {
  public static void main(String[] args) {
```

```
    //1byte 처리 스트림
    InputStream is = System.in;
    //2byte 처리 스트림
    InputStreamReader isr = new InputStreamReader(is);
    System.out.println("입력");
    isr.read()
    try {
      int code = isr.read();
      System.out.println("code : "+code);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

기능을 개선해서 한 글자 뿐 아니라 문자열을 받아오고 싶을 때

BufferedReader를 사용

- BufferedReader
  - InputStream, InputStreamReader 받아들임
  - int type을 리턴하기도 하고 `String type(문자열)을 리턴` 하기도 함

```
package test.main;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class MainClass03 {
  public static void main(String[] args) {
    InputStream is = System.in;
    InputStreamReader isr = new InputStreamReader(is);
    BufferedReader br = new BufferedReader(isr);
    System.out.println("문자열 입력 : ");
    try {
      String line = br.readLine();
      System.out.println("line : "+line);
    } catch (IOException e) {

      e.printStackTrace();
    }
  }
}
```

- scanner 객체를 사용해서 같은 작업을 했었다.(exception 처리 안해도 됨)

```
new Scanner(System.in)
// 역시 input 에서 불러옴
```

PrintStream

- 학습을 위해서 PrintStream 객체를 부모 type OutputStream 으로 받아보기

- OutputStream 도 1byte 처리 스트림이다.

```
package test.main;

import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;

public class MainClass04 {
  public static void main(String[] args) {
    PrintStream ps = System.out;
    OutputStream os = ps;
    try {
      os.write(97);
      os.write(98);
      os.write(99);
      os.write(44032); //OutputStream 은 1byte 처리 스트림이라 출력 안됨
      os.flush();// 방출
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

OutputStreamWriter

- 2byte 처리 스트림

```
package test.main;

import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintStream;

public class MainClass05 {
  public static void main(String[] args) {
    PrintStream ps = System.out;
    OutputStream os=ps;
    // 2byte 처리 스트림
    OutputStreamWriter osw = new OutputStreamWriter(os);
    try {
      osw.write(97);
      osw.write(44032);
      osw.flush();
    } catch (IOException e) {

      e.printStackTrace();
    }
  }
}
```

💡 "\r\n" 개행기호 = .newLine

💡 “\t” tap기호

Files

```
package test.main;

import java.io.File;

public class MainClass07 {
  public static void main(String[] args) {
    // c:/ 를 access할 수 있는 File 객체 생성
    File f = new File("c:/");
    String[] names = f.list();
    //배열에 있는 문자열 모두 출력하기
    for (int i = 0; i < names.length; i++) {
      System.out.println(names[i]);
    }
  }
}
```

```
package test.main;

import java.io.File;

public class MainClass08 {
  public static void main(String[] args) {
    File f = new File("c:/");
    // 파일객체 목록(File[]) 을 얻어내기
    File[] files = f.listFiles();
    // 반복문 사용해 File 객체를 하나씩 참조해서
    for (File tmp : files) {
      // 만일 해당 피일이 디렉토리라면
      if (tmp.isDirectory()) {
        // 대괄호를 디렉토리명 양쪽에 출력하기
        System.out.println("[ " + tmp.getName() + " ]");
      } else {
        System.out.println(tmp.getName());
      }
    }
  }
}
```

listFiles

- 객수만큼 파일 객체를 생성해서 참조값을 파일 배열에 담아서 리턴

isDirectory

- boolean type →true면 디렉토리 false면 아님

파일 생성, 삭제하기

```
package test.main;

import java.io.File;
import java.io.IOException;

public class MainClass09 {
  public static void main(String[] args) {
    //이미 존재하거나 혹은 만들 예정인 파일을 제어할 수 있는 File 객체 생성
    File f1 = new File("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\Dubu.txt");
    //만일 해당 피일이 존재하면
    if(f1.exists()) {
      //삭제
      f1.delete();
      System.out.println("Dubu.txt 파일을 삭제 했습니다.");
    }else {//존재하지 않으면
      try {
        //새 파일 만들기
        f1.createNewFile();
      } catch (IOException e) {
        e.printStackTrace();
      }
      System.out.println("Dubu.txt. 파일을 만들었습니다.");
    }
  }
}
```

폴더 생성, 삭제하기

mkdir() 을 사용

```
package test.main;

import java.io.File;
import java.io.IOException;
import java.util.Iterator;

public class MainClass10 {
  public static void main(String[] args) {
    File f1 = new File("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\folder1");

    for (int i = 0; i < 1000; i++) {
      f1= new File("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\folder"+(i+1));
      if(f1.exists()) {
        f1.delete();
      }else {
        f1.mkdir();
      }
    }

    /*
     *
     * 위의 코드를 참조해서 C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder 폴더 안에
     * 폴더명은 folder1 folder2 ... 1000개 만들기
     */
  }
}
```

내용이 있는 텍스트 파일 생성

