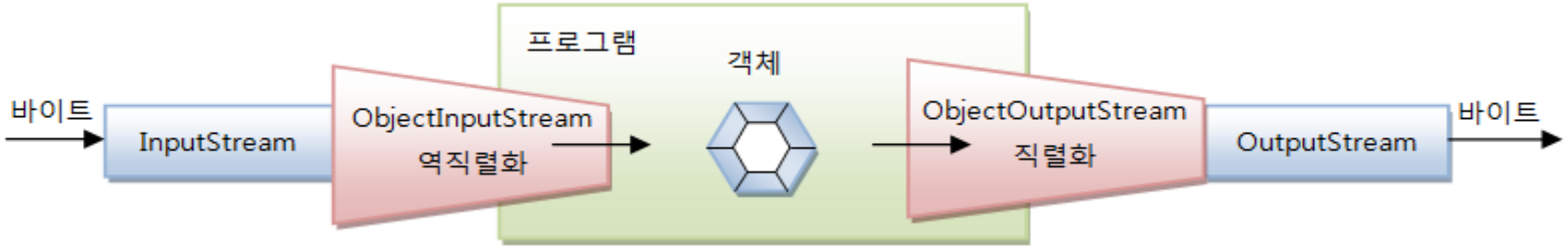
객체 직렬화

객체 직렬화

❖ 객체 입출력 보조 스트림

- 객체를 파일 또는 네트워크로 입출력할 수 있는 기능 제공
- 객체 직렬화
  - 객체는 문자가 아니므로 바이트 기반 스트림으로 데이터 변경 필요

- ObjectInputStream, ObjectOutputStream




```
ObjectInputStream ois = new ObjectInputStream(바이트입력스트림);
ObjectOutputStream oos = new ObjectOutputStream(바이트출력스트림);

oos.writeObject(객체);

객체타입 변수 = (객체타입) ois.readObject();
```

객체 직렬화

❖ 다양한 객체를 쓰고 읽기: ObjectInputOutputStreamExample.java

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class ObjectInputOutputStreamExample {
    public static void main(String[] args) throws Exception {
        FileOutputStream fos = new FileOutputStream("C:/Temp/Object.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fos);

        oos.writeObject(new Integer(10));
        oos.writeObject(new Double(3.14));
        oos.writeObject(new int[] { 1, 2, 3 });
        oos.writeObject(new String("홍길동"));

        oos.flush(); oos.close(); fos.close();
```

❖ 다양한 객체를 쓰고 읽기: ObjectInputOutputStreamExample.java

```
        FileInputStream fis = new FileInputStream("C:/Temp/Object.dat");
        ObjectInputStream ois = new ObjectInputStream(fis);

        Integer obj1 = (Integer) ois.readObject();
        Double obj2 = (Double) ois.readObject();
        int[] obj3 = (int[]) ois.readObject();
        String obj4 = (String) ois.readObject();

        ois.close(); fis.close();

        System.out.println(obj1);
        System.out.println(obj2);
        System.out.println(obj3[0] + "," + obj3[1] + "," + obj3[2]);
        System.out.println(obj4);
    }
}
```

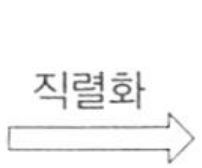
객체 직렬화

❖ 객체 입출력 보조 스트림

- 직렬화가 가능한 클래스(Serializable)
  - 자바에서는 Serializable 인터페이스를 구현한 클래스만 직렬화 할 수 있도록 제한, transient 필드는 제외
- 객체 직렬화 할 때 private 필드 포함한 모든 필드를 바이트로 변환 가능

```
public class XXX implements Serializable { }
```

```
public class XXX implements Serializable {
   public int field1;
   protected int field2;
   int field3;
   private int field4;
   public static int field5;
   transient int field6;
}
```



일렬로 늘어선 바이트 데이터

field1	field2	field3	field4

static 또는 transient 키워드가 붙은 필드는 직렬화에서 제외

❖ 직렬화가 가능한 클래스: ClassA.java

```
import java.io.Serializable;

public class ClassA implements Serializable {
    int field1;
    ClassB field2 = new ClassB();
    static int field3;
    transient int field4;
}
```

❖ 직렬화가 가능한 클래스: ClassB.java

```
import java.io.Serializable;

public class ClassB implements Serializable {
    int field1;
}
```

❖ 직렬화해서 파일에 저장: SerializableWriter.java

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class SerializableWriter {
    public static void main(String[] args) throws Exception {
        FileOutputStream fos = new FileOutputStream("C:/Temp/Object.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        ClassA classA = new ClassA();
        classA.field1 = 1;
        classA.field2.field1 = 2;
        classA.field3 = 3;
        classA.field4 = 4;
        oos.writeObject(classA);
        oos.flush(); oos.close(); fos.close();
    }
}
```

❖ 역직렬화해서 복원된 필드 조사: SerializableReader.java

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class SerializableReader {
    public static void main(String[] args) throws Exception {
        FileInputStream fis = new FileInputStream("C:/Temp/Object.dat");
        ObjectInputStream ois = new ObjectInputStream(fis);
        ClassA v = (ClassA) ois.readObject();
        System.out.println("field1: " + v.field1);
        System.out.println("field2.field1: " + v.field2.field1);
        System.out.println("field3: " + v.field3);
        System.out.println("field4: " + v.field4);
    }
}
```

객체 직렬화

❖ serialVersionUID 필드

- 직렬화된 객체를 역직렬화 할 때는 직렬화 했을 때와 같은 클래스 사용
- 클래스의 이름이 같더라도 클래스의 내용이 변경된 경우 역직렬화 실패

