Logo, company name

Description automatically generated

**UNITAR GRADUATE SCHOOL**

**Course: ITWM5113 Software Design and Development**

**Course Instructor: Simon Lau**

**Assignment Submission**

**(Group/Individual)**

Assignment Title: Company Structure

| **Name** | **Student ID** | **Section** |
| --- | --- | --- |
| A.HALIM BIN AMINNUDIN | MC220517284 | Slide + Presentation + Recording |
| MOHD SOUFI BIN YUSSOF | UNU2200768 | Report + Slide |
| DEVAN THANGA SELVARAJ | UNU2200575 | Coding + Report + Slide |

TABLE OF CONTENTS

[**1.0 ASSIGNMENT OBJECTIVE 2**](#_Toc109335635)

[**2.0 COMPANY ORGANISATION 2**](#_Toc109335636)

[**3.0 UML DIAGRAM OF COMPANY STRUCTURE 3**](#_Toc109335637)

[**4.0 UML DIAGRAM OF WORK BREAKDOWN COMPANY STRUCTURE 4**](#_Toc109335638)

[**5.0 PROJECT SOURCE CODE FOR THE COMPANY STRUCTURE 5**](#_Toc109335639)

[5.1 Company Structure 5](#_Toc109335640)

[5.3 BusinessEmployee 7](#_Toc109335641)

[5.4 BusinessLead 8](#_Toc109335642)

[5.5 Employee 9](#_Toc109335643)

[5.6 SoftwareEngineer 11](#_Toc109335644)

[5.7 TechnicalEmployee 11](#_Toc109335645)

[5.8 TechnicalLead 12](#_Toc109335646)

[**6.0 Project output code for the Company Structure 14**](#_Toc109335647)

# **1.0 ASSIGNMENT OBJECTIVE**

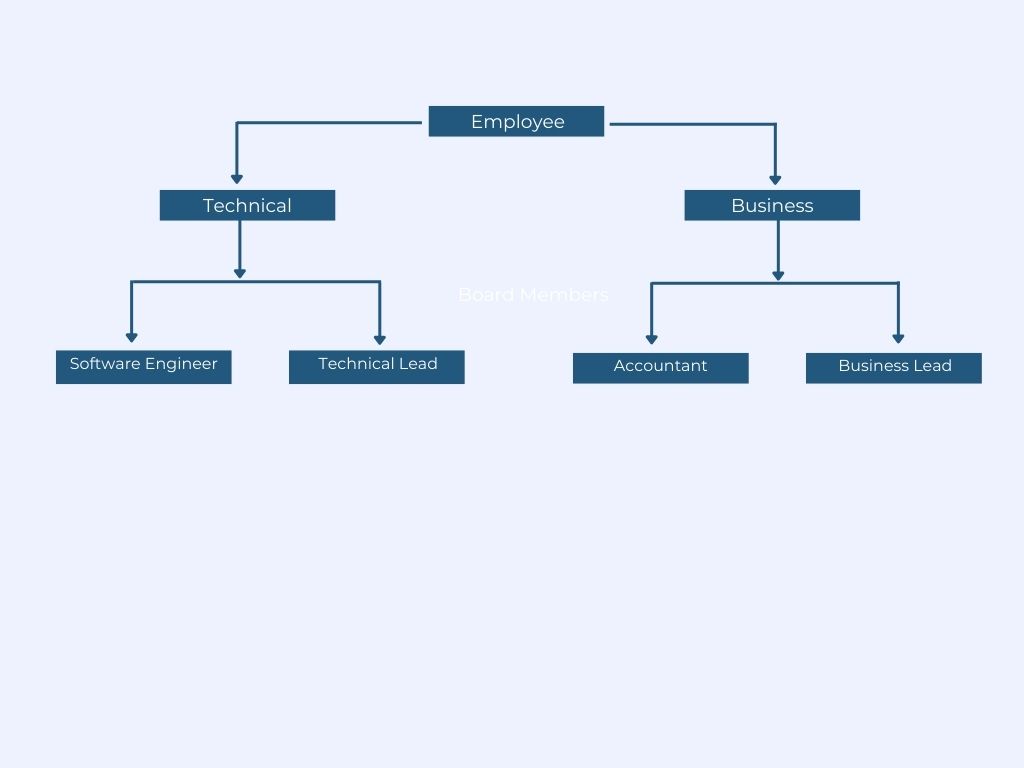
For this assignment, you are going to practice using inheritance, interfaces, composition and realization classes to relate objects to one another. The following is a description of each class and its behaviour. It is up to you to decide which classes should extend, implement or abstract which pieces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

