COMPANY STRUCTURE

GROUP ASSIGNMENT 3

Matrix# : A.HALIM BIN AMINNUDIN | MOHD SOUFI BIN YUSSOF | DEVAN
MC220517284 | UNU2200768 | UNU2200575

INSTRUCTOR: Dr. Simon Lau

ITWM 5113 SOFTWARE DESIGN AND DEVELOPMENT

28 JUL 2022

INTRODUCTION GROUP MEMBER

INTRODUCTION GROUP MEMBER

OBJECTIVE

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 system begins with Employee. This is overall parent class, which every one of our following classes will fall under.

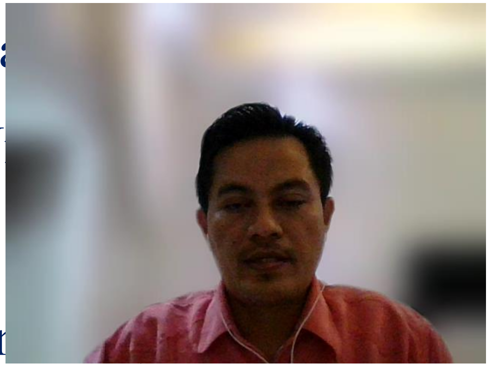
This class will contain the following methods Employee, getBaseSalary, getEmployeeID, getManager equals, toString and EmployeeStatus. Now, within Employee, we have Technical and Business. Ea

inherit all the Employee class methods and add more method specific to their roles. U

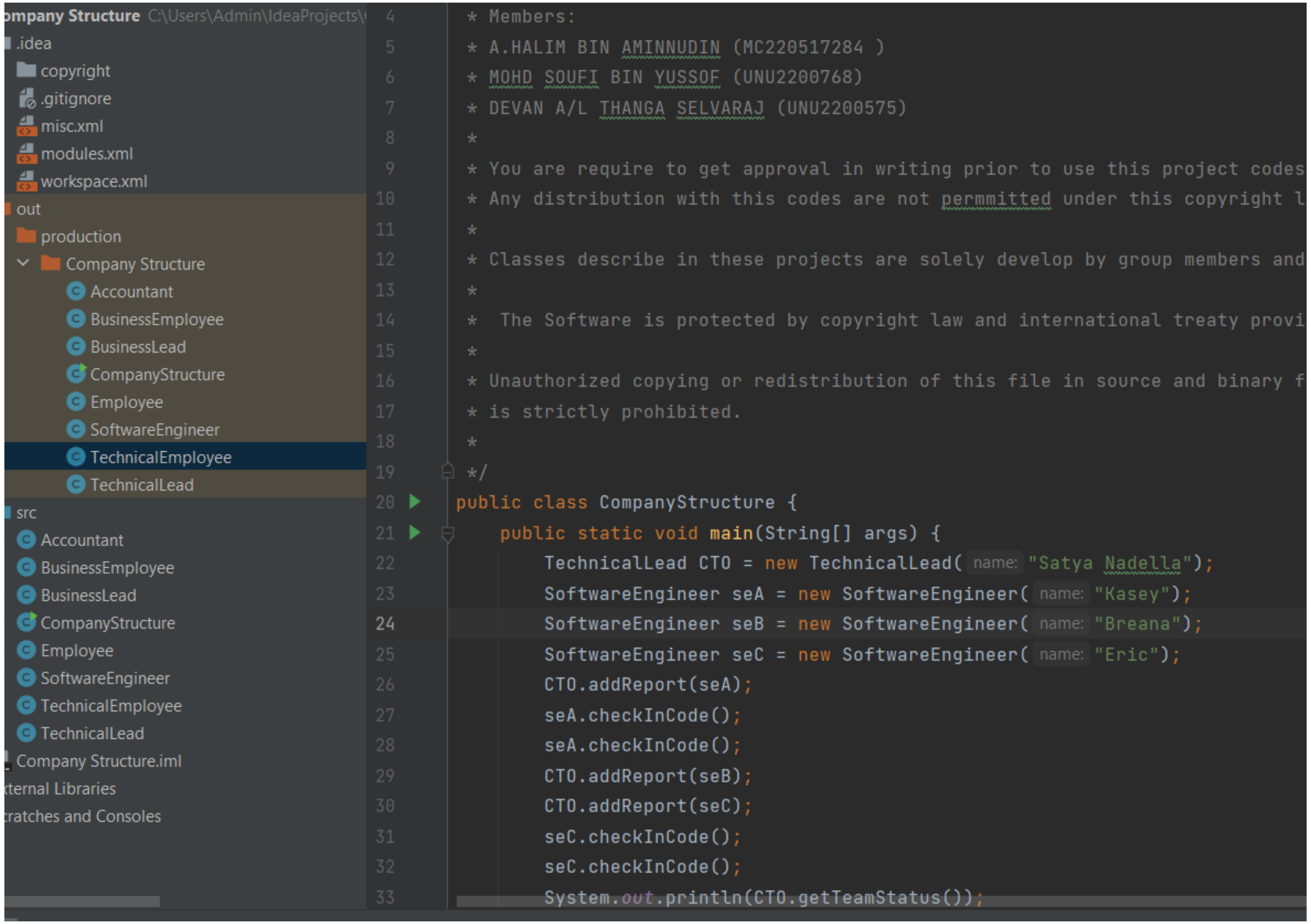
Business, we have our sub classes.

This time role specific. They will inherit all the methods from both Employee and their main

be Technical or Business. Then we add the role specific methods to the sub classes.

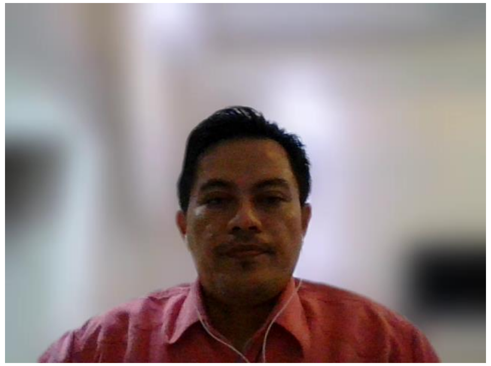


CODING DESIGN STANDARDS



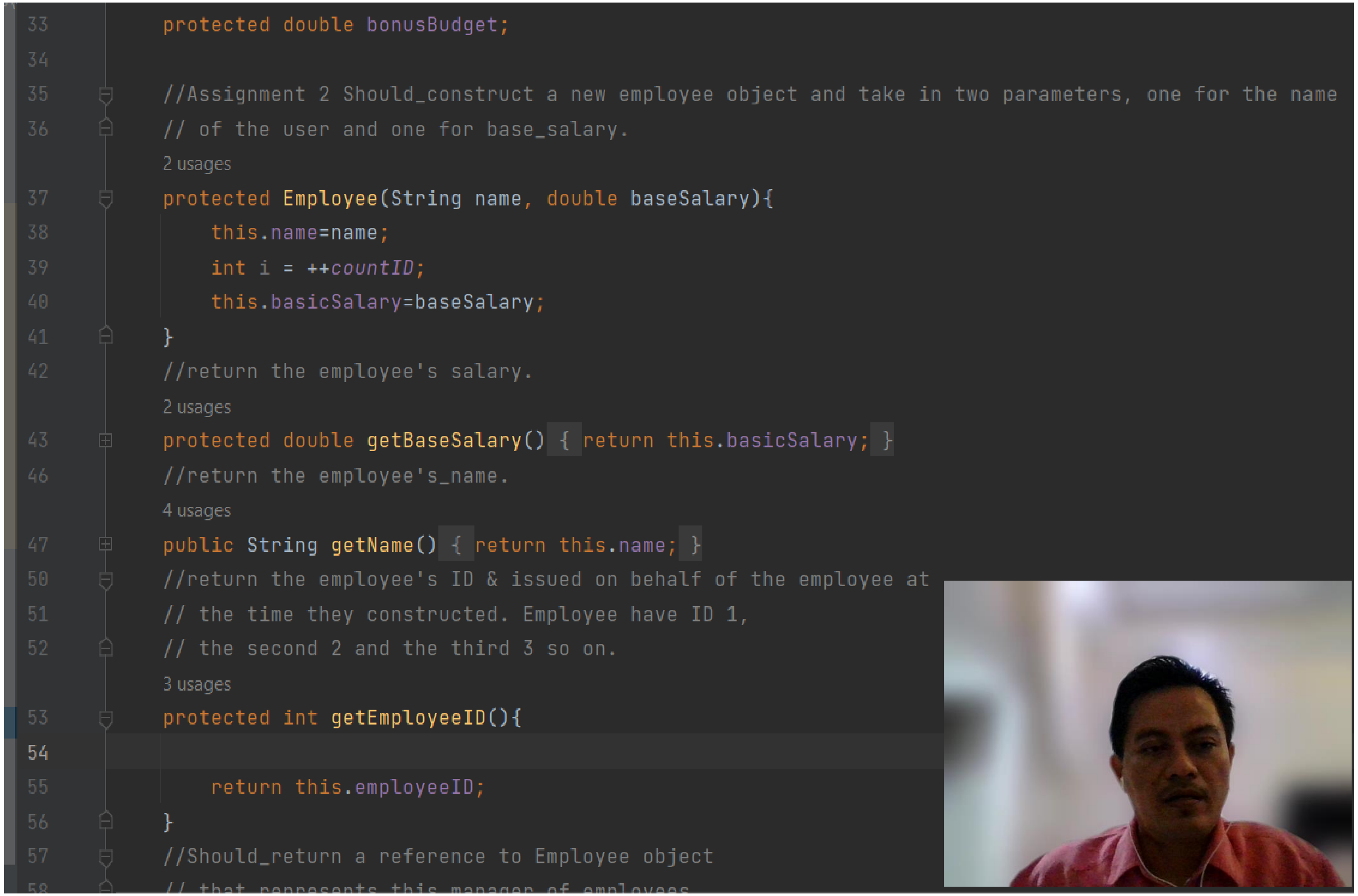
```
 * Members:
 * A.HALIM BIN AMINNUDIN (MC220517284 )
 * MOHD SOUFI BIN YUSSOF (UNU2200768)
 * DEVAN A/L THANGA SELVARAJ (UNU2200575)
 *
 * You are require to get approval in writing prior to use this project codes
 * Any distribution with this codes are not permmitted under this copyright l
 *
 * Classes describe in these projects are solely develop by group members and
 *
 *  The Software is protected by copyright law and international treaty provi
 *
 * Unauthorized copying or redistribution of this file in source and binary f
 * is strictly prohibited.
 *
 */
public class CompanyStructure {
    public static void main(String[] args) {
        TechnicalLead CTO = new TechnicalLead( name: "Satya Nadella");
        SoftwareEngineer seA = new SoftwareEngineer( name: "Kasey");
        SoftwareEngineer seB = new SoftwareEngineer( name: "Breana");
        SoftwareEngineer seC = new SoftwareEngineer( name: "Eric");
        CTO.addReport(seA);
        seA.checkInCode();
        seA.checkInCode();
        CTO.addReport(seB);
        CTO.addReport(seC);
        seC.checkInCode();
        seC.checkInCode();
        System.out.println(CTO.getTeamStatus());
```

The coding design arrangement is well-structured and systematic, and it follows best practices.