```
java.io.InvalidClassException: XXX; local class incompatible: stream classdesc
serialVersionUID = -9130799490637378756, local class serialVersionUID = -1174725809595957294
```

- serialVersionUID
  - 같은 클래스임을 알려주는 식별자 역할
  - Serializable 인터페이스 구현
  - 컴파일 시 자동적으로 serialVersionUID 정적 필드 추가
  - 재컴파일하면 serialVersionUID의 값 변경

- 불가피한 수정 있을 경우 명시적으로 serialVersionUID 선언

```
static final long serialVersionUID = 정수값;
```

객체 직렬화

❖ 직렬화가 가능한 클래스: ClassC.java

```
import java.io.Serializable;

public class ClassC implements Serializable {
   int field1;
}
```

❖ 객체 직렬화: SerialVersionUIDExample1.java

```
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;

public class SerialVersionUIDExample1 {
   public static void main(String[] args) throws Exception {
      FileOutputStream fos = new FileOutputStream("C:/Temp/Object.dat");
      ObjectOutputStream oos = new ObjectOutputStream(fos);
      ClassC classC = new ClassC();
      classC.field1 = 1;
      oos.writeObject(classC);
      oos.flush(); oos.close();  fos.close();
   }
}
```

❖ 객체 역직렬화: SerialVersionUIDExample2.java

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class SerialVersionUIDExample2 {
    public static void main(String[] args) throws Exception {
        FileInputStream fis = new FileInputStream("C:/Temp/Object.dat");
        ObjectInputStream ois = new ObjectInputStream(fis);
        ClassC classC = (ClassC) ois.readObject();
        System.out.println("field1: " + classC.field1);
    }
}
```

❖ 역직렬화할 클래스: ClassC.java

```
import java.io.Serializable;

public class ClassC implements Serializable {
    int field1;
    int field2;

}
```

- SerialVersionUIDExample2 실행시 예외 발생
- serialVersionUID 명시적 선언 필요

```
public class XXX implements Serializable {
    static final long serialVersionUID = 정수값;
    ...
}
```

객체 직렬화

❖ serialVersionUID 생성

```
c:\Temp>dir
 C 드라이브의 볼륨: 운영 디스크
 볼륨 일련 번호: A449-4261

 c:\Temp 디렉터리

2009-12-21  오후 09:32    <DIR>          .
2009-12-21  오후 09:32    <DIR>          ..
2009-12-21  오후 09:31               208 XXX.class
2009-12-21  오후 09:31                67 XXX.java
               2개 파일                 275 바이트
               2개 디렉터리  84,131,876,864 바이트 남음

c:\Temp>serialver XXX
XXX:    static final long serialVersionUID = 5835806237290266999L;
```

```
public class XXX implements Serializable  {
 static final long serialVersionUID = 5835806237290266999L;
}
```

복사해서 클래스의 필드로 만든다.

❖ writeObject()와 readObject() 메소드

- writeObject(ObjectOutputStream out)
  - 직렬화 직전 자동 호출
  - 추가 직렬화할 내용 작성 가능

- readObject(ObjectInputStream in)
  - 역직렬화 직전 자동 호출
  - 추가 역직렬화 내용 작성 가능

- 추가 직렬화 및 역직렬화 필요한 경우
  - 부모 클래스가 Serializable 구현하지 않고, 자식 클래스가 Serializable 구현한 경우
  - 부모 필드는 직렬화에서 제외
    - writeObject() 에서 부모 필드 직렬화 필요
    - readObject()에서 부모 필드 역직렬화 필요
    - 부모 클래스가 Serializable 구현하도록 하는 게 제일 쉬움

객체 직렬화

❖ writeObject()와 readObject() 메소드

```
private void writeObject(ObjectOutputStream out) throws IOException {
    out.writeXXX(부모필드);
          :                                  } 부모 객체의 필드값을 출력함
    out.defaultWriteObject(); ·····  자식 객체의 필드값을 직렬화
}

private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
    부모필드 = in.readXXX();
          :                                  } 부모 객체의 필드값을 읽어옴
    in.defaultReadObject(); ·····  자식 객체의 필드값을 역직렬화
}
```

❖ Serializable을 구현하지 않은 부모 클래스: Parent.java

```
public class Parent {
    public String field1;
}
```

❖ 직렬화되지 않는 부모 클래스의 필드 처리: Child.java

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class Child extends Parent implements Serializable {
    public String field2;

    private void writeObject(ObjectOutputStream out) throws IOException {
        out.writeUTF(field1);
        out.defaultWriteObject();
    }

    private void readObject(ObjectInputStream in) throws IOException,
                                                         ClassNotFoundException {
        field1 = in.readUTF();
        in.defaultReadObject();
    }
}
```

❖ 직렬화 및 역직렬화: NonSerializableParentExample.java

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class NonSerializableParentExample {
    public static void main(String[] args) throws Exception {
        FileOutputStream fos = new FileOutputStream("C:/Temp/Object.dat");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        Child child = new Child();
        child.field1 = "홍길동";
        child.field2 = "홍삼원";
        oos.writeObject(child);
        oos.flush(); oos.close(); fos.close();

        FileInputStream fis = new FileInputStream("C:/Temp/Object.dat");
        ObjectInputStream ois = new ObjectInputStream(fis);
        Child v = (Child) ois.readObject();
        System.out.println("field1: " + v.field1);
        System.out.println("field2: " + v.field2);
        ois.close(); fis.close();
    }
}
```