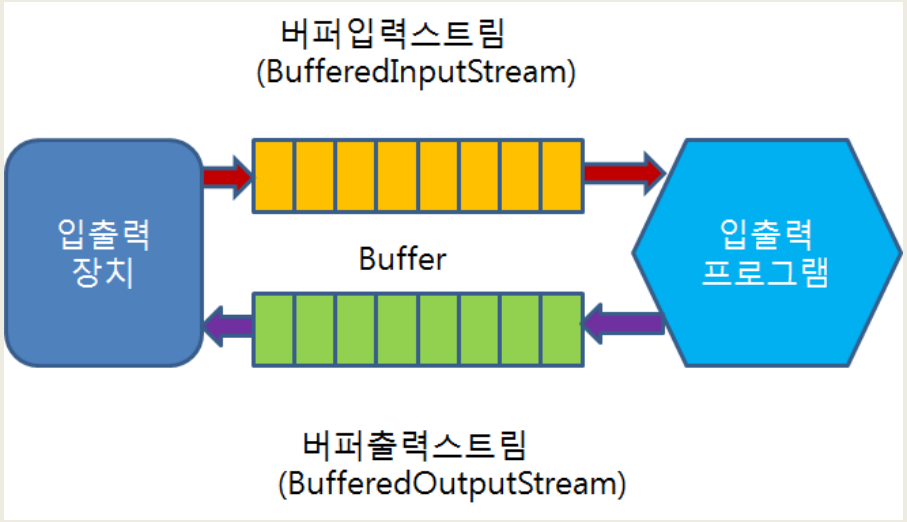
제 13 장

입출력 스트림과 통신
Part-2 : Buffered Stream

버퍼(Buffer) 스트림

- 버퍼 스트림
  - 버퍼를 가진 스트림
  - 입출력 데이터를 일시적으로 저장하는 버퍼를 이용하여 입출력 효율 개선
- 버퍼 입출력의 목적
  - 입출력 시 운영체제의 API 호출 횟수를 줄여 입출력 성능 개선
    - 출력시 여러 번 출력되는 데이터를 버퍼에 모아두고 한 번에 장치로 출력
    - 입력시 입력 데이터를 버퍼에 모아두고 한번에 프로그램에게 전달