UNITAR

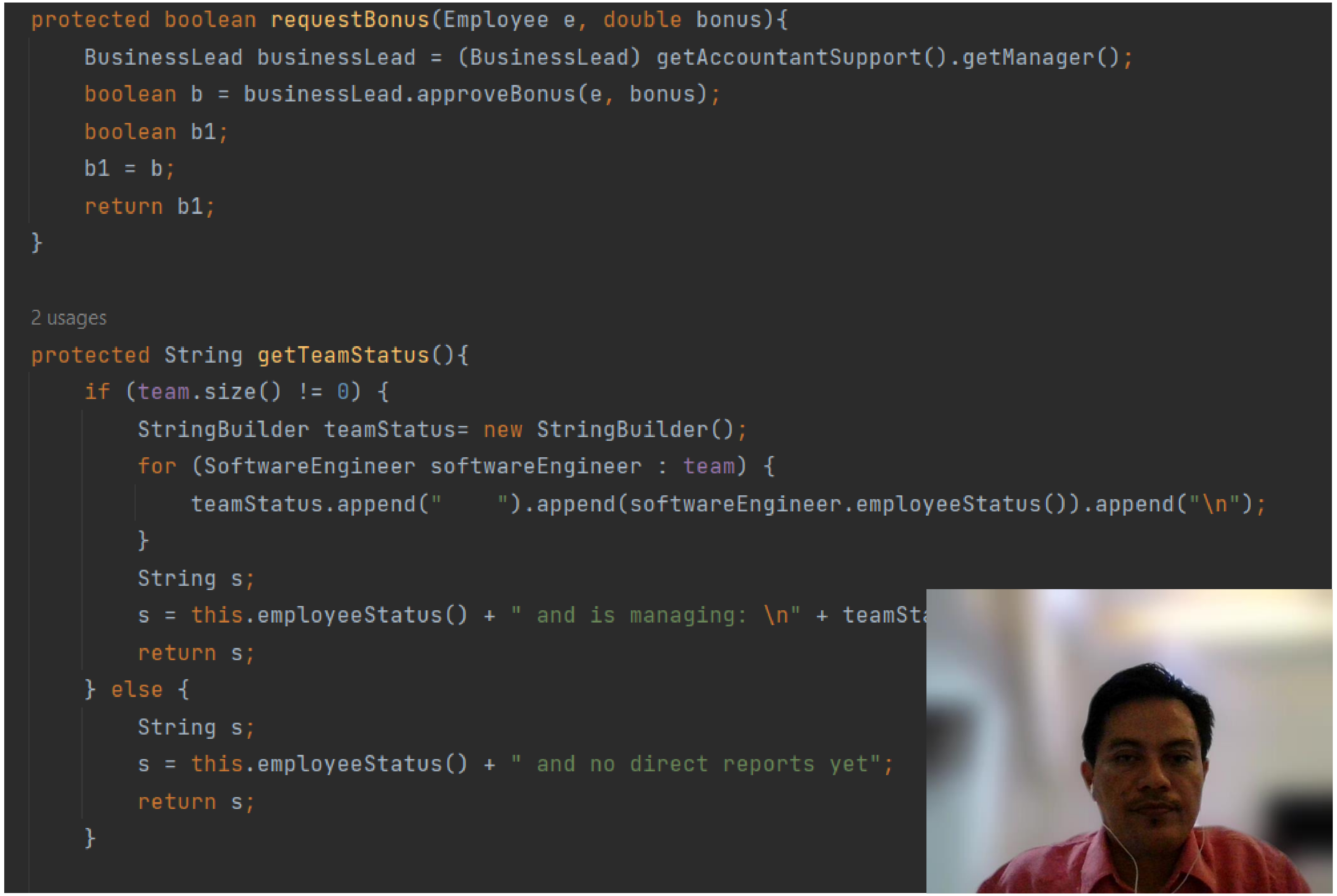
CODING DESIGN STANDARDS



```
    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.
    2 usages
    protected Employee(String name, double baseSalary){
        this.name=name;
        int i = ++countID;
        this.basicSalary=baseSalary;
    }
    //return the employee's salary.
    2 usages
    protected double getBaseSalary() { return this.basicSalary; }
    //return the employee's_name.
    4 usages
    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.
    3 usages
    protected int getEmployeeID(){


        return this.employeeID;
    }
    //Should_return a reference to Employee object
    // that represents this manager of employees
```

Further development will be enhanced by the use of professional code and the identification of each coding name as meaningful and readable for each object, variable, class, and constant.

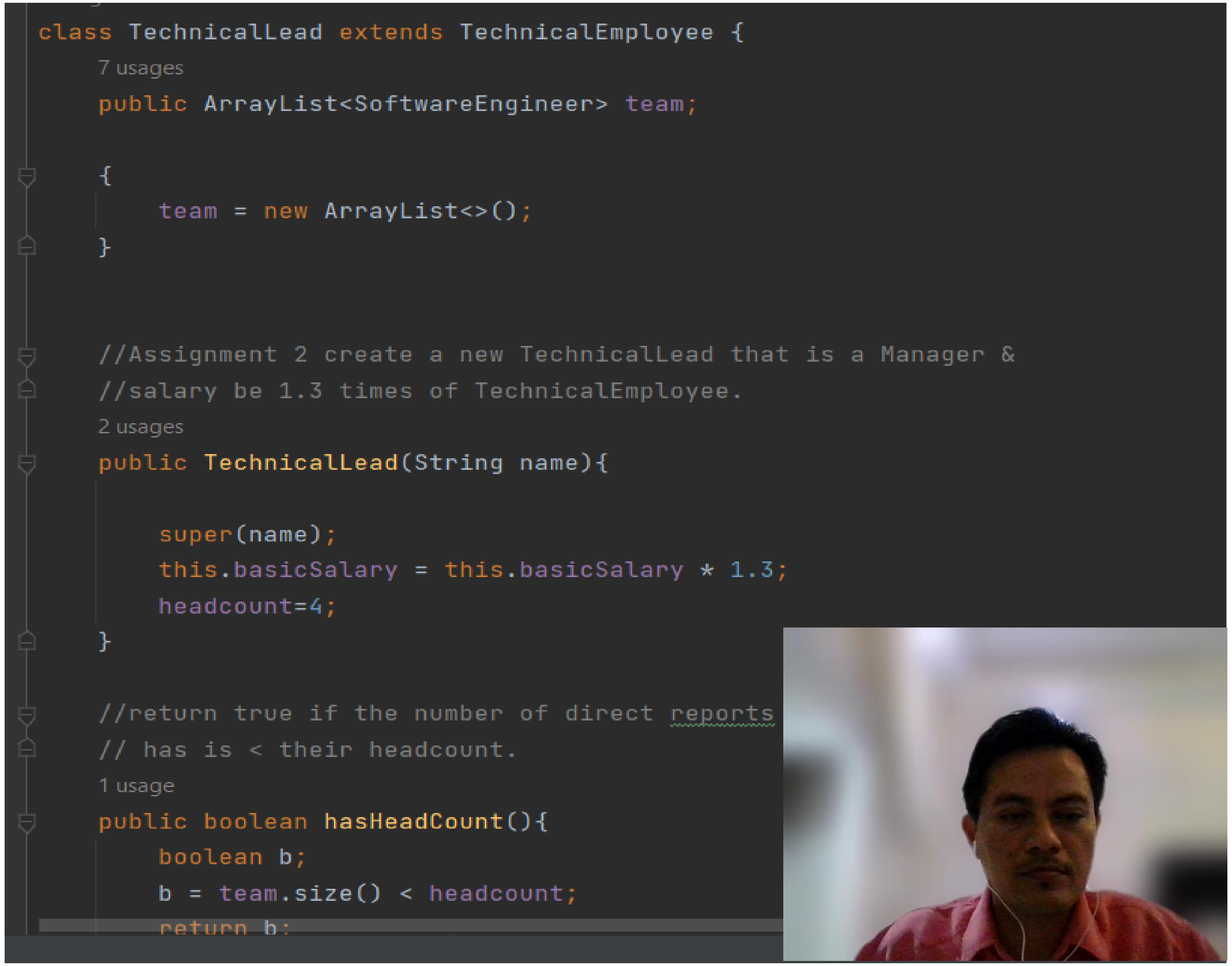
CODING DESIGN STANDARDS



```
protected boolean requestBonus(Employee e, double bonus){
    BusinessLead businessLead = (BusinessLead) getAccountantSupport().getManager();
    boolean b = businessLead.approveBonus(e, bonus);
    boolean b1;
    b1 = b;
    return b1;
}

2 usages
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" + teamSta
        return s;
    } else {
        String s;
        s = this.employeeStatus() + " and no direct reports yet";
        return s;
    }
```

Making use of white space and suitable indentation makes it simple to identify the boundaries of functions, loops, and conditional blocks as well as to comprehend the flow of the code and function.

CODING DESIGN STANDARDS

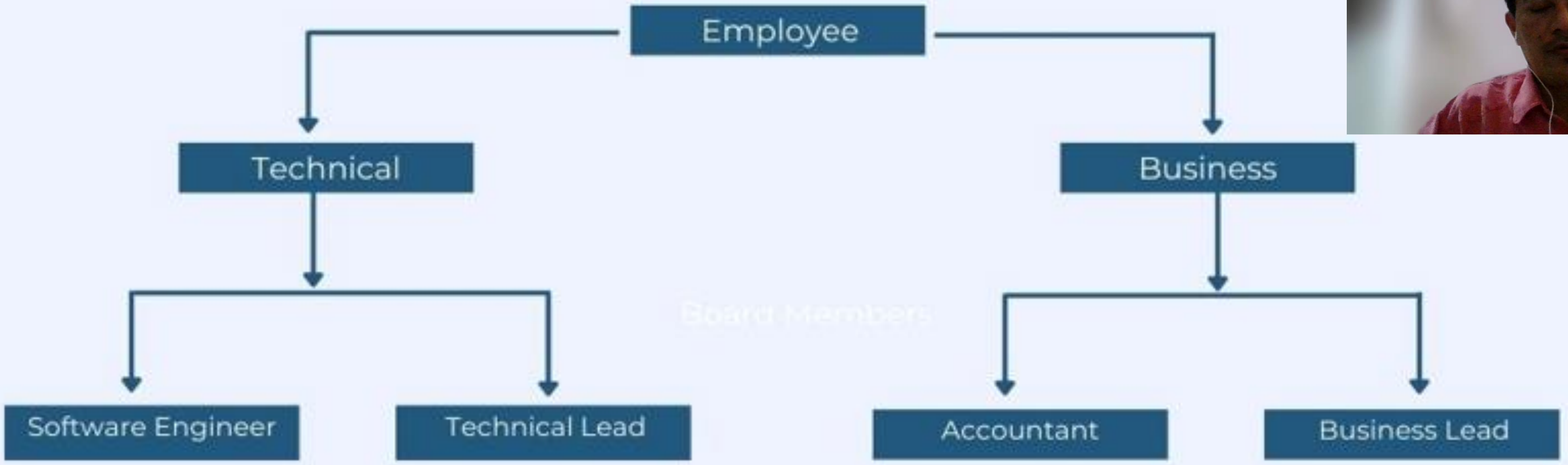


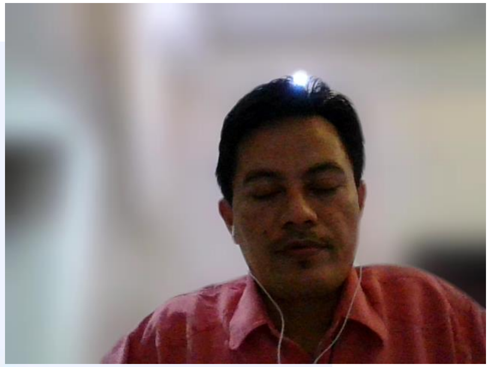
```
class TechnicalLead extends TechnicalEmployee {
    7 usages
    public ArrayList<SoftwareEngineer> team;

    {
        team = new ArrayList<>();
    }


    //Assignment 2 create a new TechnicalLead that is a Manager &
    //salary be 1.3 times of TechnicalEmployee.
    2 usages
    public TechnicalLead(String name){

        super(name);
        this.basicSalary = this.basicSalary * 1.3;
        headcount=4;
    }

    //return true if the number of direct reports
    // has is < their headcount.
    1 usage
    public boolean hasHeadCount(){
        boolean b;
        b = team.size() < headcount;
        return b;
```

Code commenting will helps a project's codebase maintainable. It can speed up code reviews, making it easier for newcomers and supporting developers to get up to speed on a codebase.

