手写版

```
public int getScore() {
    int score = 0;

    if (levelOne.goalReached()) {
        score = levelOne.getPoints();
        if (levelTwo.goalReached()) {
            score += levelTwo.getPoints();
            if (levelThree.goalReached()) {
                score += levelThree.getPoints();
            }
        }
    }

    if (isBonus()) {
        score *= 3;
    }

    return score;
}

//---
```

```
public int playManyTimes(int num) {
    int max = 0;
    for (int i = 0; i < num; i++) {
        play();
        int score = getScore();
        if (score > max) {
            max = score;
        }
    }
    return max;
}
```

```
public class Textbook extends Book {
    private int edition;

    public Textbook(String tbTitle, double tbPrice, int
tbEdition) {
        super(tbTitle, tbPrice);
        edition = tbEdition;
    }
    public int getEdition() {
        return edition;
    }

    public boolean canSubstituteFor(Textbook other) {
        return other.getTitle().equals(getTitle()) &&
                edition >= other.getEdition();
    }

    public String getBookInfo() {
        return super.getBookInfo() + "-" + edition;
    }
}
```

```
public double getAverageRating() {
    int sum = 0;
    for (Review r : allReviews) {
        sum += r.getRating();
    }
    return (double) sum / allReviews.length;
}
```

```
//---

public ArrayList<String> collectComments() {
    ArrayList<String>      commentList      =      new
ArrayList<String>();
    for (int i = 0; i < allReviews.length; i++) {
        String comment = allReviews[i].getComment();
        if (comment.indexOf("!") >= 0) {
            String                last                =
comment.substring(comment.length() - 1);
            if (!last.equals("!") && !last.equals(".")) {
                comment += ".";
            }
            commentList.add(i + "-" + comment);
        }
    }
    return commentList;
}
```

```
public void repopulate() {
    for (int row = 0; row < grid.length; row++) {
        for (int col = 0; col < grid[row].length; col++) {
            int rval = (int) (Math.random() * MAX) +
1;
            while (rval / 10 != 0 || rval / 100 == 0) {
                rval = (int) (Math.random() * MAX) +
1;
            }
            grid[row][col] = rval;
        }
    }
}

//---

public int countIncreasingCols() {
    int count = 0;
    for (int col = 0; col < grid[0].length; col++) {
        boolean ordered = true;
        for (int row = 1; row < grid.length; row++) {
            if (grid[row][col] < grid[row - 1][col]) {
                ordered = false;
                break;
            }
        }
```

```
            if (ordered) {
                count++;
            }
        }
        return count;
    }
```