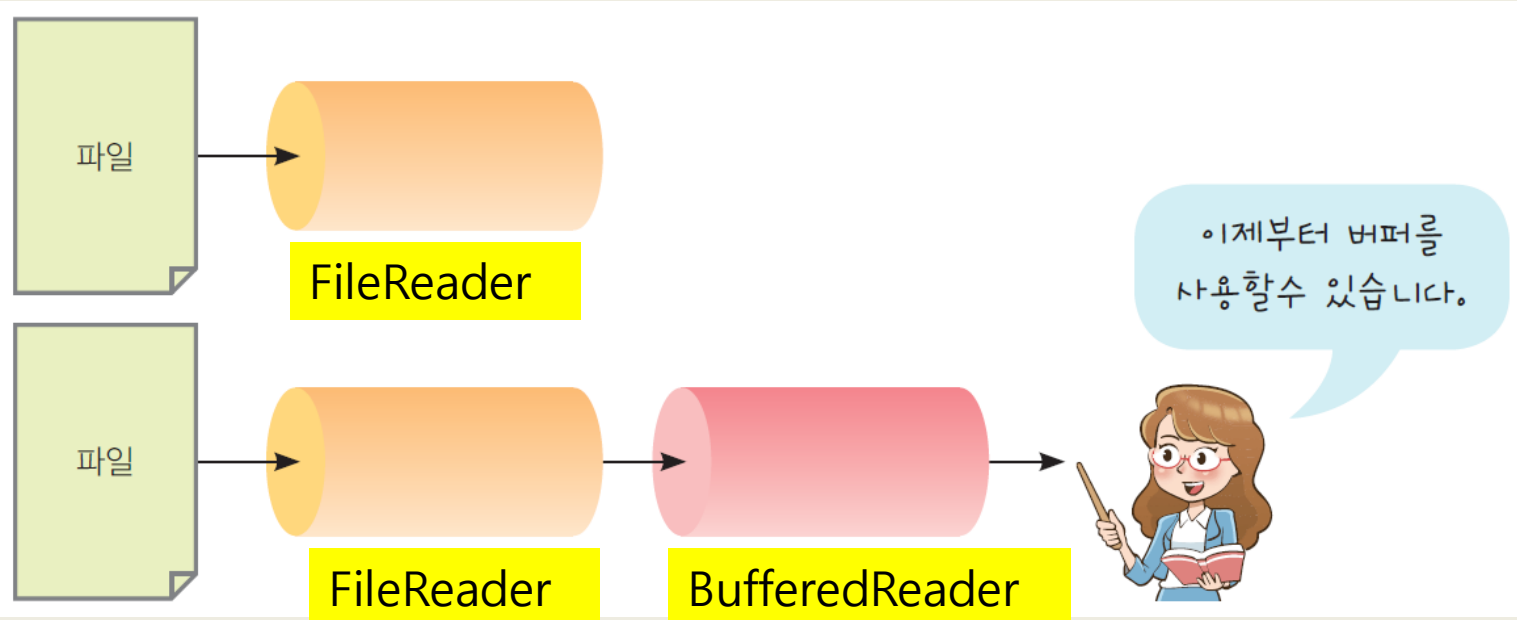
버퍼 스트림의 종류

- 바이트 버퍼 스트림
  - 바이트 단위의 바이너리 데이터를 처리하는 버퍼 스트림
  - BufferedInputStream과 BufferedOutputStream
- 문자 버퍼 스트림
  - 유니코드의 문자 데이터만 처리하는 버퍼 스트림
  - BufferedReader와 BufferedWriter

```
FileReader inputStream = new FileReader("input.txt");
FileWriter outputStream = new FileWriter("out put.txt");
```

buffer를 사용하지 않는 경우

```
FileReader inputStream = new BufferedReader(new FileReader("input.txt"));
FileWriter outputStream = new BufferedWriter(new FileWriter("out put.txt"));
```



