Mini Project 2 – Company Structure

For this project you are going to practice using inheritance, interfaces and abstract classes to objects to one another. The following is a description of each class and its behavior. It is up to you to decide which classes should extend, implement or abstract which pieces to maximize your code sharing.

Employeee

public Employee(String name, double baseSalary) Should construct a new employee object and take in two parameters, one for the name of the user and one for their base salary public double getBaseSalary() Should return the employee's current salary public String getName() Should return the employee's current name public int getEmployeeID() Should return the employee's ID. The ID should be issued on behalf of the employee at the time they are constructed. The first ever employee should have an ID of "1", the second "2" and so on public Employee getManager() Should return a reference to the Employee object that represents this employee's manager public boolean equals(Employee other) Should return true if the two employee IDs are the same, false otherwise public String toString() Should return a String representation of the employee that is a combination of their id followed by their name. Example: "1 Kasey" public String employeeStatus() Should return a String representation of that Employee's current status.

TechnicalEmployee

public TechnicalEmployee(String name) Has a default base salary of 75000 public String employeeStatus() Should return a String representation of this TechnicalEmployee that includes their ID, name and how many successful check ins they have had. Example: "1 Kasey has 10 successful check ins" BusinessEmployee Method Header Description public BusinessEmployee(String name) Has a default salary of 50000 public double getBonusBudget() Should establish a running tally of the remaining bonusBudget for the team this employee supports. How that budget is determined will depend on which type of Business Employee it is public String employeeStatus() Should return a String representation of this BusinessEmployee that includes their ID, name and the size of their currently managed budget. Example: "1 Kasey with a budget of 22500.0”

SoftWareEmployee

public SoftwareEngineer(String name) Should start without access to code and with 0 code check ins public boolean getCodeAccess() Should return whether or not this SoftwareEngineer has access to make changes to the code base public void setCodeAccess(boolean access) Should allow an external piece of code to update the SoftwareEngieer's code privileges to either true or false public int getSuccessfulCheckIns() Should return the current count of how many times this SoftwareEngineer has successfully checked in code public boolean checkInCode() Should check if this SoftwareEngineer's manager approves of their check in. If the check in is approved their successful checkin count should be increased and the method should return "true". If the manager does not approve the check in the SoftwareEngineer's code access should be changed to false and the method should return "false"

TechnicalLead

public Accountant(String name) Should start with a bonus budget of 0 and no team they are officially supporting public TechnicalLead getTeamSupported() Should return a reference to the TechnicalLead that this Accountant is currently supporting. If they have not been assigned a TechnicalLead null should be returned public void supportTeam(TechnicalLead lead) Should allow a reference to a TechnicalLead to be passed in and saved. Once this happens the Accountant's bonus budget should be updated to be the total of each SoftwareEngineer base salary that reports to that TechnicalLead plus 10%. For example, if the TechnicalLead supports 2 SoftwareEngineers, each with a salary of 75000, the Accountant's budget should be 150000 + 15000 for a total of 165000 public boolean approveBonus(double bonus) Should take in a suggested bonus amount and check if there is still enough room in the budget. If the bonus is greater than the remaining budget, false should be returned, otherwise true. If the accountant is not supporting any team false should be returned. public String employeeStatus() Should return a String representation of this Accountant that includes their ID, name, the size of their currently managed budget and the name of the TechnicalLead they are currently supporting. Example: "1 Kasey with a budget of 22500.0 is supporting Satya Nadella"

BusinessLead

public BusinessLead(String name) Should create a new BusinessLead that is a Manager. The BusinessLead's base salary should be twice that of an Accountant. They should start with a head count of 10. public boolean hasHeadCount() Should return true if the number of direct reports this manager has is less than their headcount. public boolean addReport(Accountant e, TechnicalLead supportTeam) Should accept the reference to an Accountant object, and if the BusinessLead has head count left should add this employee to their list of direct reports. If the employee is successfully added to the BusinessLead's direct reports true should be returned, false should be returned otherwise. Each time a report is added the BusinessLead's bonus budget should be increased by 1.1 times that new employee's base salary. That employee's team they are supporting should be updated to reflect the reference to the TechnicalLead given. If the employee is successfully added true should be returned, false otherwise. public boolean requestBonus(Employee e, double bonus) Should check if the bonus amount requested would fit in current BusinessLead's budget. If it is, that employee should get that bonus, the BusinessLeader's budget should be deducted and true should be returned. False should be returned otherwise public boolean approveBonus(Employee e, double bonus) This function should look through the Accountants the BusinessLead manages, and if any of them are supporting a the TechnicalLead that is the manager of the Employee passed in then the Accountant's budget should be consulted to see if the bonus could be afforded. If the team can afford the bonus it should be rewarded and true returned, false otherwise