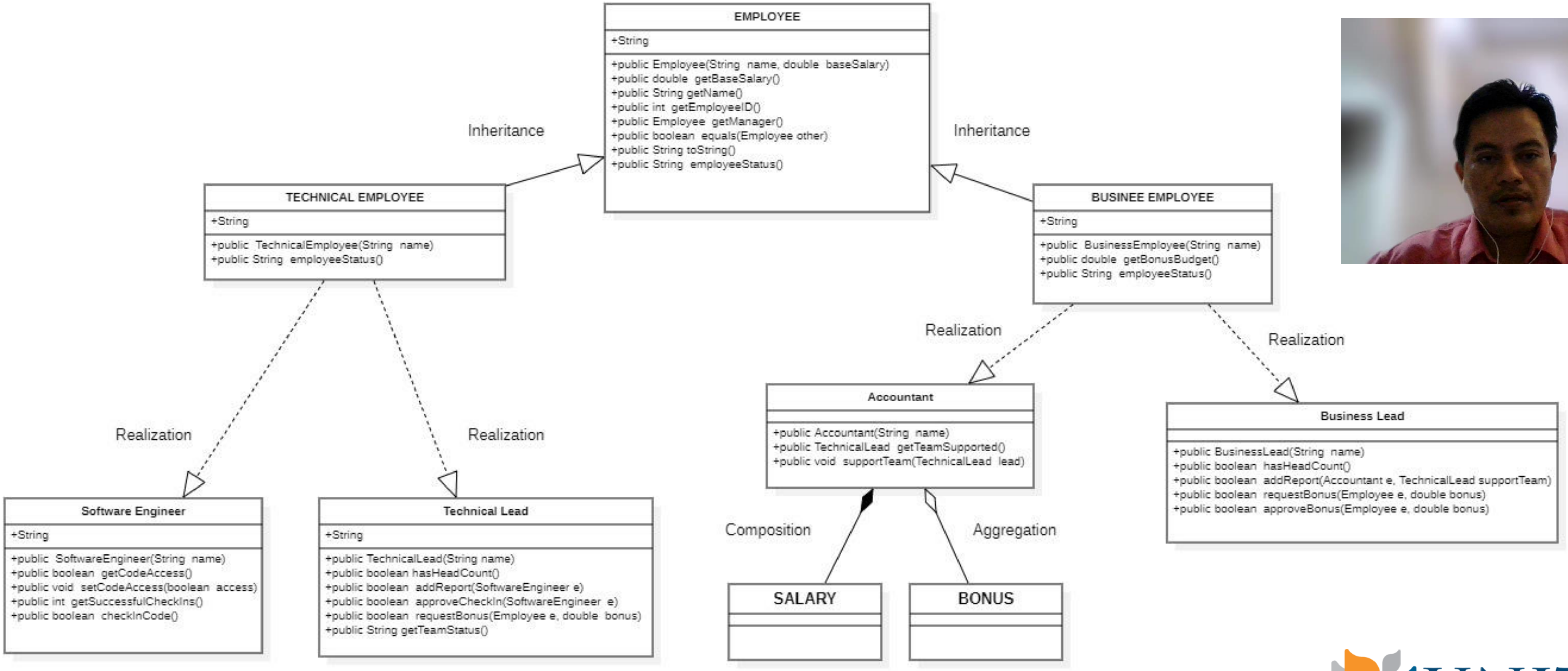
INTRODUCTION GROUP MEMBER

COMPANY ORGANISATION CHART





UML DIAGRAM C0MPANY STRCTURE



EMPLOYEE

+String

+public Employee(String name, double baseSalary)
+public double getBaseSalary()
+public String getName()
+public int getEmployeeID()
+public Employee getManager()
+public boolean equals(Employee other)
+public String toString()
+public String employeeStatus()

Inheritance

TECHNICAL EMPLOYEE

+String

+public TechnicalEmployee(String name)
+public String employeeStatus()

Inheritance

BUSINEE EMPLOYEE

+String

+public BusinessEmployee(String name)
+public double getBonusBudget()
+public String employeeStatus()

Realization

Realization

Realization

Realization

Software Engineer

+String

+public SoftwareEngineer(String name)
+public boolean getCodeAccess()
+public void setCodeAccess(boolean access)
+public int getSuccessfulCheckIns()
+public boolean checkInCode()

Technical Lead

+String

+public TechnicalLead(String name)
+public boolean hasHeadCount()
+public boolean addReport(SoftwareEngineer e)
+public boolean approveCheckIn(SoftwareEngineer e)
+public boolean requestBonus(Employee e, double bonus)
+public String getTeamStatus()

Accountant

+public Accountant(String name)
+public TechnicalLead getTeamSupported()
+public void supportTeam(TechnicalLead lead)

Composition

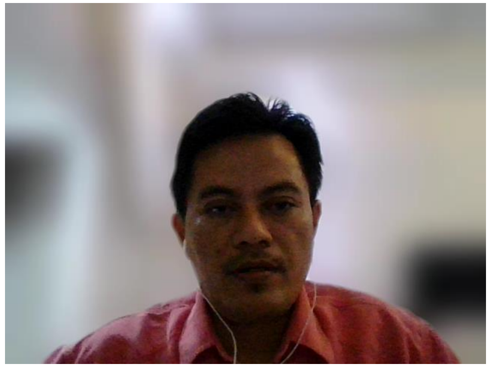
Aggregation

SALARY

BONUS

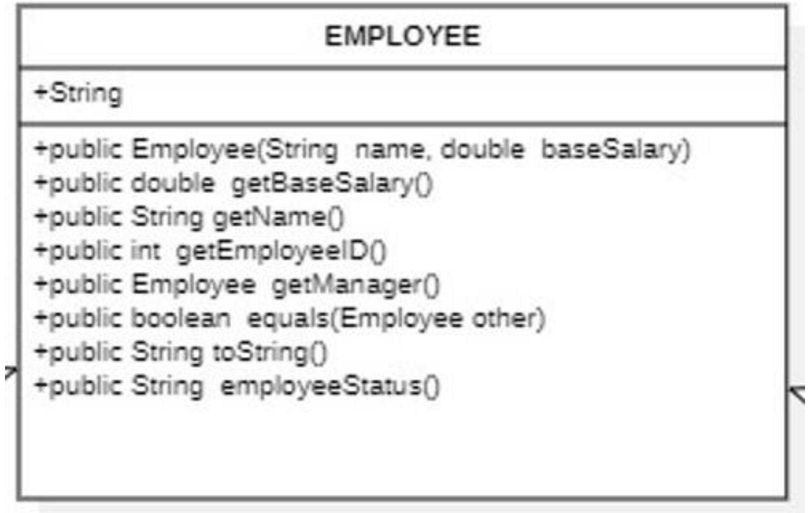
Business Lead

+public BusinessLead(String name)
+public boolean hasHeadCount()
+public boolean addReport(Accountant e, TechnicalLead supportTeam)
+public boolean requestBonus(Employee e, double bonus)
+public boolean approveBonus(Employee e, double bonus)



UNITAR

UML DIAGRAM C0MPANY STRCTURE

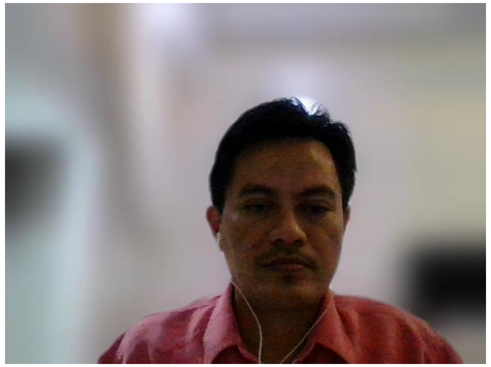


EMPLOYEE

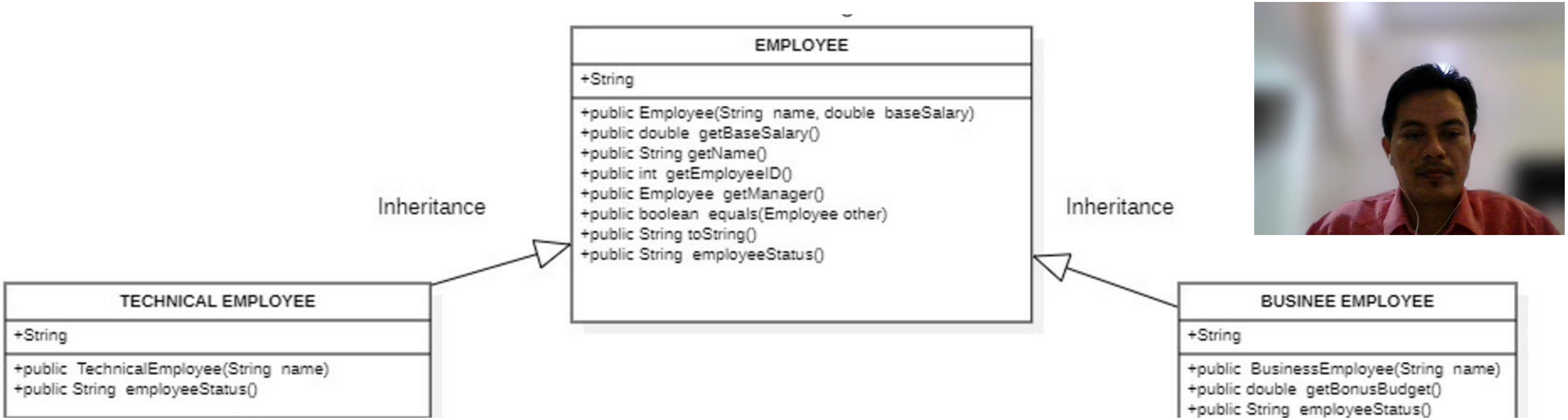
+String

+public Employee(String name, double baseSalary)
+public double getBaseSalary()
+public String getName()
+public int getEmployeeID()
+public Employee getManager()
+public boolean equals(Employee other)
+public String toString()
+public String employeeStatus()

ENTITI / WORK FLOW	Class Diagram Relationships	Remarks
Employee	Parent	Main Class



UML DIAGRAM C0MPANY STRCTURE



EMPLOYEE

+String

+public Employee(String name, double baseSalary)
+public double getBaseSalary()
+public String getName()
+public int getEmployeeID()
+public Employee getManager()
+public boolean equals(Employee other)
+public String toString()
+public String employeeStatus()

TECHNICAL EMPLOYEE

+String

+public TechnicalEmployee(String name)
+public String employeeStatus()