The project has several parts that are employee, technical employee, business employees, software engineers and accountant. The source code developed with practice using inheritance, interfaces, and abstract classes to relate objects to one another. The following is a description of each class and its behaviour.

The system begins with Employee. This is overall parent class, which every one of our following classes will fall under. This class will contain the following methods Employee, getBaseSalary, getEmployeeID, getManager equals, toString and EmployeeStatus. Now, within Employee, we have Technical and Business. Each sub classes will inherit all the

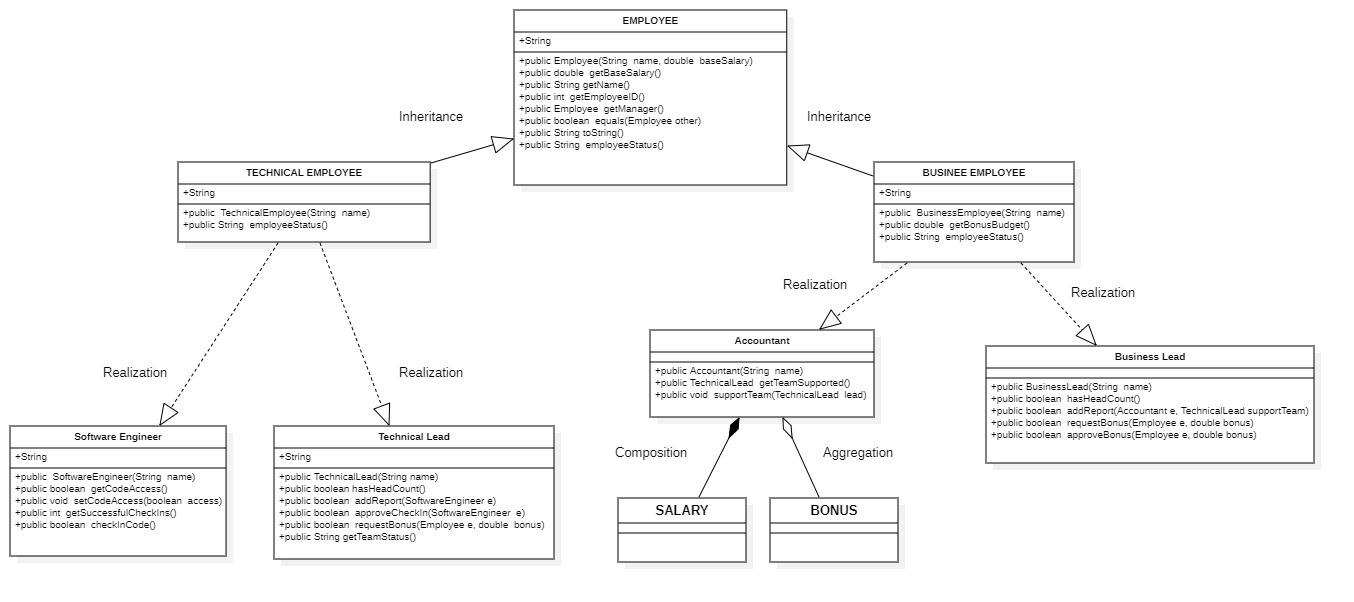
Employee class methods and add more method specific to their roles. Under Technical and Business, we have our sub classes. This time role specific. They will inherit all the methods from both Employee and their main class, whether that be Technical or Business. Then we add the role specific methods to the sub classes

