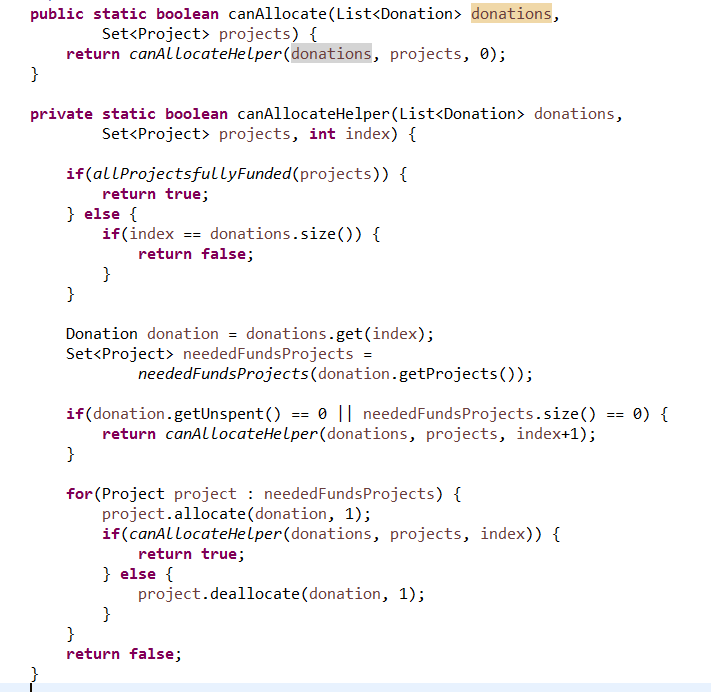
Task2



Donation number: n

Project number: m

Total number of dollars worth of donations available: x

Total number of dollars required to fund all the projects: y

Algorithm analysis:

According to task 1, the recursive method canAllocateHelper(donations,projects,index) will run donations.size() times (because 0<=index<=donations.size()). And then if projects in one donation need to fund, donation will allocate 1 dollars for each project. When all projects in this donation are fully funded, then go to next donation until last donation.

If X <= Y, this recursive method will run n\*m\*x times.

If X > Y, this recursive method will run n\*m\*y times.

Example:

If there are n donations and m projects, suppose each donation has m projects, so there is n\*m times to donate all projects from first donation to last donation. In addition, all donations total have x dollars, this means all donations have to donate to x times (because allocate one dollar of donation to project each time). Therefore, the worst case behaviour should be Ω(m\*n\*x).