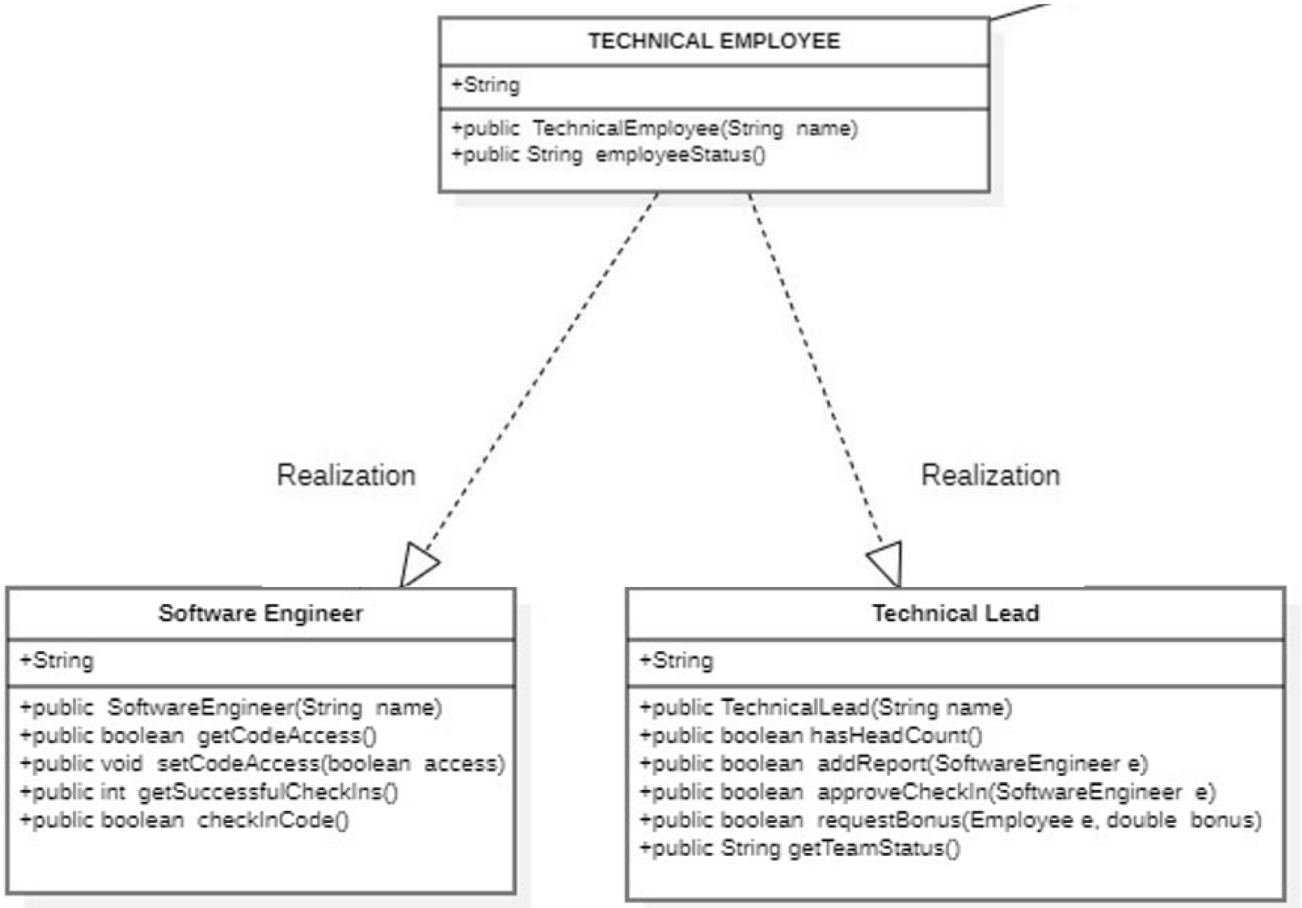
BUSINEE EMPLOYEE

+String

+public BusinessEmployee(String name)
+public double getBonusBudget()
+public String employeeStatus()

ENTITI / WORK FLOW	Class Diagram Relationships	Remarks
Technical employee	Inheritance	Same functionalities of the parent class. In other words, the child class is a specific type of the parent class
Business employee		

UML DIAGRAM C0MPANY STRCTURE



TECHNICAL EMPLOYEE

+String

+public TechnicalEmployee(String name)
+public String employeeStatus()

Realization

Realization

Software Engineer

+String

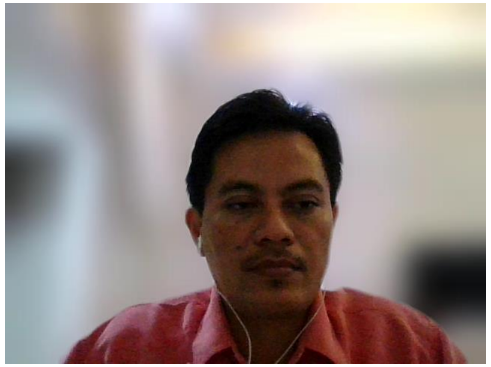
+public SoftwareEngineer(String name)
+public boolean getCodeAccess()
+public void setCodeAccess(boolean access)
+public int getSuccessfulCheckIns()
+public boolean checkInCode()

Technical Lead

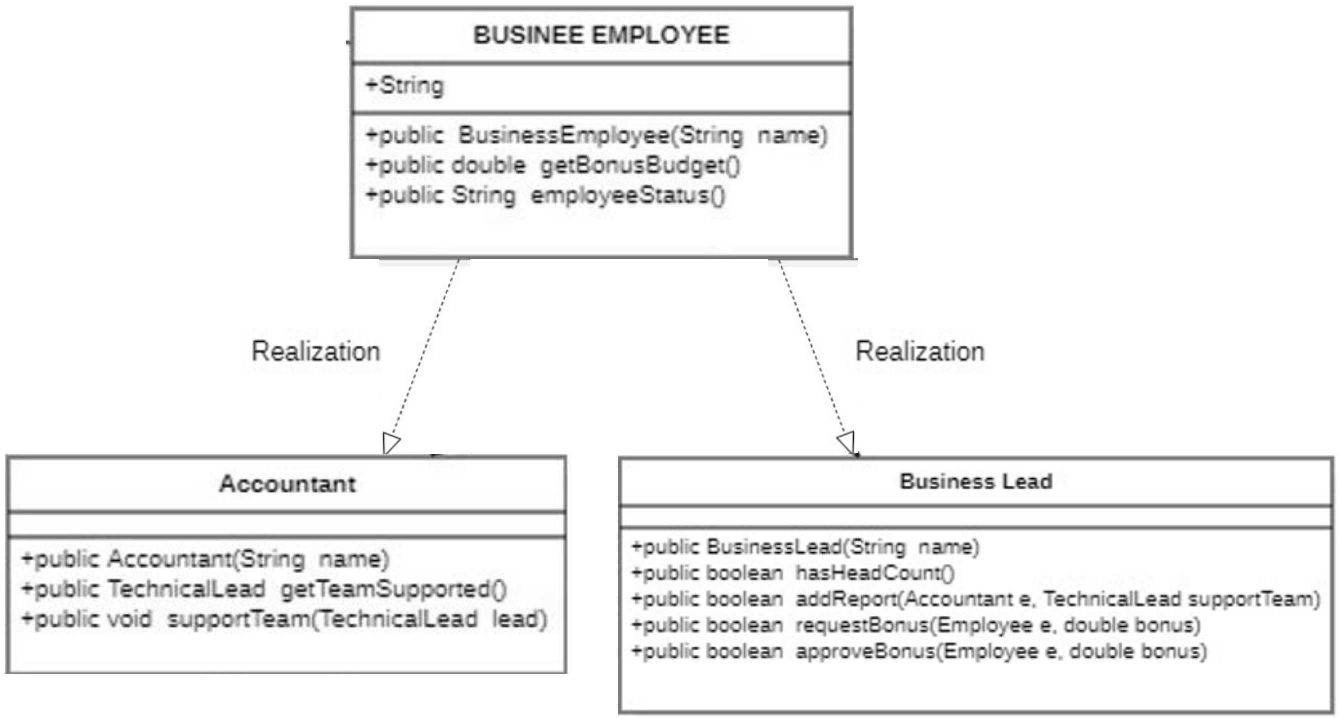
+String

+public TechnicalLead(String name)
+public boolean hasHeadCount()
+public boolean addReport(SoftwareEngineer e)
+public boolean approveCheckIn(SoftwareEngineer e)
+public boolean requestBonus(Employee e, double bonus)
+public String getTeamStatus()

ENTITI / WORK FLOW	Class Diagram Relationships	Remarks
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 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



UML DIAGRAM C0MPANY STRCTURE



BUSINEE EMPLOYEE

+String

+public BusinessEmployee(String name)
+public double getBonusBudget()
+public String employeeStatus()

Accountant

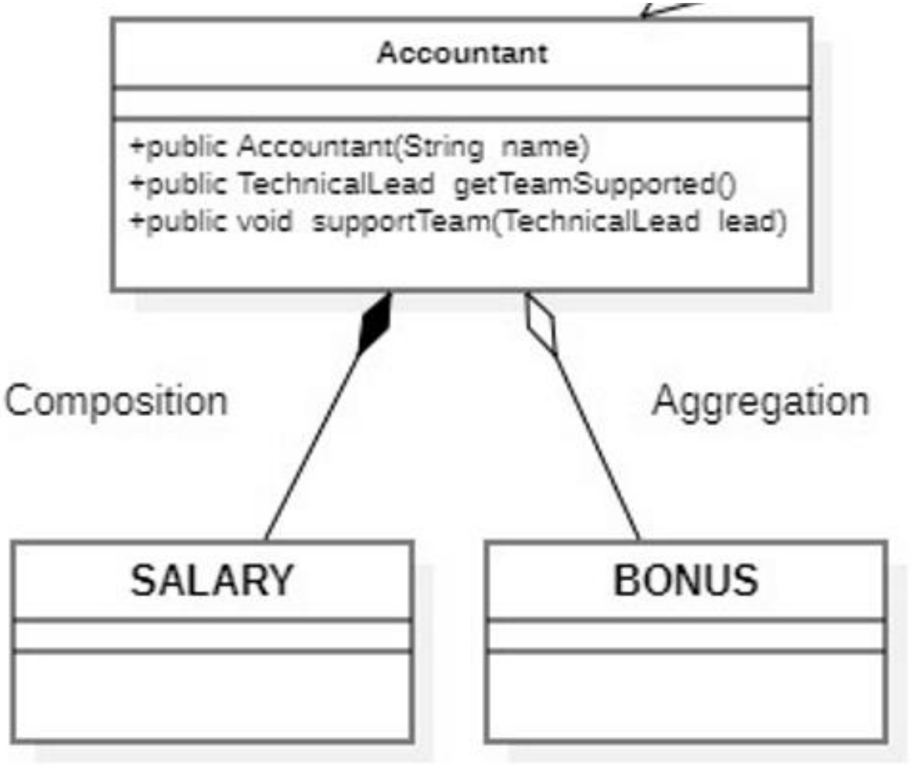
+public Accountant(String name)
+public TechnicalLead getTeamSupported()
+public void supportTeam(TechnicalLead lead)

Business Lead

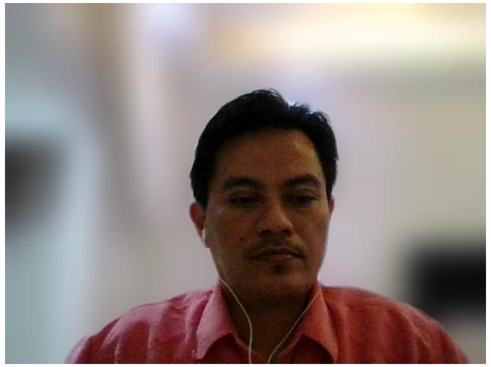
+public BusinessLead(String name)
+public boolean hasHeadCount()
+public boolean addReport(Accountant e, TechnicalLead supportTeam)
+public boolean requestBonus(Employee e, double bonus)
+public boolean approveBonus(Employee e, double bonus)

ENTITI / WORK FLOW	Class Diagram Relationships	Remarks
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 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