```
package test.main;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class MainClass11 {
  public static void main(String[] args) {
    //문자열을 저장할 파일을 만들기 위한 File 객체
    File memoFile=new File("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\memo.txt");
    try {
      /*
       *  if(memoFile.exists() == false){ }
       *  if(!memoFile.exist()){ }
       *  위의 if 문은 동일한 if 문이다.
       */
      //파일이 존재하지 않으면
      if(!memoFile.exists()) {
        //파일을 만든다
        memoFile.createNewFile();
      }
      //파일에 문자열을 출력할 수  있는 FileWriter 객체
      FileWriter fw = new FileWriter(memoFile);
      fw.write("나도 우영우 보고싶다.");
      fw.write("\r\n");
      fw.write("우 투더 영 투더 우!");
      fw.write("\r\n");
      fw.write("동 투더 그 투더 롸미~");
      fw.write("\r\n");
      fw.flush();
      fw.close(); // 출력을 다 했으면 마무리를 한다.
      System.out.println("파일에 문자열을 저장했습니다.");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

덮어쓰기 하지 않고 계속 추가 하는 방법

```
FileWriter fw = new FileWriter(memoFile, true);
```

출력된 텍스트파일에서 읽어오는 방법

```
package test.main;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class MainClass12 {
  public static void main(String[] args) {
    // 문자열을 저장할 파일을 만들기 위한 File 객체
    File file = new File("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\memo.txt");
```

```
    try {
      FileReader fr = new FileReader(file);
      while (true) {
        // 문자 code 하나씩 읽어낸다
        int code = fr.read();
        // 더 이상 읽을 code 값이 없으면?
        if (code == -1) {
          break;// 반복문 탈출
        }
        // 코드값을 문자로 변환해서
        char ch = (char) code;
        // 개행기호 없이 출력하기
        System.out.print(ch);
      }

    } catch (IOException e) {
      e.printStackTrace();
    }

  }
}
```

line을 그대로 읽어오는 방법

- readLine() 개행기호를 읽어내지 않는다

```
package test.main;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class MainClass13 {

  public static void main(String[] args) {
    // 문자열을 저장할 파일을 만들기 위한 File 객체
    File file = new File("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\memo.txt");

    try {
      FileReader fr = new FileReader(file);
      BufferedReader br = new BufferedReader(fr);
      br.readLine();
      while(true) {
        //문자열 한줄 읽어내기
        String line = br.readLine();
        //더이상 읽을 데이터가 없으면 반복문 탈출
        if(line==null) {
          break;
        }
        System.out.println(line);
      }
    } catch (IOException e) {
      e.printStackTrace();
    }

  }
}
```

파일 복사

```
package test.main;

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

public class MainClass16 {
  public static void main(String[] args) {
    // 필요한 참조값을 담을 지역 변수를 미리 만든다.
    FileInputStream fis = null;
    FileOutputStream fos = null;
    try {
      fis = new FileInputStream("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\1.jpg");
      fos = new FileOutputStream("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\copied.jpg");

      while (true) {
        //1byte 읽어내기
        int data = fis.read();
        System.out.println(data);
        //읽어낸 1byte 출력하기
        fos.write(data);
        fos.flush();
        //다 읽어냈으면 탈출
        if (data == -1)break;
      }
    } catch (IOException e) {
      e.printStackTrace();
    } finally {
      // fos, fis 마무리 하기

        try {
          if(fis!=null)fis.close();
          if(fos!=null)fos.close();
        } catch (IOException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
    }
  }
}
```

1씩 로딩하면 느리까 1024씩 끌어오기

```
package test.main;

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

public class MainClass16 {
  public static void main(String[] args) {
    // 필요한 참조값을 담을 지역 변수를 미리 만든다.
    FileInputStream fis = null;
    FileOutputStream fos = null;
    try {
      // 1. jpg 에서 byte 를 읽어낼 객체
      fis = new FileInputStream("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\1.jpg");
      //copied.jpg 에 byte를 출력할 객체
      fos = new FileOutputStream("C:\\Users\\HNJN-PC\\Desktop\\school\\java_work\\myFolder\\copied.jpg");
      //byte 알갱이를 담을 방 1024개 짜리 byte[]객체 생성
      byte[] buffer=new byte[1024];
```

```
      while (true) {
        //byte[] 객체를 read() 메소드에 전달해서  byte를 읽어내고 몇 byte를 읽었는지 리턴 받는다
        int readByte=fis.read(buffer);
        System.out.println(readByte);
        //만일 더이상 읽을 byte 가 없다면
        if(readByte==-1)break;
        fos.write(buffer, 0, readByte);
      }
    } catch (IOException e) {
      e.printStackTrace();
    } finally {
      // fos, fis 마무리 하기

        try {
          if(fis!=null)fis.close();
          if(fos!=null)fos.close();
        } catch (IOException e) {
          e.printStackTrace();
        }
    }
  }
}
```