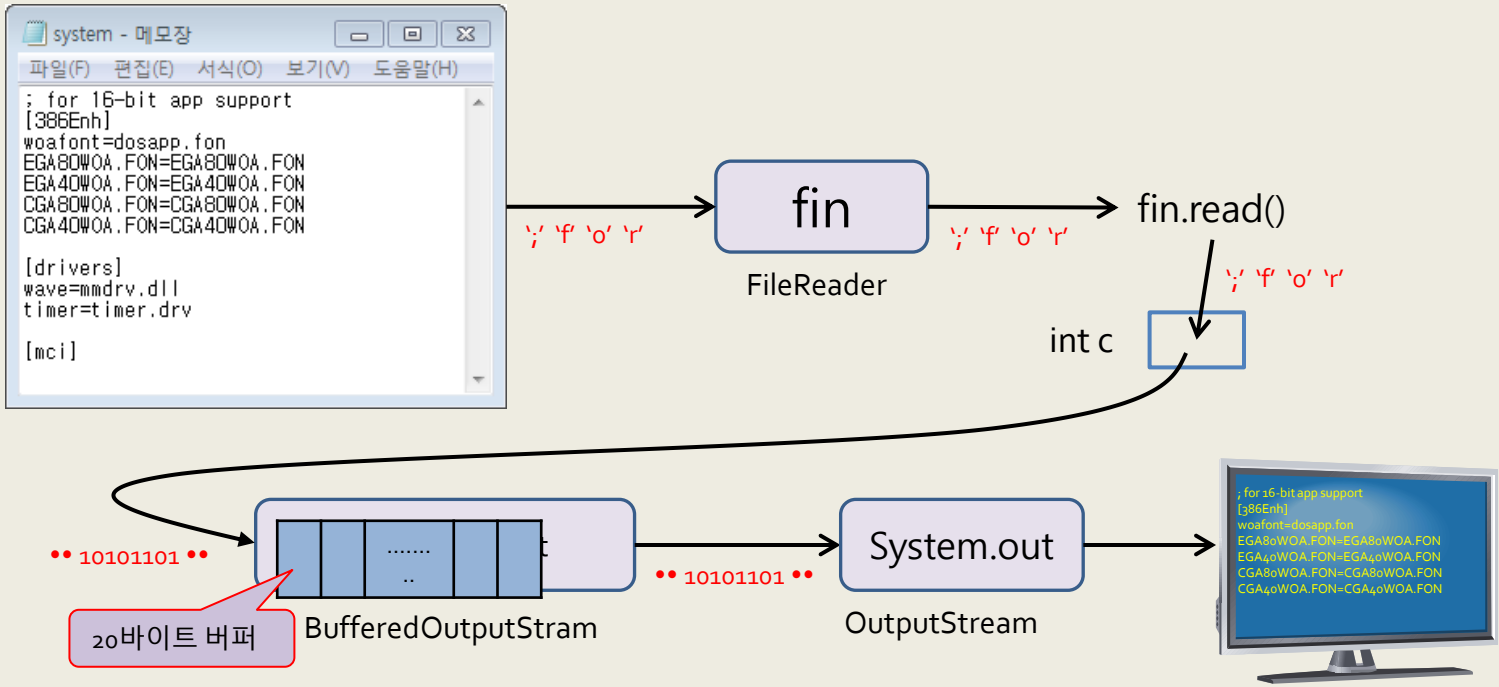
20바이트 버퍼를 가진 BufferedOutputStream

```
BufferedOutputStream bout = new BufferedOutputStream(System.out, 20);

FileReader fin = new FileReader("c:₩₩windows₩₩system.ini");

int c;
while ((c = fin.read()) != -1) {
    bout.write((char)c);
}
fin.close();
bout.close();
```