# **2.0 COMPANY ORGANISATION**



*Company Organisation Chart*

1. **UML DIAGRAM OF COMPANY STRUCTURE**



|  |  |  |
| --- | --- | --- |
| **ENTITI / WORK FLOW** | **Class Diagram Relationships** | **Remarks** |
| Employee | Parent | Main Class |
| Technical employee | Inheritance / Generalization | Same functionalities of the parent class. In other words, the child class is a specific type of the parent class |
| Business employee | Inheritance / Generalization | Same functionalities of the parent class. In other words, the child class is a specific type of the parent class |
| Software engineer | Realization | Realization is a relationship between the Technical Employee and the Software Engineer containing its respective implementation level details. This object is said to realize the software development |
| Technical lead | Realization | Realization is a relationship between the Technical Employee and the Technical Lead containing its respective implementation level details. This object is said to realize the software development |
| Accountant | Realization | Realization is a relationship between the Business Employee and the Accountant containing its respective implementation level details. This object is said to realize the accounting company |
| Business lead | Realization | Realization is a relationship between the Business Employee and the Business Lead containing its respective implementation level details. This object is said to realize the business operation |
| Salary | Composition | The lifetimes of both the objects or class are the same. |
| Bonus | Aggregation | Bonus a particular class as a result of one class being aggregated because Bonus is based on company budget and profit |

# **4.0 UML DIAGRAM OF WORK BREAKDOWN COMPANY STRUCTURE**

## **Employee**

This class consists of new employee object and two parameters that are name and base salary. The method header public Employee (String name, double baseSalary) is construct a new employee object and take in two parameters, one for the name of the user and one for their base salary. Method header public double getBaseSalary() should return the employee's current salary. Method header public String getName() should return the employee's current name. Method header 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. This will be different for every subclass of Employee.

## **Technical Employee**

Method header public TechnicalEmployee(String name) has a default base salary of 75000 and 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 and public double getBonusBudget() Should establish a running tally of the remaining bonus Budget for the team this employee supports. How that budget is determined will depend on which type of Business Employee it is public String employee Status () Should return a String representation of this Business Employee that includes their ID, name and the size of their currently managed budget. Example: "1 Kasey with a budget of 22500.0".

## **Software Engineer**

Method header public SoftwareEngineer(String name) should start without access to code and with 0 code check ins. Method header public boolean getCodeAccess() should return whether or not this SoftwareEngineer has access to make changes to the code base. Method header 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. Method header 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".

## **Accountant**

Method header public Accountant(String name) should start with a bonus budget of 0 and no team they are officially supporting. Method header 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's 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. Method header 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. Method header 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".

## **TechnicalLead**

Method header public TechnicalLead(String name) should create a new TechnicalLead that is a Manager. The TechnicalLead's base salary should be 1.3 times that of a TechnicalEmployee. TechnicalLeads should have a default head count of 4. Method header public boolean hasHeadCount() should return true if the number of direct reports this manager has is less than their headcount. Method header public boolean addReport(SoftwareEngineer e) should accept the reference to a SoftwareEngineer object, and if the TechnicalLead has head count left should add this employee to their list of direct reports. If the employee is successfully added to the TechnicalLead's direct reports true should be returned, false should be returned otherwise. Method header public boolean approveCheckIn(SoftwareEngineer e) Should see if the employee passed in does report to this manager and if their code access is currently set to "true". If both those things are true, true is returned, otherwise false is returned public boolean requestBonus(Employee e, double bonus) should check if the bonus amount requested would be approved by the BusinessLead supporting this TechnicalLead. If it is, that employee should get that bonus and true should be returned. False should be returned otherwise. Method header public String getTeamStatus() should return a String that gives insight into this Manager and all their direct reports. It should return a string that is a combination of the TechnicalLead's employee status followed by each of their direct employee's status on subsequent lines. If the TechnicalLead has no reports, it should print their employee status followed by the text " and no direct reports yet ". Example: "10 Kasey has 5 successful check ins and no direct reports yet". If the TechnicalLead does have reports, it might look something like "10 Kasey has 5 successful check ins and is managing: /n 5 Niky has 2 successful check ins".

## **BusinessLead**

Method header 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. Method header public boolean has HeadCount() Should return true if the number of direct reports this manager has is less than their headcount. Method header public boolean add Report (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. Method header 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. Method header 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.

