function map<T, U> (items T): U[] {

let result: U[] = [];

for (const item of items) {

result.push(item.func4map());

}

return result;

}

abstract class Student {

float func4map() {

return getScore();

}

abstract float getScore();

}

class Boy extends Student {

float runScore;

@Override

float getScore() {

return runScore;

}

}

class Girl extends Student {

float walkScore;

@Override

float getScore() {

return walkScore;

}

}

let students: Student[] = [s1, s2, …, s10]

let score: float[] = map(students);

总的来说就是用到了OOP，那么可以规定所有能作为map参数的T类型都必须提供一个func4map()接口，这样就不需要提供函数作为map参数，在map中直接对items的每个项目调用它自己的func4map。对于统计成绩这项工作，在基类的func4map中调用getScore()，再在派生类中给getScore()提供不同的实现。