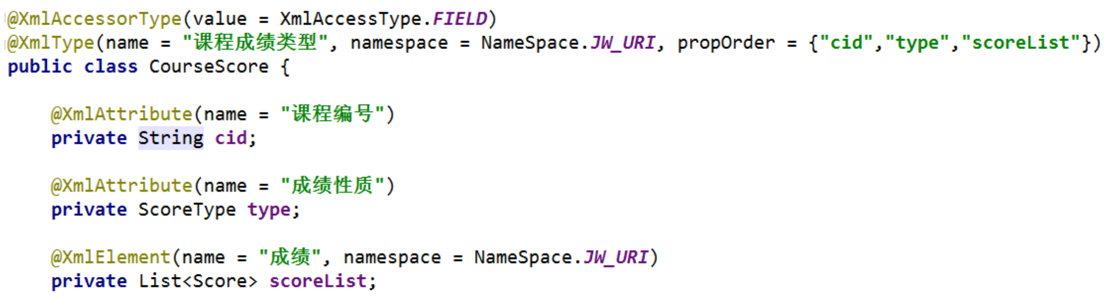
1、生成 model

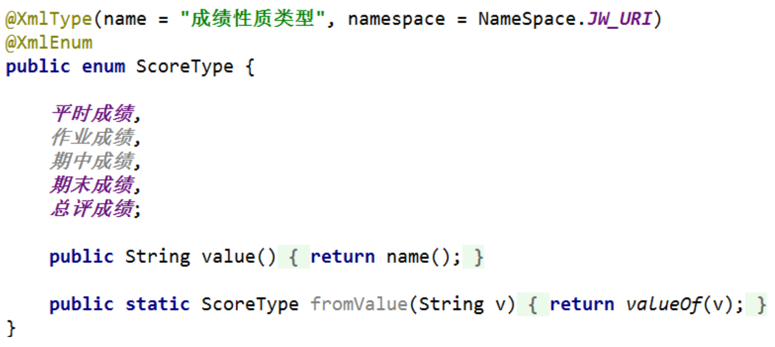
(1)　生成 java 类

使用 jaxb 可以根据 xsd 逆向生成 Java 代码，但考虑到中文编程不太方便，修改为英文类和属性，并且加上了 name 属性。
转换命令为 xjc -p edu.nju.soa.model *.xsd -d src



```
@XmlAccessorType(value = XmlAccessType.FIELD)
@XmlType(name = "课程成绩类型", namespace = NameSpace.JW_URI, propOrder = {"cid","type","scoreList"})
public class CourseScore {

    @XmlAttribute(name = "课程编号")
    private String cid;

    @XmlAttribute(name = "成绩性质")
    private ScoreType type;

    @XmlElement(name = "成绩", namespace = NameSpace.JW_URI)
    private List<Score> scoreList;
```

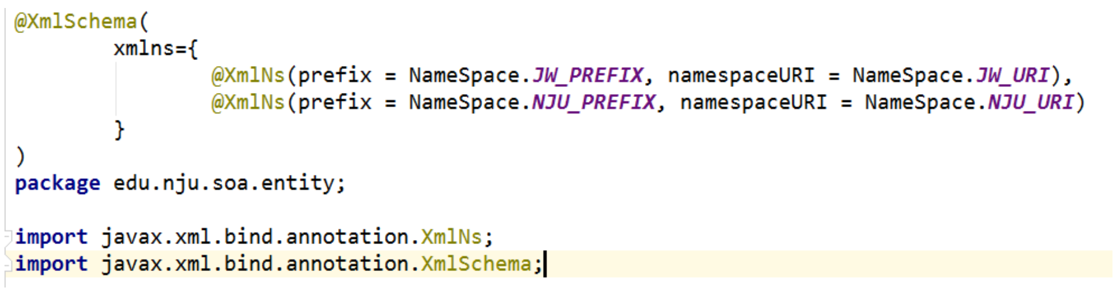
完整保留了 schema 定义的所有类型



```
@XmlType(name = "成绩性质类型", namespace = NameSpace.JW_URI)
@XmlEnum
public enum ScoreType {

    平时成绩,
    作业成绩,
    期中成绩,
    期末成绩,
    总评成绩;

    public String value() { return name(); }

    public static ScoreType fromValue(String v) { return valueOf(v); }
}
```