# **5.0 PROJECT SOURCE CODE FOR THE COMPANY STRUCTURE**

## **5.1 Company Structure**

|  |
| --- |
| public class CompanyStructure {  public static void main(String[] args) {  TechnicalLead CTO = new TechnicalLead("Satya Nadella");  SoftwareEngineer seA = new SoftwareEngineer("Kasey");  SoftwareEngineer seB = new SoftwareEngineer("Breana");  SoftwareEngineer seC = new SoftwareEngineer("Eric");  CTO.addReport(seA);  seA.checkInCode();  seA.checkInCode();  CTO.addReport(seB);  CTO.addReport(seC);  seC.checkInCode();  seC.checkInCode();  System.out.println(CTO.getTeamStatus());  TechnicalLead VPofENG = new TechnicalLead("Bill Gates");  SoftwareEngineer seD = new SoftwareEngineer("Winter");  SoftwareEngineer seE = new SoftwareEngineer("Libby");  SoftwareEngineer seF = new SoftwareEngineer("Gizan");  SoftwareEngineer seG = new SoftwareEngineer("Zaynah");  VPofENG.addReport(seD);  VPofENG.addReport(seE);  VPofENG.addReport(seF);  VPofENG.addReport(seG);  seD.checkInCode();  seF.checkInCode();  seF.checkInCode();  seF.checkInCode();  seF.checkInCode();  System.out.println(VPofENG.getTeamStatus());  BusinessLead CFO = new BusinessLead("Amy Hood");  Accountant actA = new Accountant("Niky");  Accountant actB = new Accountant("Andrew");  CFO.addReport(actA, CTO);  CFO.addReport(actB, VPofENG);  System.out.println(CFO.getTeamStatus());  System.out.println(seB.toString() + "'s manager is " + seB.getManager().toString());  System.out.println(seF.toString() + "'s manager is " + seF.getManager().toString());  System.out.println(actB.toString() + "'s manager is " + actB.getManager().toString());  System.out.println();  System.out.println("Testing BusinessLead approvedBonus()");  System.out.print(seA.getManager() + " is asking for $10,000 bonus for "+seA.getName()+", (the Approval result should be TRUE): ");  System.out.println(CTO.requestBonus(seA, 10000));  System.out.println("Updated budget is: "+seA.getManager().getAccountantSupport().getBonusBudget()+"\n");  System.out.print(seF.getManager() + " is asking for $5,000 bonus for "+seF.getName() +", (the Approval result should be TRUE): ");  System.out.println(VPofENG.requestBonus(seF, 5000));  System.out.println("Updated budget is: "+seF.getManager().getAccountantSupport().getBonusBudget()+"\n");  System.out.print(seF.getManager() + " is asking for $400,000 bonus for "+seF.getName()+", (the Approval result should be FALSE): ");  System.out.println(VPofENG.requestBonus(seF, 400000));  System.out.println("Updated budget is: "+seF.getManager().getAccountantSupport().getBonusBudget()+"\n");  System.out.println();  }  } |

**5.2 Accountant**

|  |
| --- |
| public class Accountant extends BusinessEmployee {  protected TechnicalLead teamSupported;  //Assignment 2:Should start bonus budget of 0.  public Accountant(String name){  super(name);  bonusBudget=0;  }  //Should return a reference to TechnicalLead that this Accountant currently supporting.TechnicalLead null  // should be returned if they have not been assigned .  protected TechnicalLead getTeamSupported(){  return this.teamSupported;  }  //allow reference to TechnicalLead to be passed in & saved.  // Accountant's bonus budget updated in the total SoftwareEngineer salary.  // reports to that TechnicalLead plus ten(10)%.  protected void supportTeam(TechnicalLead lead){  this.teamSupported=lead;  int bound = lead.team.size();  for (int i = 0; i < bound; i++) {  this.bonusBudget += lead.team.get(i).getBaseSalary() \* 1.1;  }  }  //take in suggested bonus amount & check if there still enough room in budget.  // false returned if the bonus is > remaining budget.  // false returned if accountant not supporting any team.  public boolean canApproveBonus(double bonus){  if (bonus <=getBonusBudget()){  return true;  } else {  System.out.print(" Rejected because Budget not sufficient. ");  return false;  }  }  //return a String representation name, their ID,the size  // of their currently managed budget & the name of TechnicalLead and  // currently supporting.  public String employeeStatus(){  return this +" with a budget of "+ getBonusBudget()+" is supporting "+ this.getTeamSupported();  }  } |

