A university manages its students using a program that has a class `Student` with subclasses `FirstYear`, `SecondYear`, and `ThirdYear` for year-specific state and behaviour. The program has a `List<Student>` that contains all students.

(a) Should `Student` be a class, an abstract class or an interface? Explain your answer.  

(b) Write a `Comparator` that can be used to sort the `List<Student>` by year group and then by name, both ascending, and show how it would be used. You should assume the existence of a `String getName()` method within `Student`. 

(c) At the end of each year, the third year students graduate and must be removed. This is done by passing the list to the following method:

```java
void removeThirdYears(List<Student> students) {
    for (Student student : students) {
        try {
            ThirdYear thirdyear = (ThirdYear) student;
            students.remove(thirdyear);
        }
        catch(ClassCastException cce) { }
    }
}
```

(i) What will happen when the call to `remove()` is made? Explain why and fix the code. 

(ii) Comment on this use of exceptions and propose an alternative that does not rely on them. 

(d) Also at the end of the academic year, the first and second year students move up a year.

(i) Explain why this class design makes this problematic. 

(ii) Propose an alternative design and explain in detail how it addresses the problems you identified in (d)(i).