UML DIAGRAM C0MPANY STRCTURE

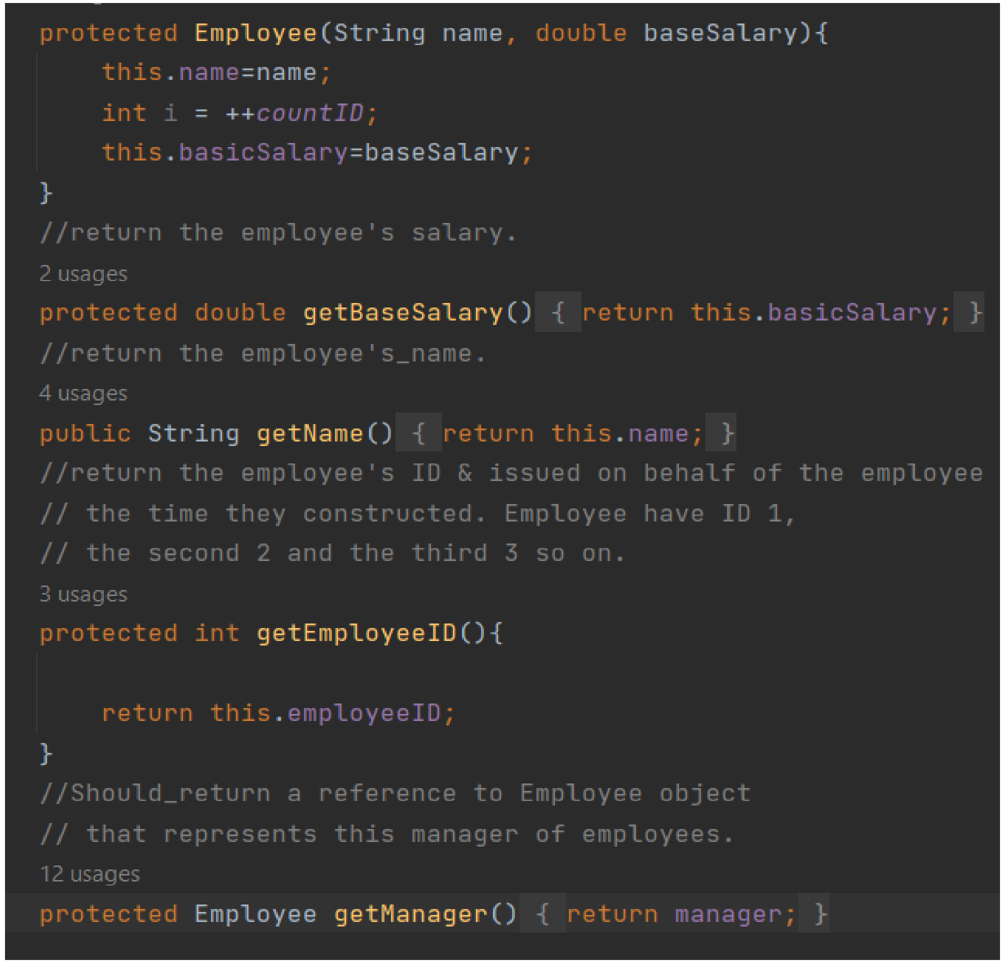

Accountant
+public Accountant(String name)
+public TechnicalLead getTeamSupported()
+public void supportTeam(TechnicalLead lead)
Composition
Aggregation
SALARY
BONUS

ENTITI / WORK FLOW	Class Diagram Relationships	Remarks
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



DEVELOPMENT AND RUNTIME
Class and Behavior

EMPLOYEE



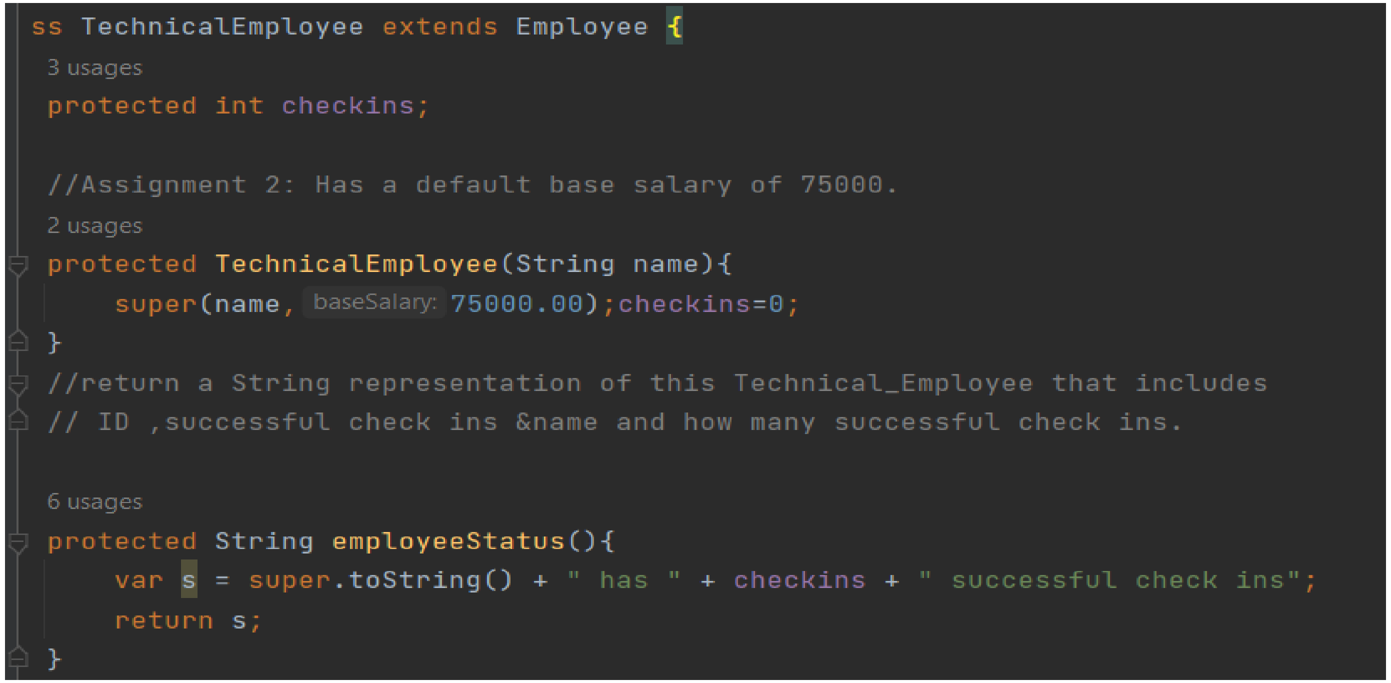
```
protected Employee(String name, double baseSalary){
    this.name=name;
    int i = ++countID;
    this.basicSalary=baseSalary;
}
//return the employee's salary.
2 usages
protected double getBaseSalary() { return this.basicSalary; }
//return the employee's_name.
4 usages
public String getName() { return this.name; }
//return the employee's ID & issued on behalf of the employee
// the time they constructed. Employee have ID 1,
// the second 2 and the third 3 so on.
3 usages
protected int getEmployeeID(){

    return this.employeeID;
}
//Should_return a reference to Employee object
// that represents this manager of employees.
12 usages
protected Employee getManager() { return manager; }
```

Method header	Description
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 otherwise
public String toString()	Should return a String representation o combination of their id followed by thei
public String employeeStatus()	Should return a String representation o status. This will be different for every subclass of Employee

DEVELOPMENT AND RUNTIME Class and Behavior

TECHNICAL EMPLOYEE

Method Header	Description
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"

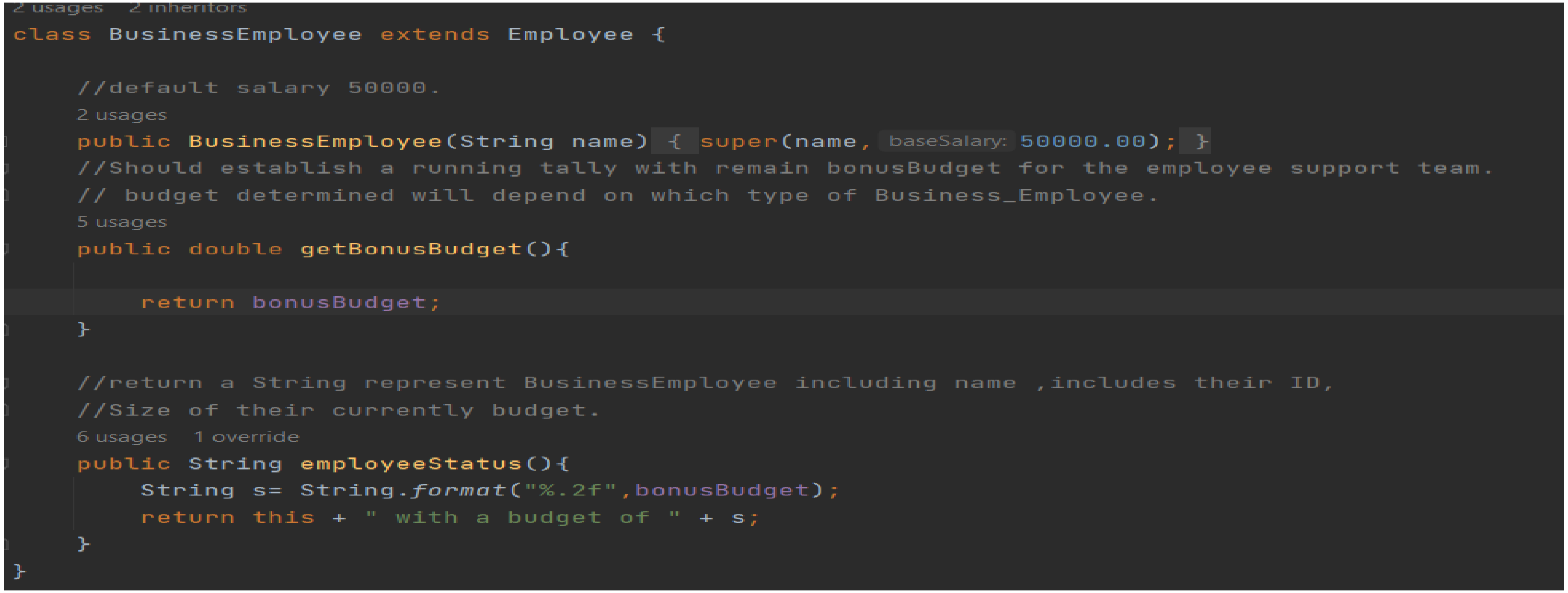


```
ss TechnicalEmployee extends Employee {
 3 usages
 protected int checkins;

 //Assignment 2: Has a default base salary of 75000.
 2 usages
 protected TechnicalEmployee(String name){
     super(name, baseSalary: 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.

 6 usages
 protected String employeeStatus(){
     var s = super.toString() + " has " + checkins + " successful check ins";
     return s;
 }
```

DEVELOPMENT AND RUNTIME
Class and Behavior

BUSINESS EMPLOYEE

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"