## **5.3 BusinessEmployee**

|  |
| --- |
| class BusinessEmployee extends Employee {  //default salary 50000.  public BusinessEmployee(String name){  super(name,50000.00);  }  //Should establish a running tally with remain bonusBudget for the employee support team.  // budget determined will depend on which type of Business\_Employee.  public double getBonusBudget(){  return bonusBudget;  }  //return a String represent BusinessEmployee including name ,includes their ID,  //Size of their currently budget.  public String employeeStatus(){  String s= String.format("%.2f",bonusBudget);  return this + " with a budget of " + s;  }  } |

## **5.4 BusinessLead**

|  |
| --- |
| import java.util.ArrayList;  public class BusinessLead extends BusinessEmployee{  public ArrayList<Accountant> team;  //Assignment 2 :Should create a new BusinessLead that is a Manager.  public BusinessLead(String name){  super(name);  this.basicSalary=getBaseSalary()\*2;  this.headcount=10;  this.team= new ArrayList<>();  }  public boolean hasHeadCount(){  return this.team.size() < this.headcount;  }  public void addReport(Accountant e, TechnicalLead supportTeam){  if (hasHeadCount()){  team.add(e);  e.setManager(this);  this.bonusBudget+=e.basicSalary\*1.1;  e.supportTeam(supportTeam);  supportTeam.accountantSupport=e;  }  }  public boolean approveBonus(Employee e, double bonus){  for (Accountant accountant : team) {  if ((accountant.getTeamSupported()).equals(e.manager) && accountant.canApproveBonus(bonus)) {  e.bonus += bonus;  accountant.bonusBudget -= bonus;  return true;  }  }  return false;  }  public String getTeamStatus(){  if (team.size()==0){  return this.employeeStatus()+ " and no direct reports yet";  } else {  StringBuilder teamStatus= new StringBuilder();  for (Accountant accountant : team) {  teamStatus.append(" ").append(accountant.employeeStatus()).append("\n");  }  return this.employeeStatus()+" and is managing: \n"+teamStatus;  }  }  } |

## **5.5 Employee**

|  |
| --- |
| class Employee {  protected String name;  protected int employeeID = countID;  protected static int countID;  protected double bonus;  protected Employee manager;  protected int headcount=0;  protected double basicSalary;  protected Accountant accountantSupport;  protected double bonusBudget;  //Assignment 2 Should\_construct a new employee object and take in two parameters, one for the name  // of the user and one for base salary.  protected Employee(String name, double baseSalary){  this.name=name;  int i = ++countID;  this.basicSalary=baseSalary;  }  //return the employee's salary.  protected double getBaseSalary(){  return this.basicSalary;  }  //return the employee's\_name.  public String getName(){  return this.name;  }  //return the employee's ID & issued on behalf of the employee at  // the time they constructed. Employee have ID 1,  // the second 2 and the third 3 so on.  protected int getEmployeeID(){  return this.employeeID;  }  //Should\_return a reference to Employee object  // that represents this manager of employees.  protected Employee getManager(){  return manager;  }  protected Accountant getAccountantSupport() {  return this.accountantSupport;  }  protected void setManager(Employee manager){  this.manager=manager;  }  //Should\_return true if the two employee\_IDs are the same,  // false otherwise.  protected boolean equals(Employee other){  return this.getEmployeeID()==other.getEmployeeID();  }  //Should\_return a String. employee representation that is a combination of their id  // followed by the employee name.  public String toString(){  String s;  s = getEmployeeID() + " " + getName();  return s;  }  //String should return as representation of that Employee's current status.  // This will be different for every subclass of Employee.  protected String employeeStatus() {  return null;  }  } |

