1. Detailed Class Diagram

**class WFLUManager**<T extends Comparable>

**WFLUManager Data;**

protected ArrayList<T> genericList = new ArrayList<T>();

**WFLUManager Functions;**

public WLFUManager(){}

public int Add(T item);

public T Get(int i){}

public void Sort()

abstract **class Employees**

**Employees Data;**

**Employees Functions;**

public abstract int GetSpecialInfo()

public abstract void UpdateSpecialInfo(int x)

public abstract void calcSlary(float basepay, float COLA, float bonus, float totalSalary)

**class The\_WLFU\_Team** extends Employees

**The\_WLFU\_Team Data;**

protected int specialInfo

**The\_WLFU\_Team Functions:**

public int GetSpecialInfo()

public void UpdateSpeicalInfo(int x)

public void calcSalary(float basePay, float COLA, float bonus, float totalSalary)

**class RegionalCitiesBoston** extends The\_WLFU\_Team implements Comparable

**RegionalCitiesBoston Data;**

**RegionalCitiesBoston Functions;**

public int compareTo(Object o)

**class BostonAdministrative** extends RegionalCitiesBoston

**BostonAdministrative Data;**

protected String employeeName;

protected String employeeSSN;

protected String employeeRace;

protected int specialEmployee;

protected String jobTitle;

protected float basePay;

protected float bonus;

protected float COLA;

protected float totalSalary;

**BostonAdministrative Functions;**

public String getJobTitle()

public float getBasePay()

public float getBonus()

public float getCOLA()

public float getTotalSalary()

public BostonAdministrative(String employeeName, String employeeSSN, String employeeRace, int specialEmployee)

public String toString()

**class BostonDoctor** extends RegionalCitiesBoston

**BostonDoctor Data;**

protected String employeeName;

protected String employeeSSN;

protecred String employeeRace;

protected int specialEmployee;

protected String jobTitle;

protected float basePay;

protected float bonus;

protected float COLA;

protected float totalSalary;

**BostonDoctor Functions;**

public String getJobTitle()

public float getBasePay()

public float getBonus()

public float getCOLA()

public float getTotalSalary()

public BostonDoctor(String employeeName, String employeeSSN, String employeeRace, int specialEmployee)

public String toString()

**class BostonNurse** extends RegionalCitiesBoston

**BostonNurse Data;**

protected String employeeName;

protected String employeeSSN;

protected String employeeRace;

protected int sepcialEmployee;

protected String jobTitle;

protected float basePay;

protected float bonus;

protected float COLA;

protected float totalSalary;

**BostonNurse Functions**

public String getJobTitle()

public float getBasePay()

public float getBonus()

public float getCOLA()

public float getTotalSalary()

public BostonNurse(String employeeName, String employeeSSN, String employeeRace, int specialEmployee)

public String toString()

**class BostonMedicalSupport** extends RegionalCitiesBoston

**BostonMedicalSupport Data;**

protected String employeeName;

protected String employeeSSN;

protected String employeeRace;

protected int specialEmployee;

protected String jobTitle;

protected float basePay;

protected float bonus;

protected float COLA;

protected float totalSalary;

**BostonMedicalSupport Functions;**

public String getJobTitle()

public float getBasePay()

public float getBonus()

public float getCOLA()

public float getTotalSalary()

public BostonMedicalSupport(String employeeName, String employeeSSN, String employeeRace, int specialEmployee)

public String toString()

1. Code a mainline to create objects, store them into an array, calculate the salary and taxes for each individual , sort them based on salary.
2. public static void main(String[] args){  
    WLFUManager<RegionalCitiesBoston> bostonEmployees = new WLFUManager<RegionalCitiesBoston>();//creating a new Boston Array List  
     
    BostonDoctor doc1 = new BostonDoctor("IM Bones","455657890","AA",100);  
    //creating a new object of BostonDoctor  
    doc1.calcSalary(doc1.getBasePay(),doc1.getCOLA(),doc1.getBonus(),doc1.getTotalSalary()); //using the calcSalary function  
    bostonEmployees.Add(doc1); //adding employee to ArrayList  
     
    BostonNurse nurse1 = new BostonNurse("UR Temp","789302345","CA",3);  
    //creating a new object of BostonNurse  
    nurse1.calcSalary(nurse1.getBasePay(),nurse1.getCOLA(),nurse1.getBonus(),nurse1.getTotalSalary()); //using the calcSalary function  
    bostonEmployees.Add(nurse1); //adding employee to ArrayList  
     