20바이트 크기의 버퍼 설정.
System.out 표준 스트림에 출력

파일 전체를 읽어 화면에 출력

스트림 닫음



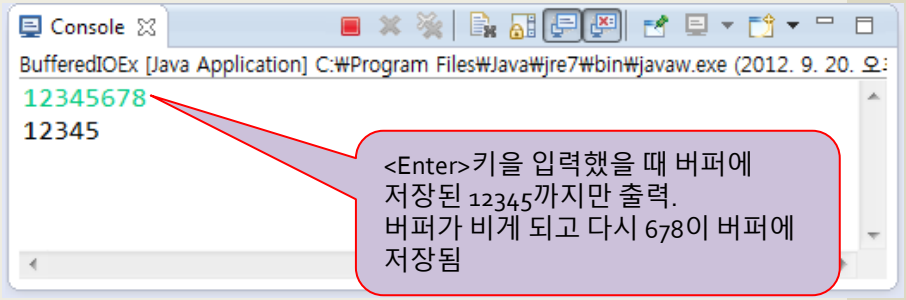
예제 : 버퍼 스트림을 이용하는 출력 예제

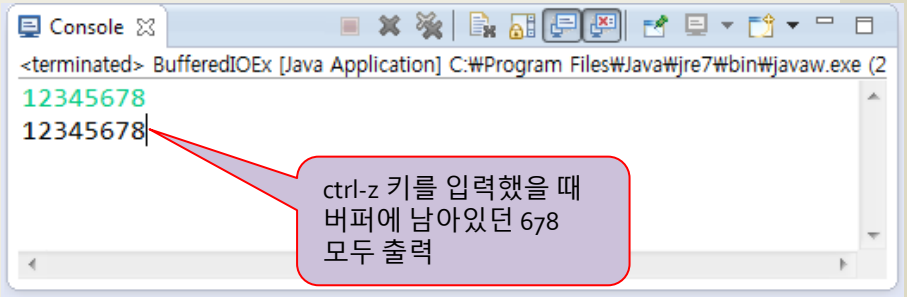
버퍼 크기를 5로하고, 표준 출력 스트림과 연결된 버퍼 출력 스트림을 생성하라.그리고 키보드에서 입력 받은 문자를 출력 스트림에 출력하고, 입력의 끝을 알리면(ctrl-z) 버퍼에 남아 있는 모든 문자를 출력하는 프로그램을 작성하라.

```
import java.io.*;

public class BufferedIOEx {
    public static void main(String[] args) {
        InputStreamReader in =
                new InputStreamReader(System.in);
        BufferedOutputStream out =
                new BufferedOutputStream(System.out, 5);
        try {
            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
            out.flush(); // 버퍼에 남아 있던 문자 출력
            if (in != null) {
                in.close();
                out.close();
            }
        } catch (IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```

ctrl-z가 입력될 때까지 반복

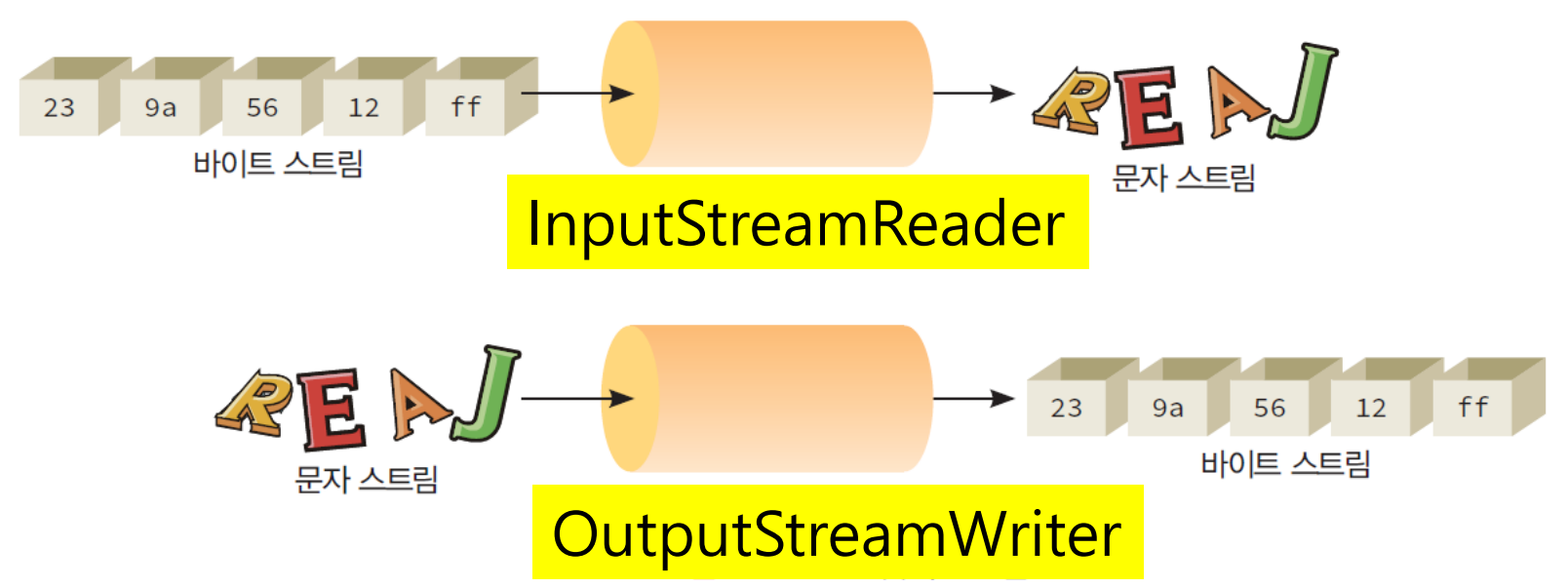
버퍼가 다 찰 때 문자가 화면에 출력





브릿지 스트림

- 바이트 스트림과 문자 스트림을 연결하는 범용 스트림
  - InputStreamReader
  - OutputStreamWriter



문자 Encoding

- 자바에서는 StandardCharsets 클래스 안에 각 엔코딩 방법이 StandardCharsets.UTF_8, StandardCharsets.UTF_16과 같이 상수로 정의되어 있다.
- String s = **new** String(100, StandardCharsets.UTF_8);
- 파일에서 읽을 때는 한글과 같은 특별 문자를 읽고자 하는 경우에는 FileReader를 사용하지 말고 InputStreamReader 클래스를 사용한다.