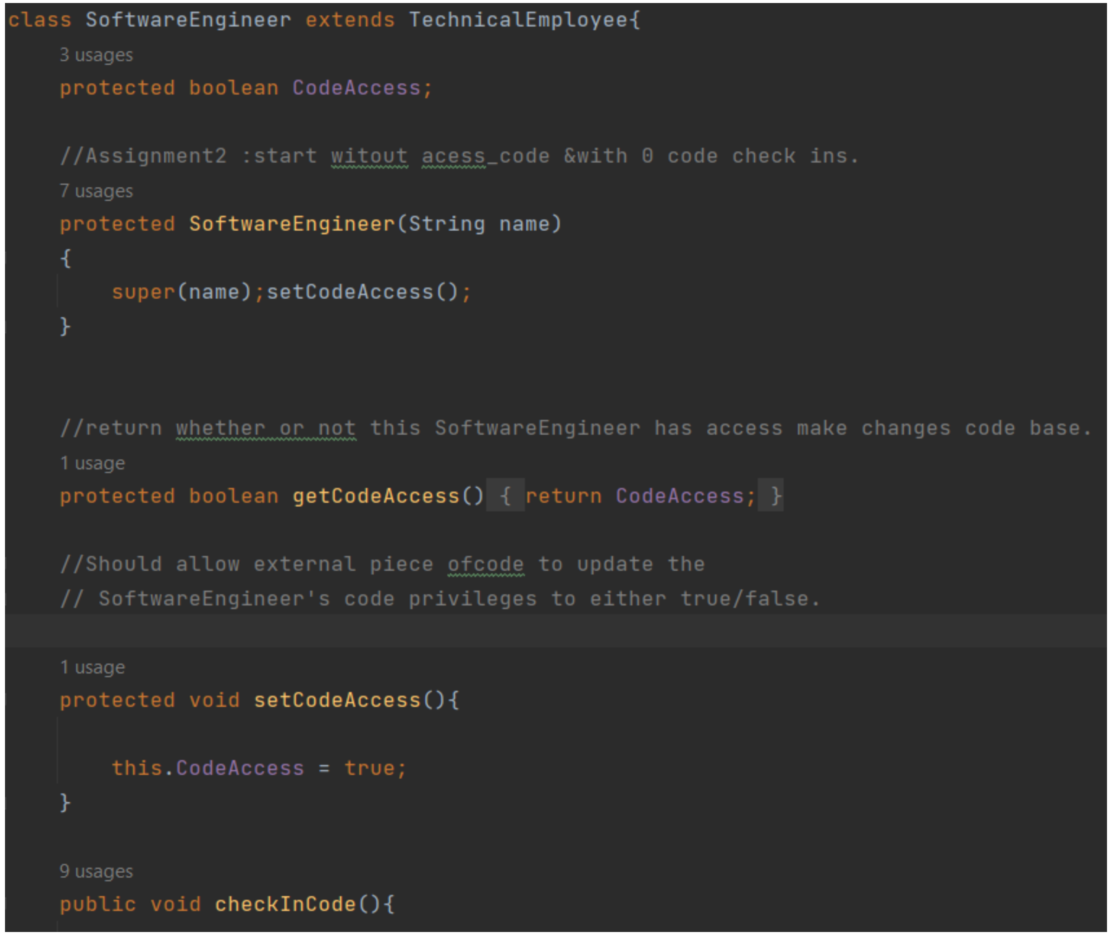


```
2 usages   2 inheritors
class BusinessEmployee extends Employee {

    //default salary 50000.
    2 usages
    public BusinessEmployee(String name) { super(name, baseSalary: 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.
    5 usages
    public double getBonusBudget(){

        return bonusBudget;
    }

    //return a String represent BusinessEmployee including name ,includes their ID,
    //Size of their currently budget.
    6 usages   1 override
    public String employeeStatus(){
        String s= String.format("%.2f",bonusBudget);
        return this + " with a budget of " + s;
    }
}
```

DEVELOPMENT AND RUNTIME
Class and Behavior

SOFTWARE ENGINEER

Method Header	Description
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"

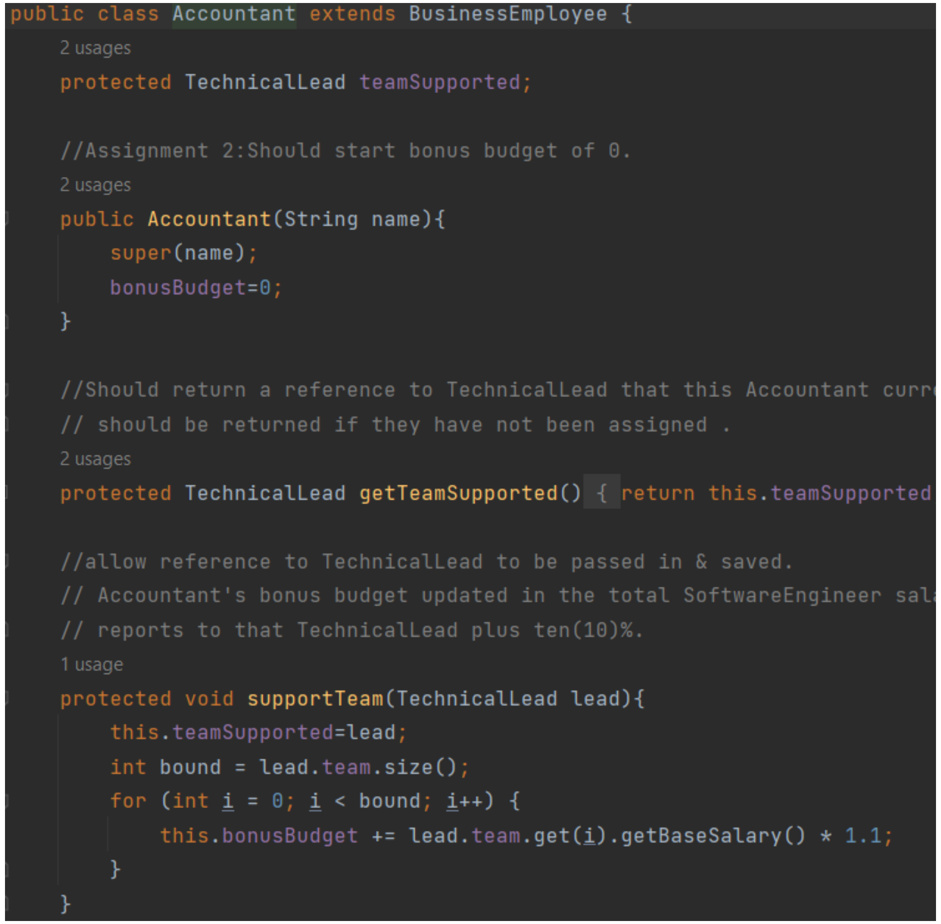


```
class SoftwareEngineer extends TechnicalEmployee{
    3 usages
    protected boolean CodeAccess;

    //Assignment2 :start witout acess_code &with 0 code check ins.
    7 usages
    protected SoftwareEngineer(String name)
    {
        super(name);setCodeAccess();
    }


    //return whether or not this SoftwareEngineer has access make changes code base.
    1 usage
    protected boolean getCodeAccess() { return CodeAccess; }

    //Should allow external piece ofcode to update the
    // SoftwareEngineer's code privileges to either true/false.

    1 usage
    protected void setCodeAccess(){

        this.CodeAccess = true;
    }

    9 usages
    public void checkInCode(){
```

DEVELOPMENT AND RUNTIME
Class and Behavior

ACCOUNTANT

Method Header	Description
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'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
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"

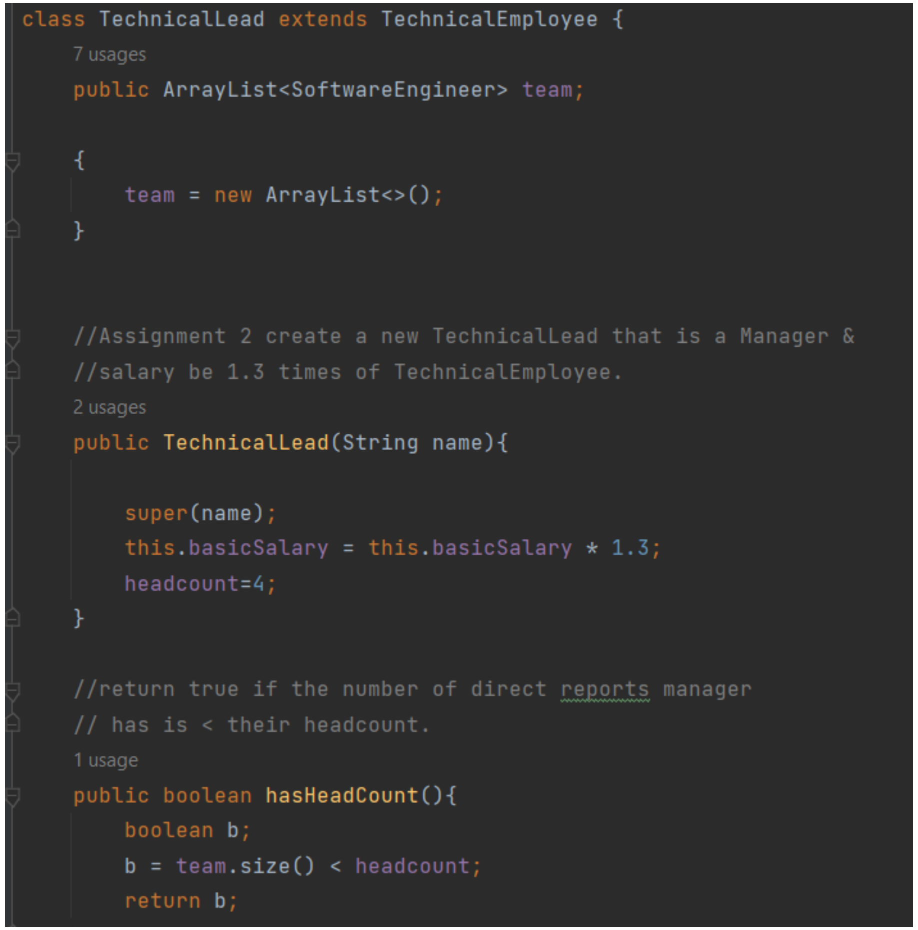


```
public class Accountant extends BusinessEmployee {
    2 usages
    protected TechnicalLead teamSupported;

    //Assignment 2:Should start bonus budget of 0.
    2 usages
    public Accountant(String name){
        super(name);
        bonusBudget=0;
    }

    //Should return a reference to TechnicalLead that this Accountant curr
    // should be returned if they have not been assigned .
    2 usages
    protected TechnicalLead getTeamSupported() { return this.teamSupported

    //allow reference to TechnicalLead to be passed in & saved.
    // Accountant's bonus budget updated in the total SoftwareEngineer sal
    // reports to that TechnicalLead plus ten(10)%.
    1 usage
    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;
        }
    }
```

DEVELOPMENT AND RUNTIME
Class and Behavior

TECHNICAL LEAD

Method Header	Description
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.
public boolean hasHeadCount()	Should return true if the number of direct reports this manager has is less than their headcount.
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
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
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"