    BostonDoctor doc2 = new BostonDoctor("DVM Frakes","786456712","CA",120);  
    //creating a new object of BostonDoctor  
    doc2.calcSalary(doc2.getBasePay(),doc2.getCOLA(),doc2.getBonus(),doc2.getTotalSalary()); //using the calcSalary function  
    bostonEmployees.Add(doc2); //adding employee to ArrayList  
      
    BostonAdministrative admin1 = new BostonAdministrative("IM Boss","543126787","HS", 1);  
    //creating a new object of BostonAdministrative  
    admin1.calcSalary(admin1.getBasePay(),admin1.getCOLA(),admin1.getBonus(),admin1.getTotalSalary()); //using the calcSalary function  
    bostonEmployees.Add(admin1); //adding employee to ArrayList  
     
    for(int i =0; i < bostonEmployees.size(); i++){ //for loop is used to iterate through the ArrayList and print out each object  
    System.*out*.println(bostonEmployees.Get(i));  
    System.*out*.println("");  
    }//end of for loop  
   }//end of main method

3. Generic Manager

class WLFUManager<T extends Comparable>{  
 protected ArrayList<T> genericList = new ArrayList<T>(); //creating a generic ArrayList  
 public WLFUManager(){} //WFLUManager Constructor  
 public int Add(T item){ //generic add item method that will add a new object to array list and return the size of array list  
 genericList.add(item);  
 return genericList.size();  
 }  
  
 public T Get(int i){return genericList.get(i);} //a generic get method that will return the object from position i  
  
 public int size(){return genericList.size();}  
  
 public void Sort(){//a generic sort method  
 //we are sorting from smallest salary to largest salary  
 T saveItem ,saveItem2; //saveItem and saveItem2 are new objects that will be used to help swap objects in the array  
  
 int whileCondition = 1; //whileCondition is an int used to end the loop  
 while(whileCondition == 1){  
 whileCondition = 0;  
 for(int i=0; i<genericList.size()-1; i++){  
 switch (genericList.get(i).compareTo(genericList.get(i+1))){  
 case 1: //the objects are out of order and must be changed  
 saveItem=genericList.get(i);  
 saveItem2=genericList.get(i+1);  
 genericList.remove(i);  
 genericList.add(i,saveItem2);  
 genericList.remove(i+1);  
 genericList.add(i+1,saveItem);  
 whileCondition = 1;  
 case -1: //the objects are in the right order  
 break;  
 default: //objects are equal or no change  
 }//end of switch  
 }//end of for loop  
 }//end of while loop  
 }//end of Sort method  
}//end of WLFUManager class

5. program your manager and run the code

import java.util.ArrayList; //necessary to have ArrayList in code  
  
/\*The following program is designed to be able to delete, add, and manage employees of the We Live For You (WLFU) Medical Corporation across its  
\* four different locations in America. The locations of the four hospitals are Boston Massachusetts, St. Joseph Missouri, Los Angeles California, and Dallas Texas  
\* Employees can have an occupation of Administrative, Doctors, Nurses, or Medical Support.  
\* There is a generic class manager named WLFUManager that will allow the employees to be added to an ArrayList and sorted by their total salaries from smallest to largest\*/  
class WLFUManager<T extends Comparable>{  
 protected ArrayList<T> genericList = new ArrayList<T>(); //creating a generic ArrayList  
 public WLFUManager(){} //WFLUManager Constructor  
 public int Add(T item){ //generic add item method that will add a new object to array list and return the size of array list  
 genericList.add(item);  
 return genericList.size();  
 }  
  
 public T Get(int i){return genericList.get(i);} //a generic get method that will return the object from position i  
  
 public int size(){return genericList.size();}  
  
 public void Sort(){//a generic sort method  
 //we are sorting from smallest salary to largest salary  
 T saveItem ,saveItem2; //saveItem and saveItem2 are new objects that will be used to help swap objects in the array  
  
 int whileCondition = 1; //whileCondition is an int used to end the loop  
 while(whileCondition == 1){  
 whileCondition = 0;  
 for(int i=0; i<genericList.size()-1; i++){  
 switch (genericList.get(i).compareTo(genericList.get(i+1))){  
 case 1: //the objects are out of order and must be changed  
 saveItem=genericList.get(i);  
 saveItem2=genericList.get(i+1);  
 genericList.remove(i);  
 genericList.add(i,saveItem2);  
 genericList.remove(i+1);  
 genericList.add(i+1,saveItem);  
 whileCondition = 1;  
 case -1: //the objects are in the right order  
 break;  
 default: //objects are equal or no change  
 }//end of switch  
 }//end of for loop  
 }//end of while loop  
 }//end of Sort method  
}//end of WLFUManager class  
  