(2)　关于命名空间

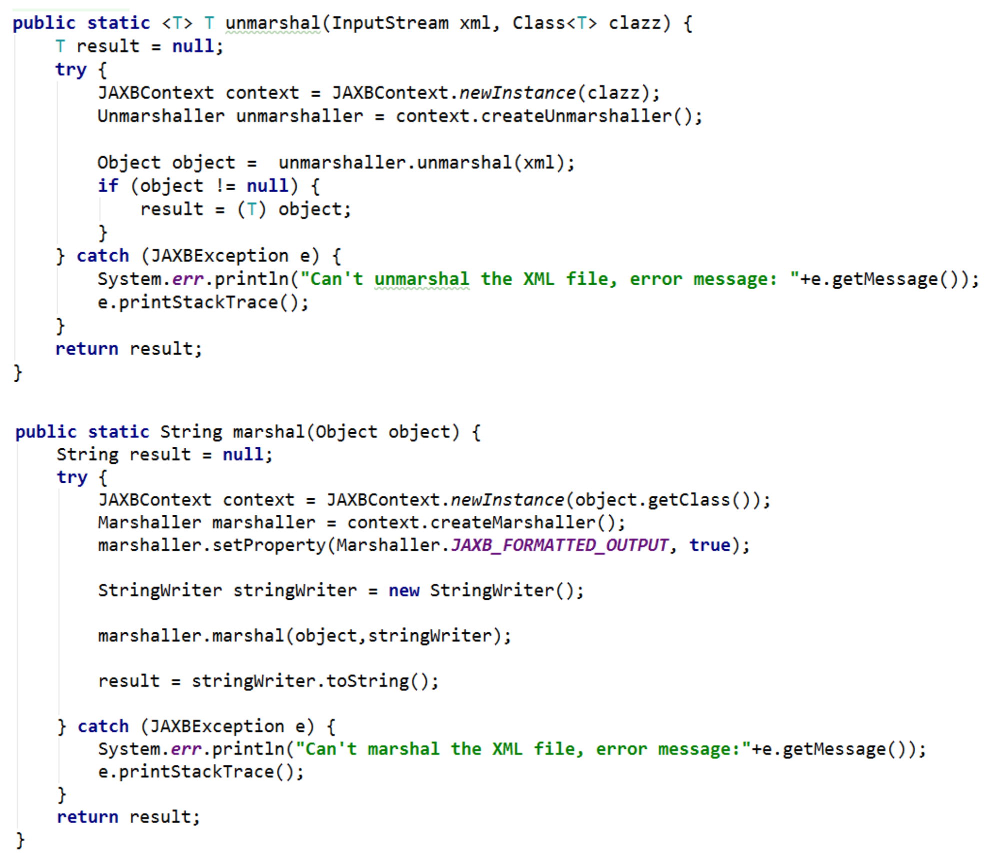
在 package-info.java 中添加：



```
@XmlSchema(
        xmlns={
                @XmlNs(prefix = NameSpace.JW_PREFIX, namespaceURI = NameSpace.JW_URI),
                @XmlNs(prefix = NameSpace.NJU_PREFIX, namespaceURI = NameSpace.NJU_URI)
        }
)
package edu.nju.soa.entity;

import javax.xml.bind.annotation.XmlNs;
import javax.xml.bind.annotation.XmlSchema;
```