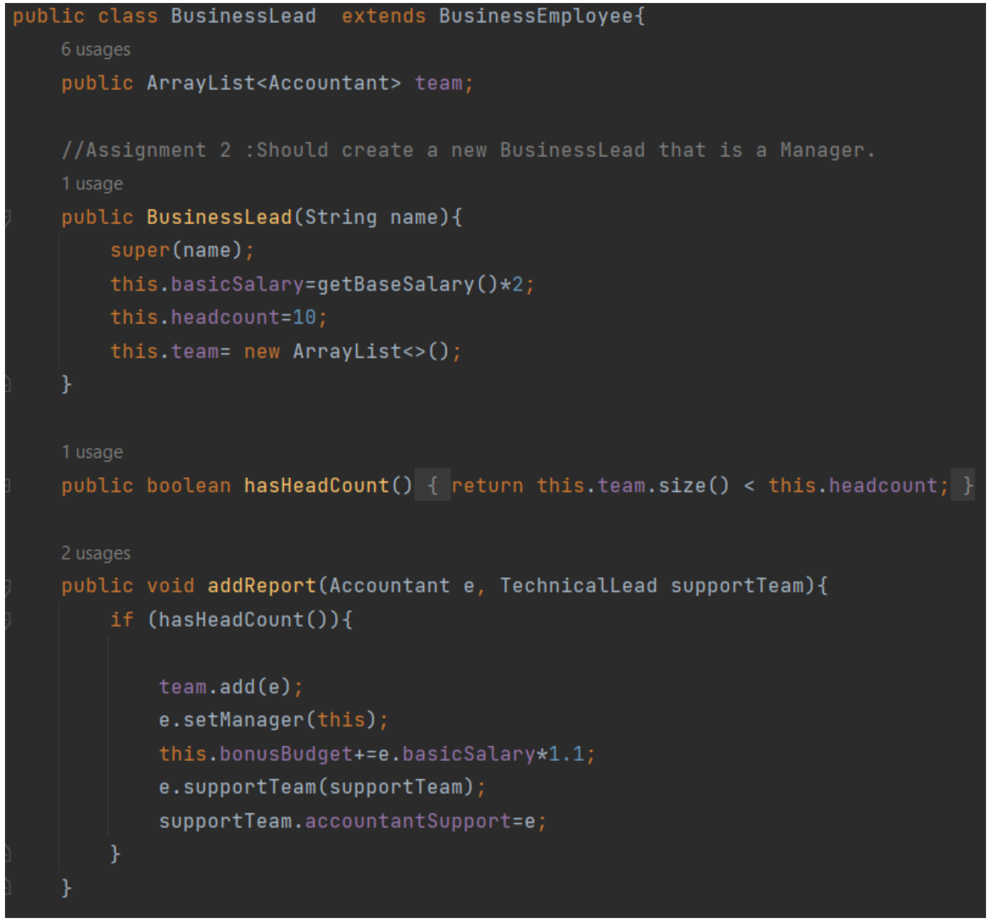


```
class TechnicalLead extends TechnicalEmployee {
    7 usages
    public ArrayList<SoftwareEngineer> team;

    {
        team = new ArrayList<>();
    }


    //Assignment 2 create a new TechnicalLead that is a Manager &
    //salary be 1.3 times of TechnicalEmployee.
    2 usages
    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.
    1 usage
    public boolean hasHeadCount(){
        boolean b;
        b = team.size() < headcount;
        return b;
```

DEVELOPMENT AND RUNTIME
Class and Behavior

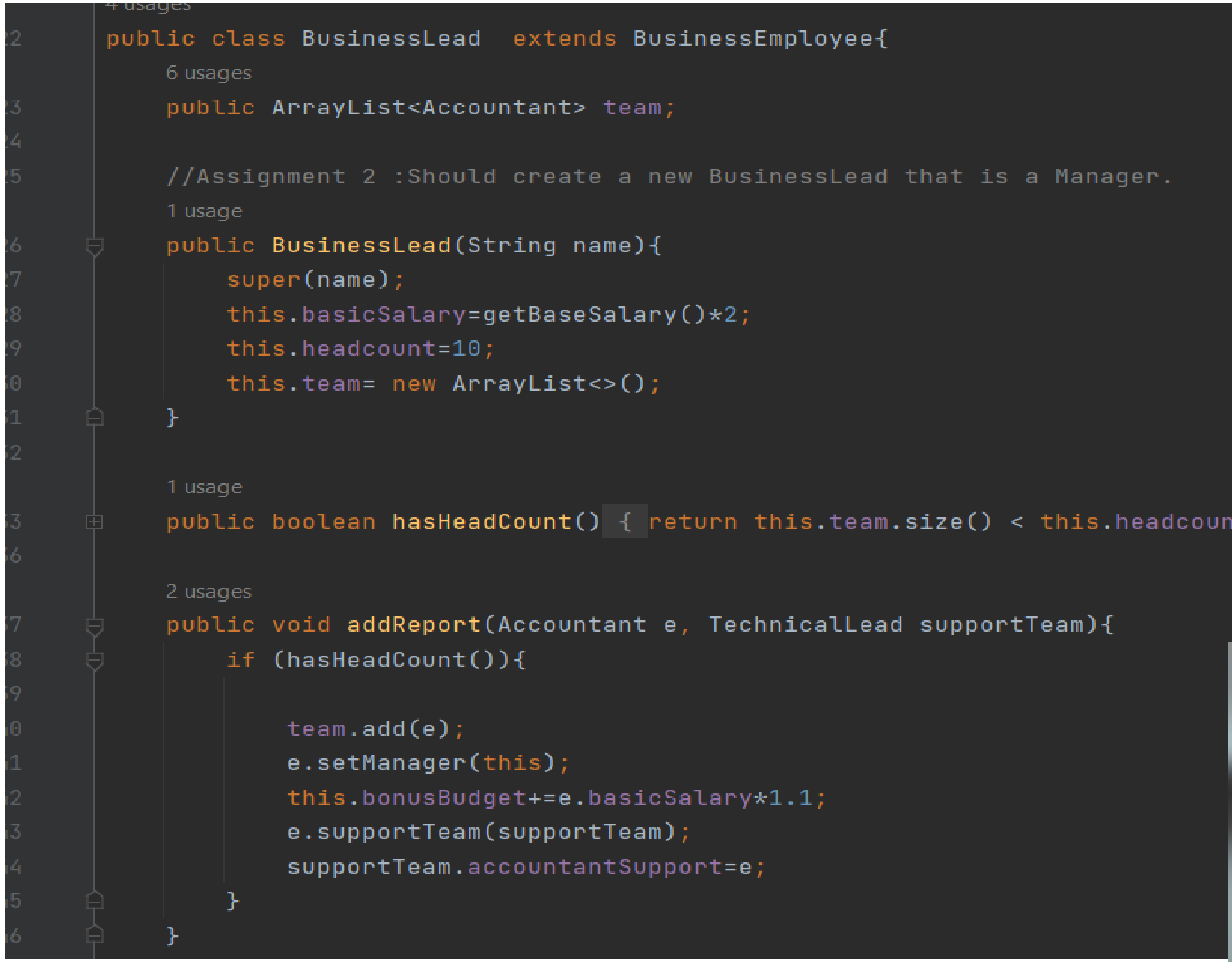
BUSINESS LEAD

Method Header	Description
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



```
public class BusinessLead  extends BusinessEmployee{
    6 usages
    public ArrayList<Accountant> team;

    //Assignment 2 :Should create a new BusinessLead that is a Manager.
    1 usage
    public BusinessLead(String name){
        super(name);
        this.basicSalary=getBaseSalary()*2;
        this.headcount=10;
        this.team= new ArrayList<>();
    }

    1 usage
    public boolean hasHeadCount() { return this.team.size() < this.headcount; }

    2 usages
    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;
        }
    }
```

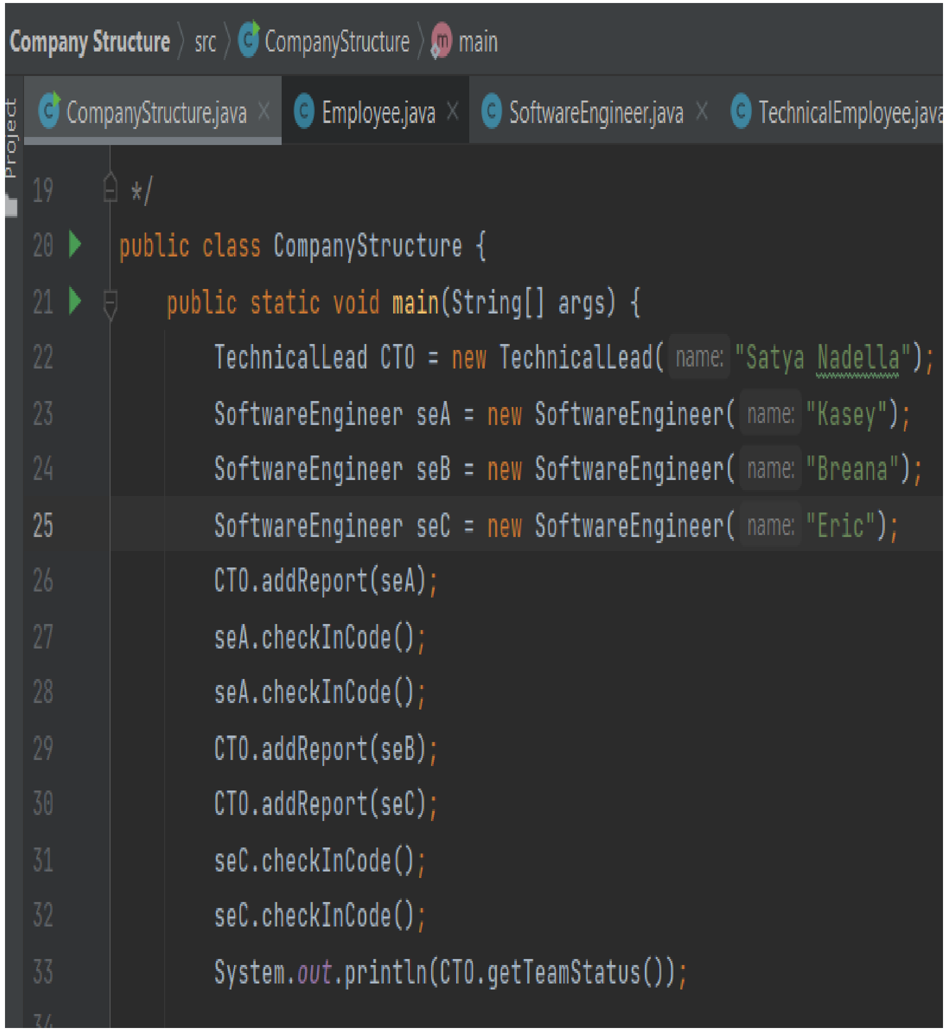
DEVELOPMENT AND RUNTIME



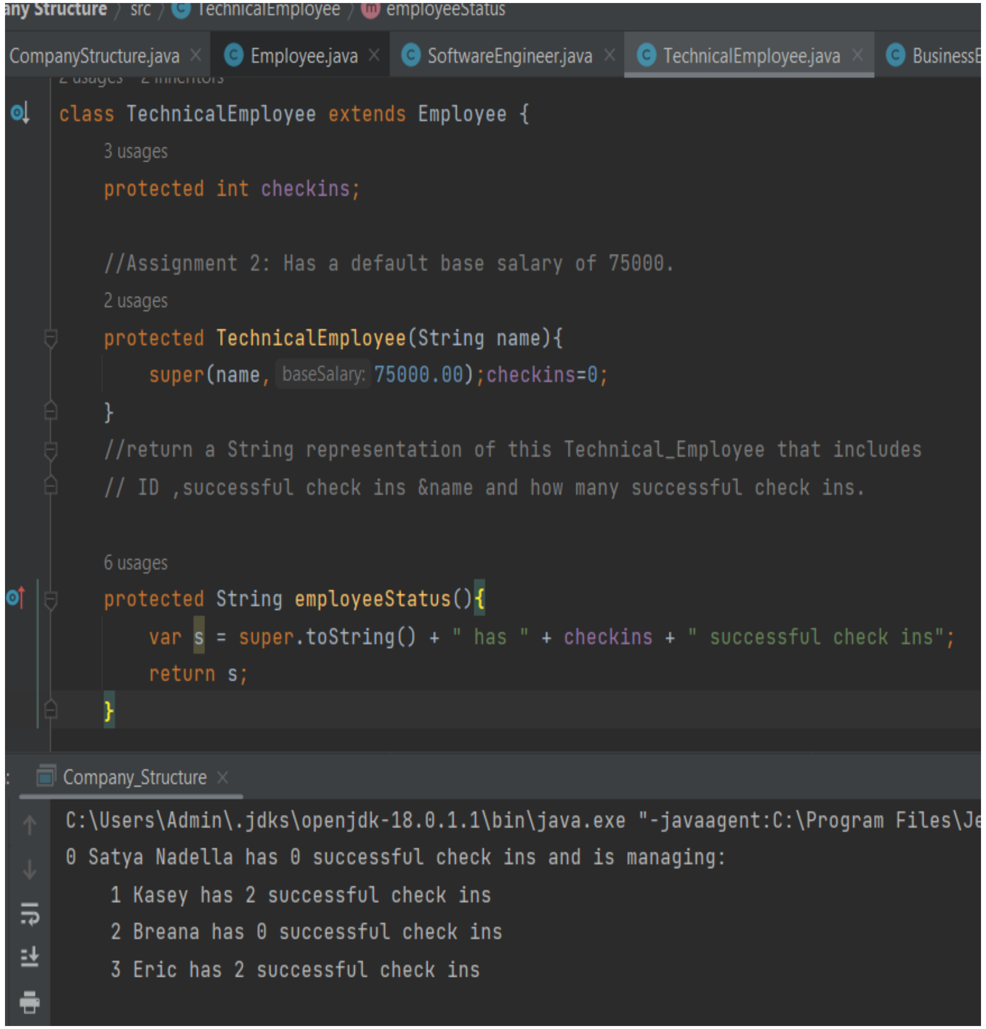
```
4 usages
public class BusinessLead  extends BusinessEmployee{
    6 usages
    public ArrayList<Accountant> team;

    //Assignment 2 :Should create a new BusinessLead that is a Manager.
    1 usage
    public BusinessLead(String name){
        super(name);
        this.basicSalary=getBaseSalary()*2;
        this.headcount=10;
        this.team= new ArrayList<>();
    }

    1 usage
    public boolean hasHeadCount() { return this.team.size() < this.headcount; }

    2 usages
    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;
        }
    }
```

The coding has been created to guarantee that it will perform in accordance with its intended function and has been optimised to execute rapidly.