abstract class Employees{  
 public abstract int GetSpecialInfo();  
 public abstract void UpdateSpecialInfo(int x);  
 public abstract void calcSalary(float basePay, float COLA, float bonus, float totalSalary);  
}  
  
class The\_WLFU\_Team extends Employees{  
 protected int specialInfo;  
 @Override  
 public int GetSpecialInfo() { //a method to return an employees special info  
 return specialInfo;  
 }  
 @Override  
 public void UpdateSpecialInfo(int x) { // a method to update an employees special info  
 specialInfo = x;  
 }  
 @Override  
 public void calcSalary(float basePay, float COLA, float bonus, float totalSalary) {//a method to calculate salary of an employee  
 totalSalary = basePay+COLA+bonus;  
 }  
}  
  
class RegionalCitiesBoston extends The\_WLFU\_Team implements Comparable { //the class for the city of Boston  
 @Override  
 public int compareTo(Object o) { //override for comparable so that we may sort  
 return 0;  
 }  
} //end of RegionalCitiesBoston  
  
class BostonAdministrative extends RegionalCitiesBoston{ //a class for Administrative employees of Boston  
 protected String employeeName;//a String for the employee name  
 protected String employeeSSN;//a String which holds the employee's social security number  
 protected String employeeRace;//a String which holds the employee's race  
 protected int specialEmployee; //an integer to hold special info of the employee  
 //for Administrative, int specialEmployee has the following values (1-Senior Executive, 2-Junior Executive, 3-Support)  
 protected String jobTitle; //a string to hold the job title of administrative  
 protected float basePay;//a float which holds the base pay of administrative  
 protected float bonus;//creating a bonus variable the holds the employee's bonus  
 protected float COLA;//a float to hold COLA of boston  
 protected float totalSalary;  
  
 //Getter methods for jobTitle, basePay, bonus, COLA, and totalSalary  
 public String getJobTitle() {  
 return jobTitle;  
 }  
 public float getBasePay() {  
 return basePay;  
 }  
 public float getBonus() {  
 return bonus;  
 }  
 public float getCOLA() {  
 return COLA;  
 }  
 public float getTotalSalary() {  
 return totalSalary;  
 }  
  
 public BostonAdministrative(String employeeName, String employeeSSN, String employeeRace, int specialEmployee){ //constructor for BostonAdministrative  
 this.employeeName = employeeName;  
 this.employeeSSN = employeeSSN;  
 this.employeeRace = employeeRace;  
 this.specialEmployee = specialEmployee;  
  
 //if else chain to assign jobTitle  
 if(specialEmployee==1){  
 jobTitle="Senior Executive";  
 }else if(specialEmployee==2){  
 jobTitle="Junior Executive";  
 }else{  
 jobTitle="Support";  
 }  
  
 //if else chain will determine the employees base pay  
 if(specialEmployee==1){  
 basePay = 400000;  
 }else if(specialEmployee==2){  
 basePay = 175000;  
 }else{  
 basePay = 40000;  
 }  
 this.basePay = basePay;  
  
 //if else statement calculates bonus  
 if(specialEmployee==1){  
 bonus = (float)(basePay\*.2);  
 }else if(specialEmployee==2){  
 bonus = (float)(basePay\*.1);  
 }else{  
 bonus = 0;  
 }  
  
 COLA = (float)(0.15\*basePay);//calculating cola of boston  
 } //constructor for BostonAdministrative Class  
  
 public String toString(){ //changing the format for when we want to print out an object  
 return employeeName+"\n"+"Resident Boston Hospital\n"+"Race: "+employeeRace+"\nSSN: "+employeeSSN+"\nSpecial Integer: "+specialEmployee+  
 "\nJob Title: "+jobTitle+"\nBase Salary: $"+basePay  
 +"\nCOLA: $"+COLA+"\nSpecial Bonus: $"+bonus;  
 }  
} //end of BostonAdministrative  
  
class BostonDoctor extends RegionalCitiesBoston{ //a class for Doctors of Boston  
 protected String employeeName;//a String for the employee name  
 protected String employeeSSN;//a String which holds the employee's social security number  
 protected String employeeRace;//a String which holds the employee's race  
 protected int specialEmployee; //an integer to hold special info of the employee  
 //specialEmployee in class BostonDoctor represents the number of patients a doctor has  
 protected String jobTitle; //a string to hold the job title of administrative  
 protected float basePay;//a float which holds the base pay of administrative  
 protected float bonus;//creating a bonus variable the holds the employee's bonus  
 protected float COLA;//a float to hold COLA of boston  
 protected float totalSalary;  
  