2、marshal 和 unmarshal 方法

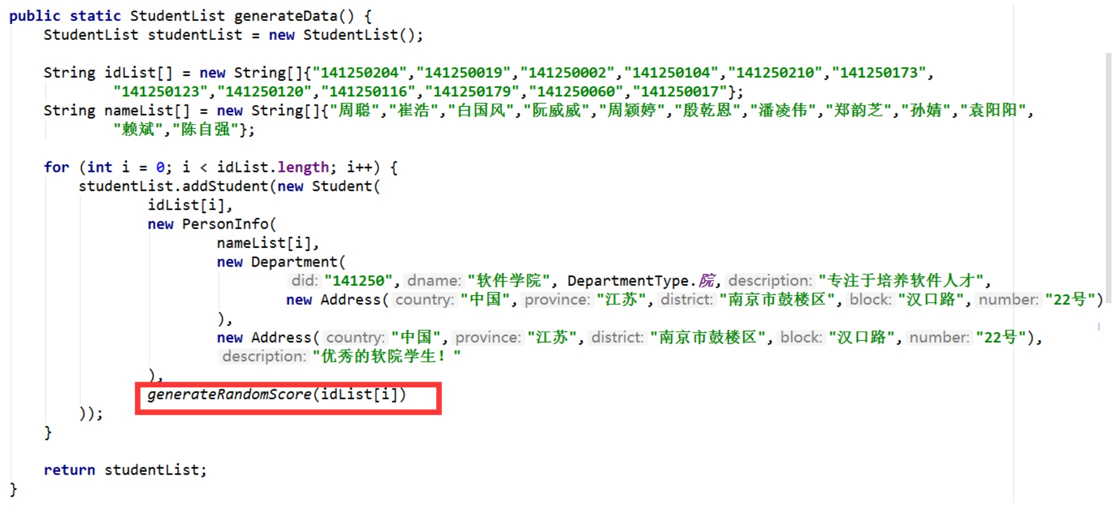
编写基于泛型的 marshall 和 unmarshall 工具类：



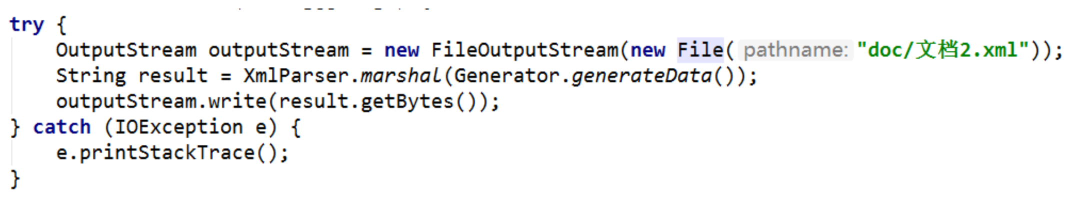
```java
public static <T> T unmarshal(InputStream xml, Class<T> clazz) {
    T result = null;
    try {
        JAXBContext context = JAXBContext.newInstance(clazz);
        Unmarshaller unmarshaller = context.createUnmarshaller();

        Object object =  unmarshaller.unmarshal(xml);
        if (object != null) {
            result = (T) object;
        }
    } catch (JAXBException e) {
        System.err.println("Can't unmarshal the XML file, error message: "+e.getMessage());
        e.printStackTrace();
    }
    return result;
}

public static String marshal(Object object) {
    String result = null;
    try {
        JAXBContext context = JAXBContext.newInstance(object.getClass());
        Marshaller marshaller = context.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

        StringWriter stringWriter = new StringWriter();

        marshaller.marshal(object,stringWriter);

        result = stringWriter.toString();

    } catch (JAXBException e) {
        System.err.println("Can't marshal the XML file, error message:"+e.getMessage());
        e.printStackTrace();
    }
    return result;
}
```