DEVELOPMENT AND RUNTIME



```
*/
public class CompanyStructure {
    public static void main(String[] args) {
        TechnicalLead CTO = new TechnicalLead( name: "Satya Nadella");
        SoftwareEngineer seA = new SoftwareEngineer( name: "Kasey");
        SoftwareEngineer seB = new SoftwareEngineer( name: "Breana");
        SoftwareEngineer seC = new SoftwareEngineer( name: "Eric");
        CTO.addReport(seA);
        seA.checkInCode();
        seA.checkInCode();
        CTO.addReport(seB);
        CTO.addReport(seC);
        seC.checkInCode();
        seC.checkInCode();
        System.out.println(CTO.getTeamStatus());
```

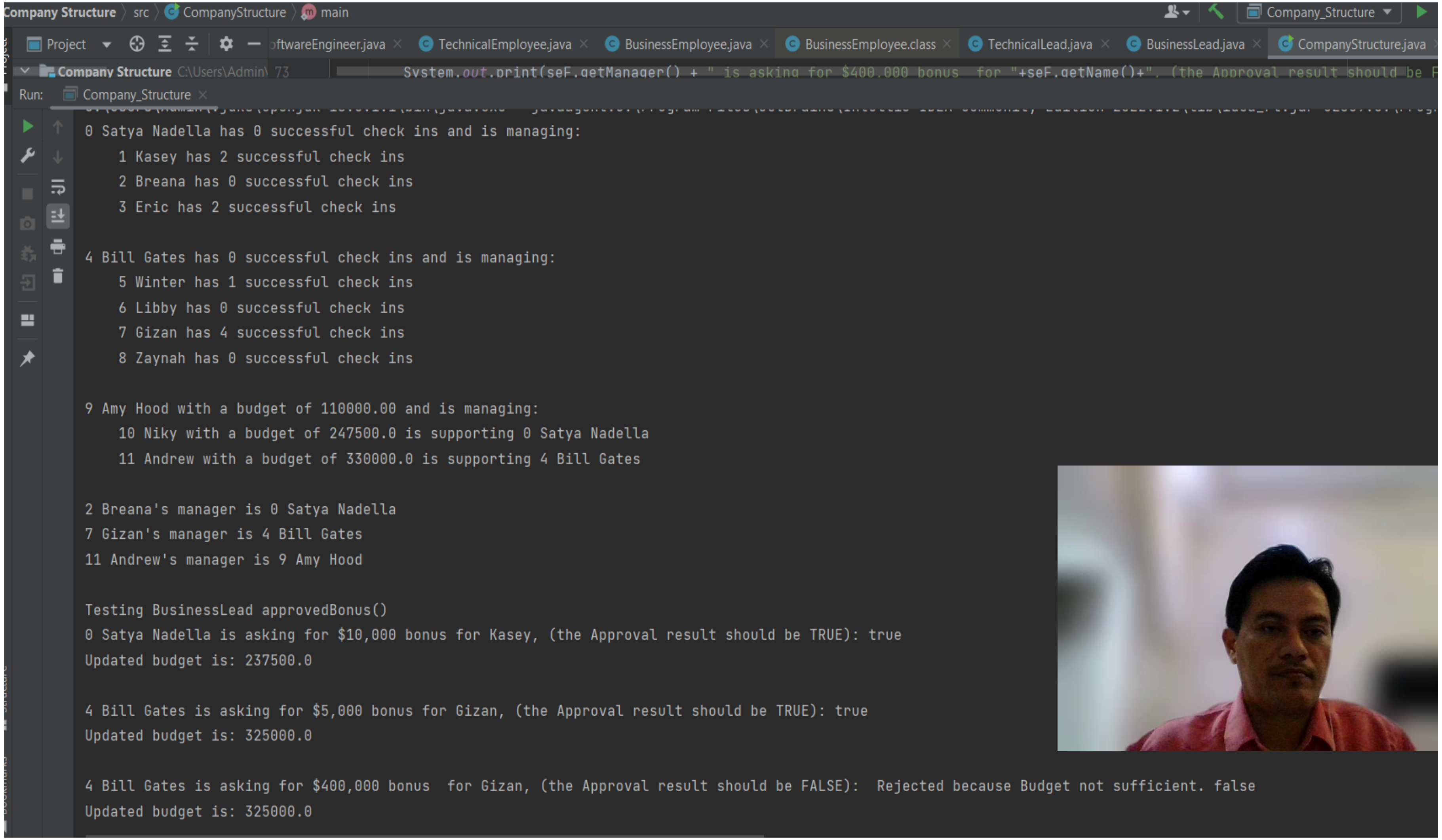


```
2 usages   2 inheritors
class TechnicalEmployee extends Employee {
    3 usages
    protected int checkins;

    //Assignment 2: Has a default base salary of 75000.
    2 usages
    protected TechnicalEmployee(String name){
        super(name, baseSalary: 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.

    6 usages
    protected String employeeStatus(){
        var s = super.toString() + " has " + checkins + " successful check ins";
        return s;
    }
```

```
C:\Users\Admin\.jdks\openjdk-18.0.1.1\bin\java.exe "-javaagent:C:\Program Files\Je
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
```

Each command and its syntax have undergone testing and are error-free.

DEVELOPMENT AND RUNTIME

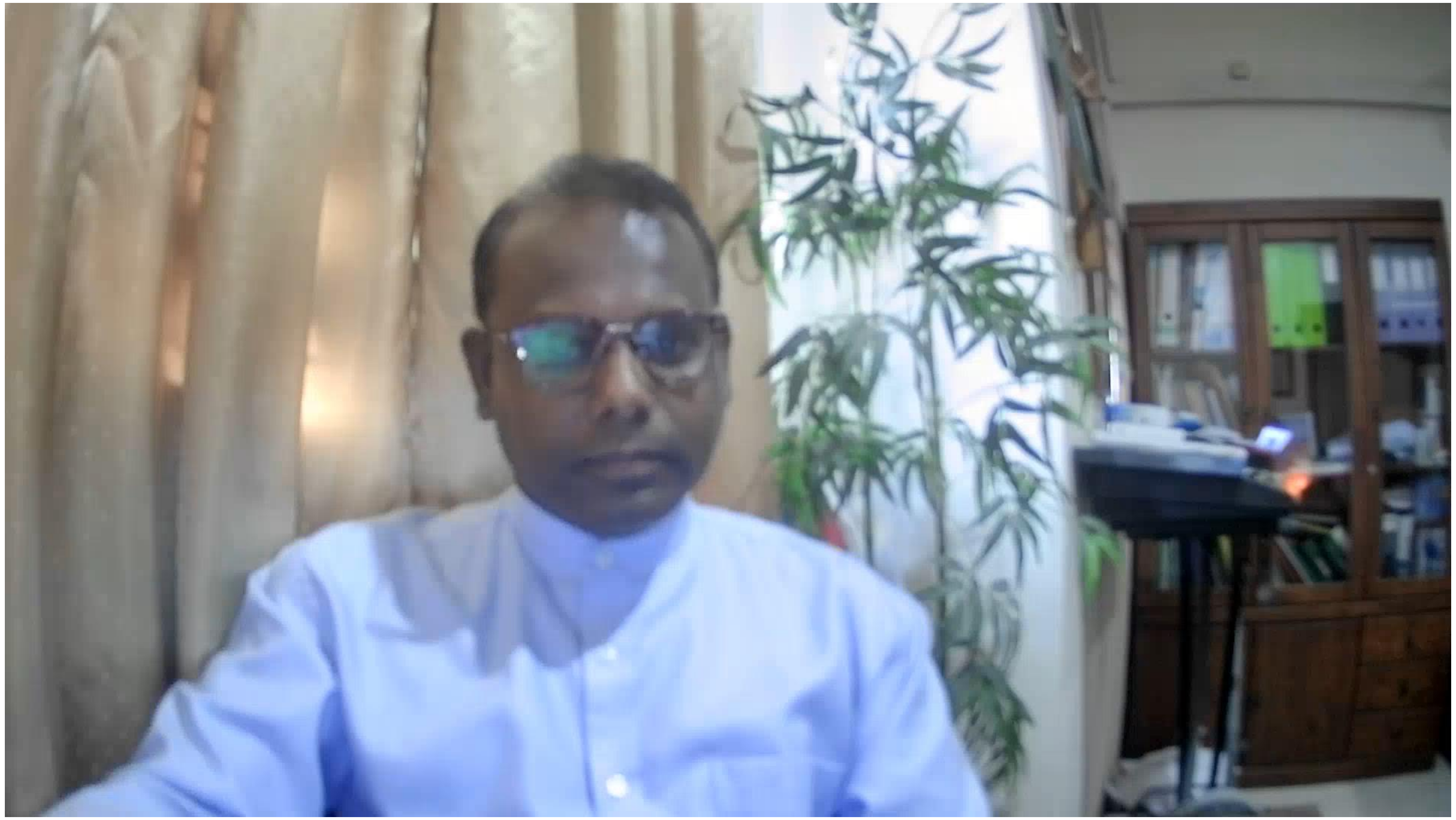


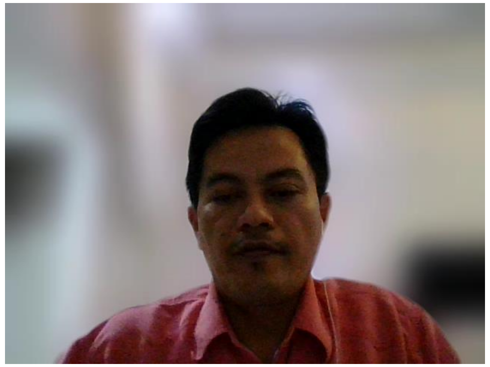
```
System.out.print(seF.getManager() + " is asking for $400,000 bonus  for "+seF.getName()+", (the Approval result should be F
```

```
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
```

The end result is the project output. These are the indicators that all the code functions correctly and successfully.

FINAL PRESENTATION





UNITAR

UNITAR

2022

INSTRUCTOR: Dr. Simon Lau

ITWM5113 SOFTWARE DESIGN & DEVELOPMENT

28 JUL 2022

GROUP ASSIGNMENT 3

THANK YOU!

A.HALIM BIN AMINNUDIN | MOHD SOUFI BIN YUSSOF | DEVAN

Matrix# : MC220517284 | UNU2200768 | UNU2200575