## **5.6 SoftwareEngineer**

|  |
| --- |
| class SoftwareEngineer extends TechnicalEmployee{  protected boolean CodeAccess;  //Assignment2 :start witout acess\_code &with 0 code check  protected SoftwareEngineer(String name)  {  super(name);setCodeAccess();  }  //return whether or not this SoftwareEngineer has access make changes code base.  protected boolean getCodeAccess(){  return CodeAccess;  }  //Should allow external piece ofcode to update the  // SoftwareEngineer's code privileges to either true/false.  protected void setCodeAccess(){  this.CodeAccess = true;  }    public void checkInCode(){  var manager = (TechnicalLead) this.getManager();  if (manager.approveCheckIn(this)) this.checkins++;  else {  CodeAccess=false;  }  }  } |

## **5.7 TechnicalEmployee**

|  |
| --- |
| class TechnicalEmployee extends Employee {  protected int checkins;  //Assignment 2: Has a default base salary of 75000.  protected TechnicalEmployee(String name){  super(name,75000.00);checkins=0;  }  //return a String representation of this Technical\_Employee that includes  // ID ,successful check ins &name and how many successful check ins.  protected String employeeStatus(){  var s = super.toString() + " has " + checkins + " successful check ins";  return s;  }  } |

## **TechnicalLead**

|  |
| --- |
| import java.util.ArrayList;  class TechnicalLead extends TechnicalEmployee {  public ArrayList<SoftwareEngineer> team;  {  team = new ArrayList<>();  }  //Assignment 2 create a new TechnicalLead that is a Manager &  //salary be 1.3 times of TechnicalEmployee.  public TechnicalLead(String name){  super(name);  this.basicSalary = this.basicSalary \* 1.3;  headcount=4;  }  //return true if the number of direct reports manager  // has is < their headcount.  public boolean hasHeadCount(){  boolean b;  b = team.size() < headcount;  return b;  }  //accept reference to SoftwareEngineer object.  public void addReport(SoftwareEngineer e){  if (!hasHeadCount()) {  return;  }  boolean add;  add = team.add(e);  e.setManager(this);  }  protected boolean approveCheckIn(SoftwareEngineer e){  return e.getManager().equals(this) && e.getCodeAccess();  }  //Should check if the bonus amount requested would be approved by the BusinessLead supporting this TechnicalLead.  protected boolean requestBonus(Employee e, double bonus){  BusinessLead businessLead = (BusinessLead) getAccountantSupport().getManager();  boolean b = businessLead.approveBonus(e, bonus);  boolean b1;  b1 = b;  return b1;  }  protected String getTeamStatus(){  if (team.size() != 0) {  StringBuilder teamStatus= new StringBuilder();  for (SoftwareEngineer softwareEngineer : team) {  teamStatus.append(" ").append(softwareEngineer.employeeStatus()).append("\n");  }  String s;  s = this.employeeStatus() + " and is managing: \n" + teamStatus;  return s;  } else {  String s;  s = this.employeeStatus() + " and no direct reports yet";  return s;  }  }  } |

# **6.0 Project output code for the Company Structure**

|  |
| --- |
| 0 Satya Nadella has 0 successful check ins and is managing:  1 Kasey has 2 successful check ins  2 Breana has 0 successful check ins  3 Eric has 2 successful check ins  4 Bill Gates has 0 successful check ins and is managing:  5 Winter has 1 successful check ins  6 Libby has 0 successful check ins  7 Gizan has 4 successful check ins  8 Zaynah has 0 successful check ins  9 Amy Hood with a budget of 110000.00 and is managing:  10 Niky with a budget of 247500.0 is supporting 0 Satya Nadella  11 Andrew with a budget of 330000.0 is supporting 4 Bill Gates  2 Breana's manager is 0 Satya Nadella  7 Gizan's manager is 4 Bill Gates  11 Andrew's manager is 9 Amy Hood  Testing BusinessLead approvedBonus()  0 Satya Nadella is asking for $10,000 bonus for Kasey, (the Approval result should be TRUE): true  Updated budget is: 237500.0  4 Bill Gates is asking for $5,000 bonus for Gizan, (the Approval result should be TRUE): true  Updated budget is: 325000.0  4 Bill Gates is asking for $400,000 bonus for Gizan, (the Approval result should be FALSE): Rejected because Budget not sufficient. false  Updated budget is: 325000.0  Process finished with exit code 0 |