```
public class CharEncodingTest {
        public static void main(String[] args) throws IOException {
                File fileDir = new File("input.txt");
                BufferedReader in = new BufferedReader(new InputStreamReader( new
FileInputStream(fileDir), "UTF8"));
                String str;
                while ((str = in.readLine()) != null) {
                        System.out.println(str);
                }
        }
}
```

DataInputStream과 DataOutputStream

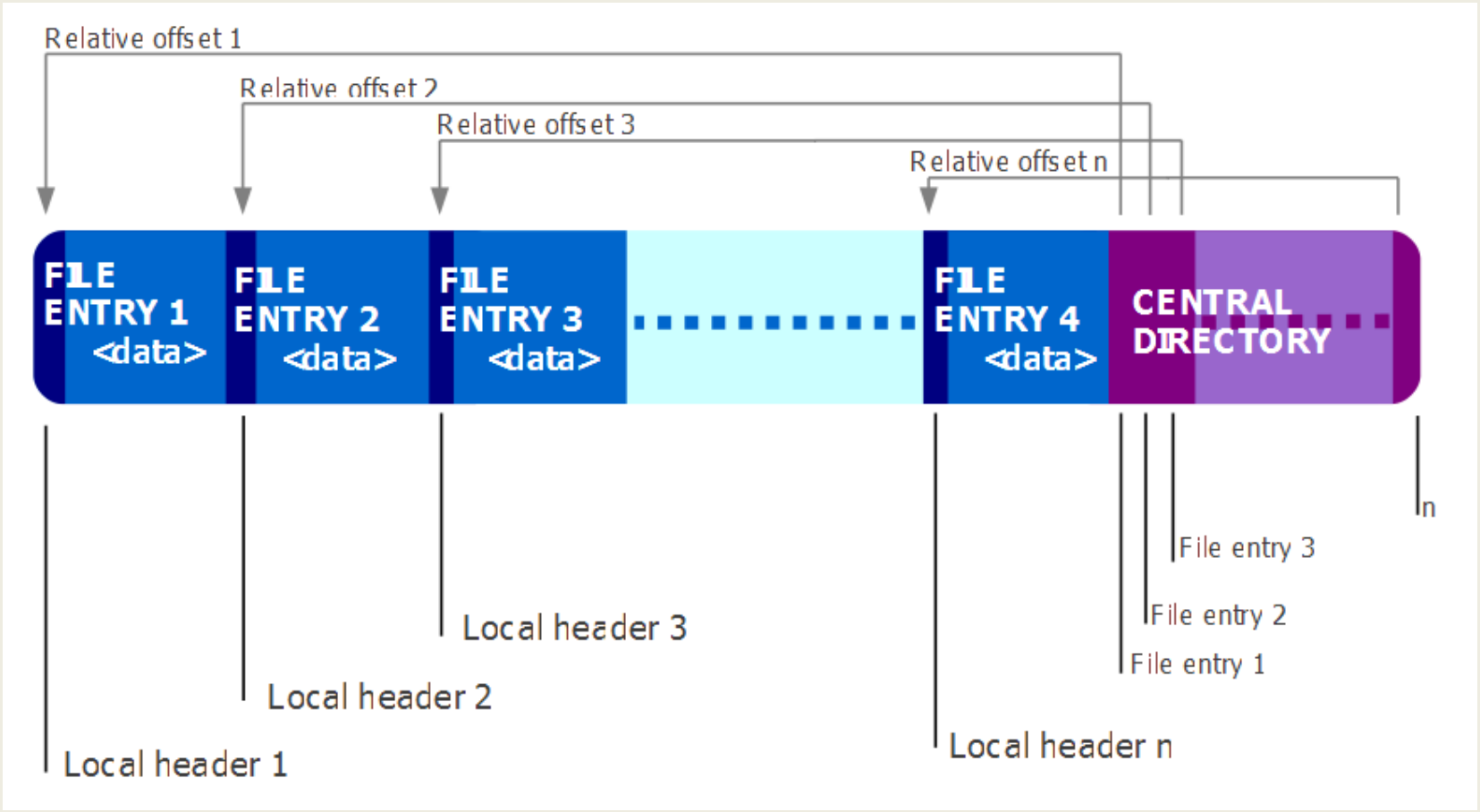
- DataInputStream 과 DataOutputStream 클래스는 기본 자료형 단위로 데이터를 읽고 쓸 수 있다.
- DataInputStream 클래스는 readByte(), readInt(), readDouble()과 같은 메소드들을 제공한다.
- DataOutputStream 클래스는 writeByte(byte v), writeInt(int v), writeDouble(double v)와 같은 메소드들을 제공한다.