3、生成文档 2.xml

生成学生列表，使用定义的学生信息以及随机生成的课程成绩，随机生成课程成绩时会保证每一名学生都有一门低于 60 分的成绩。



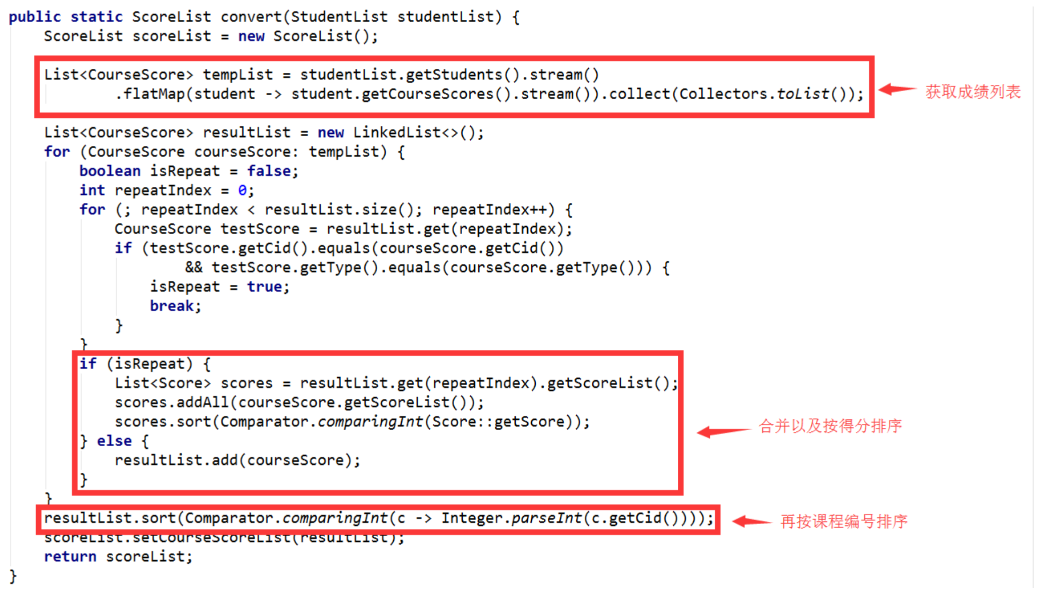
```java
public static StudentList generateData() {
    StudentList studentList = new StudentList();

    String idList[] = new String[]{"141250204","141250019","141250002","141250104","141250210","141250173",
            "141250123","141250120","141250116","141250179","141250060","141250017"};
    String nameList[] = new String[]{"周聪","崔浩","白国风","阮威威","周颖婷","殷乾恩","潘凌伟","郑韵芝","孙婧","袁阳阳",
            "赖斌","陈自强"};

    for (int i = 0; i < idList.length; i++) {
        studentList.addStudent(new Student(
                idList[i],
                new PersonInfo(
                        nameList[i],
                        new Department(
                                did: "141250", dname: "软件学院", DepartmentType.院, description: "专注于培养软件人才",
                                new Address( country: "中国", province: "江苏", district: "南京市鼓楼区", block: "汉口路", number: "22号")
                        ),
                        new Address( country: "中国", province: "江苏", district: "南京市鼓楼区", block: "汉口路", number: "22号"),
                        description: "优秀的软院学生！"
                ),
                generateRandomScore(idList[i])
        ));
    }

    return studentList;
}
```



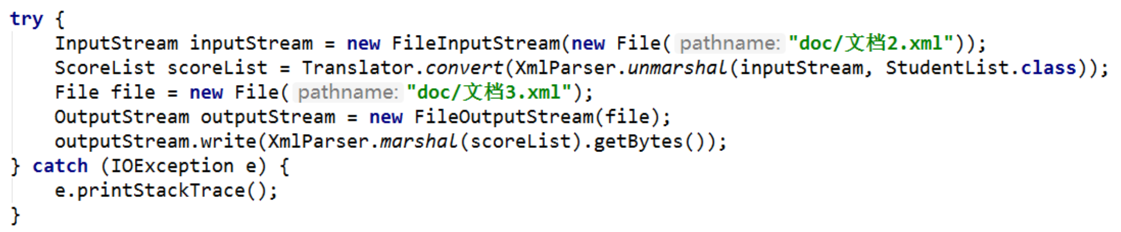
```java
try {
    OutputStream outputStream = new FileOutputStream(new File( pathname: "doc/文档2.xml"));
    String result = XmlParser.marshal(Generator.generateData());
    outputStream.write(result.getBytes());
} catch (IOException e) {
    e.printStackTrace();
}
```

4、生成文档 3.xml

这一步的重点在于获取每位学生的成绩之后进行合并，使得同一门课程同种类型的成绩合并到一个根节点下，并对每门课程的成绩进行排序（根据得分），以及对整个课程成绩进行排序（根据课程编号）。