```
import java.io.*;
public class DataStreamTest {
	public static void main(String[] args) throws IOException {
		DataInputStream in = null;
		DataOutputStream out = null;
		try {
			int c;
			 out = new DataOutputStream(new BufferedOutputStream(
					new FileOutputStream("data.bin")));
			out.writeDouble(3.14);
			out.writeInt(100);
			out.writeUTF("자신의 생각을 바꾸지 못하는 사람은 결코 현실을 바꿀 수 없다.");
			 out.flush();
			in = new DataInputStream(new BufferedInputStream(
					new FileInputStream("data.bin")));
			  System.out.println(in.readDouble());
			System.out.println(in.readInt());
			System.out.println(in.readUTF());

		} finally {
			if (in != null) {
				in.close();
			}
			if (out != null) {
				out.close();
			}
		}
	} }
```

```
3.14
100
자신의 생각을 바꾸지 못하는 사람은 결코 현실을 바꿀 수 없다.
```

압축파일 풀기

- 자바에서는 ZipInputStream을 이용하여서 ZIP 파일을 읽을 수 있다.



```
public class ZipTest {
        public static void main(String[] args) throws IOException {
                FileInputStream fin = new FileInputStream("test.zip");
                ZipInputStream zin = new ZipInputStream(fin);
                ZipEntry entry = null;
                while ((entry = zin.getNextEntry()) != null) {
                        System.out.println("압축 해제: " + entry.getName());
                        FileOutputStream fout = new
FileOutputStream(entry.getName());
                        for (int c = zin.read(); c != -1; c = zin.read()) {
                                fout.write(c);
                        }
                        zin.closeEntry();
                        fout.close();
                }
                zin.close();
        }
}
```

압축 해제: eclipse.ini

ObjectInputStream과 ObjectOutputStream

- **직렬화(serialization):**
  - 객체가 가진 데이터들을 순차적인 데이터로 변환하는 것

객체 직렬화의 개념

```
import java.io.*;
import java.util.Date;
public class ObjectStreamTest {
	public static void main(String[] args) throws IOException {
		ObjectInputStream in = null;
		ObjectOutputStream out = null;
		try {
			int c;
			out = new ObjectOutputStream(new FileOutputStream("object.dat"));
			out.writeObject(new Date());

			out.flush();
			in = new ObjectInputStream(new FileInputStream("object.dat"));
			Date d = (Date) in.readObject();
			System.out.println(d);
		} catch (ClassNotFoundException e) {

		  } finally {
			if (in != null) {
				in.close();
			}
			if (out != null) {
				out.close();
			}
		}
	}
}
```

```
Mon Jul 13 13:39:47 KST 2015
```