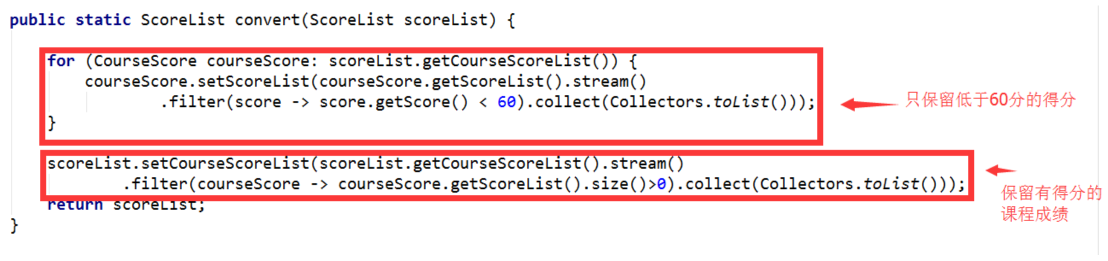
```java
public static ScoreList convert(StudentList studentList) {
    ScoreList scoreList = new ScoreList();

    List<CourseScore> tempList = studentList.getStudents().stream()
            .flatMap(student -> student.getCourseScores().stream()).collect(Collectors.toList());

    List<CourseScore> resultList = new LinkedList<>();
    for (CourseScore courseScore: tempList) {
        boolean isRepeat = false;
        int repeatIndex = 0;
        for (; repeatIndex < resultList.size(); repeatIndex++) {
            CourseScore testScore = resultList.get(repeatIndex);
            if (testScore.getCid().equals(courseScore.getCid())
                    && testScore.getType().equals(courseScore.getType())) {
                isRepeat = true;
                break;
            }
        }
        if (isRepeat) {
            List<Score> scores = resultList.get(repeatIndex).getScoreList();
            scores.addAll(courseScore.getScoreList());
            scores.sort(Comparator.comparingInt(Score::getScore));
        } else {
            resultList.add(courseScore);
        }
    }
    resultList.sort(Comparator.comparingInt(c -> Integer.parseInt(c.getCid())));
    scoreList.setCourseScoreList(resultList);
    return scoreList;
}
```



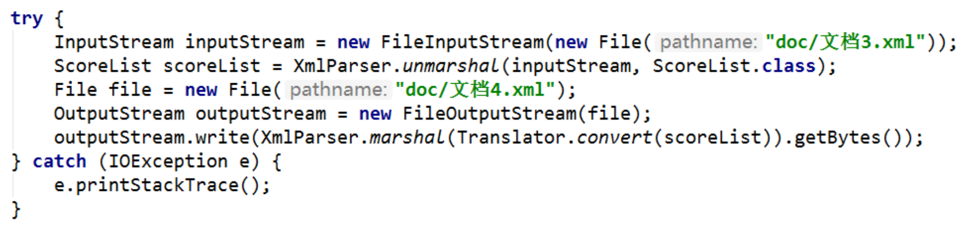
```java
try {
    InputStream inputStream = new FileInputStream(new File( pathname: "doc/文档2.xml"));
    ScoreList scoreList = Translator.convert(XmlParser.unmarshal(inputStream, StudentList.class));
    File file = new File( pathname: "doc/文档3.xml");
    OutputStream outputStream = new FileOutputStream(file);
    outputStream.write(XmlParser.marshal(scoreList).getBytes());
} catch (IOException e) {
    e.printStackTrace();
}
```

5、生成文档 4.xml

分为两步进行：1、删除低于 60 分的得分；2、如果一门课程成绩里没有得分，则删除该课程成绩列表



```java
public static ScoreList convert(ScoreList scoreList) {

    for (CourseScore courseScore: scoreList.getCourseScoreList()) {
        courseScore.setScoreList(courseScore.getScoreList().stream()
                .filter(score -> score.getScore() < 60).collect(Collectors.toList()));
    }

    scoreList.setCourseScoreList(scoreList.getCourseScoreList().stream()
            .filter(courseScore -> courseScore.getScoreList().size()>0).collect(Collectors.toList()));
    return scoreList;
}
```



```java
try {
    InputStream inputStream = new FileInputStream(new File( pathname: "doc/文档3.xml"));
    ScoreList scoreList = XmlParser.unmarshal(inputStream, ScoreList.class);
    File file = new File( pathname: "doc/文档4.xml");
    OutputStream outputStream = new FileOutputStream(file);
    outputStream.write(XmlParser.marshal(Translator.convert(scoreList)).getBytes());
} catch (IOException e) {
    e.printStackTrace();
}
```