 //Getter methods for jobTitle, basePay, bonus, COLA, and totalSalary  
 public String getJobTitle() {  
 return jobTitle;  
 }  
 public float getBasePay() {  
 return basePay;  
 }  
 public float getBonus() {  
 return bonus;  
 }  
 public float getCOLA() {  
 return COLA;  
 }  
 public float getTotalSalary() {  
 return totalSalary;  
 }  
  
 public BostonDoctor(String employeeName, String employeeSSN, String employeeRace, int specialEmployee) { //constructor for BostonAdministrative  
 this.employeeName = employeeName;  
 this.employeeSSN = employeeSSN;  
 this.employeeRace = employeeRace;  
 this.specialEmployee = specialEmployee;  
  
 basePay = 155000;//the base pay of all doctors  
 bonus = (float)(basePay\*0.0025\*specialEmployee);  
 COLA = (float)(0.15\*basePay);//calculating cola of boston  
 jobTitle = "Doctor";  
 }//end of constructor  
 public void calculateSalary(){}  
 public String toString(){ //changing the format for when we want to print out an object  
 return employeeName+"\n"+"Resident Boston Hospital\n"+"Race: "+employeeRace+"\nSSN: "+employeeSSN+"\nSpecial Integer: "+specialEmployee+  
 "\nJob Title: "+jobTitle+"\nBase Salary: $"+basePay  
 +"\nCOLA: $"+COLA+"\nSpecial Bonus: $"+bonus;  
 }  
} //end of BostonDoctor  
  
class BostonNurse extends RegionalCitiesBoston{ //a class for Nurses of Boston  
 protected String employeeName;//a String for the employee name  
 protected String employeeSSN;//a String which holds the employee's social security number  
 protected String employeeRace;//a String which holds the employee's race  
 protected int specialEmployee; //an integer to hold special info of the employee  
 //specialEmployee in class BostonNurse represents the type of nurse they are (1-Clinic nurse, 2-Hospital Floor nurse, 3-Hospital Administrative nurse)  
 protected String jobTitle; //a string to hold the job title of administrative  
 protected float basePay;//a float which holds the base pay of administrative  
 protected float bonus;//creating a bonus variable the holds the employee's bonus  
 protected float COLA;//a float to hold COLA of boston  
 protected float totalSalary;  
  
 //Getter methods for jobTitle, basePay, bonus, COLA, and totalSalary  
 public String getJobTitle() {  
 return jobTitle;  
 }  
 public float getBasePay() {  
 return basePay;  
 }  
 public float getBonus() {  
 return bonus;  
 }  
 public float getCOLA() {  
 return COLA;  
 }  
 public float getTotalSalary() {  
 return totalSalary;  
 }  
  
 public BostonNurse(String employeeName, String employeeSSN, String employeeRace, int specialEmployee) { //constructor for BostonAdministrative  
 this.employeeName = employeeName;  
 this.employeeSSN = employeeSSN;  
 this.employeeRace = employeeRace;  
 this.specialEmployee = specialEmployee;  
 basePay = 65000; //the base pay for all nurses  
  
 //if else chain calculates the nurse bonus  
 if(specialEmployee==1){  
 bonus=(float)(basePay\*.1);  
 }else if(specialEmployee==2){  
 bonus=(float)(basePay\*.15);  
 }else{  
 bonus=(float)(basePay\*.2);  
 }  
  
 //if else chain to assign jobTitle  
 if(specialEmployee==1){  
 jobTitle="Clinic Nurse";  
 }else if(specialEmployee==2){  
 jobTitle="Hospital Floor Nurse";  
 }else{  
 jobTitle="Hospital Administrative Nurse";  
 }  
 COLA = (float)(0.15\*basePay);//calculating cola of boston  
 }//end of boston nurse constructor  
 public String toString(){ //changing the format for when we want to print out an object  
 return employeeName+"\n"+"Resident Boston Hospital\n"+"Race: "+employeeRace+"\nSSN: "+employeeSSN+"\nSpecial Integer: "+specialEmployee+  
 "\nJob Title: "+jobTitle+"\nBase Salary: $"+basePay  
 +"\nCOLA: $"+COLA+"\nSpecial Bonus: $"+bonus;  
 }  
}//end of boston nurse class  
  
class BostonMedicalSupport extends RegionalCitiesBoston{ //a class for Medical Support of Boston  
 protected String employeeName;//a String for the employee name  
 protected String employeeSSN;//a String which holds the employee's social security number  
 protected String employeeRace;//a String which holds the employee's race  
 protected int specialEmployee; //an integer to hold special info of the employee  
 //specialEmployee in class BostonMedicalSupport represents the type of medical support they are (1-basePay = 45000, 2-basePay = 35000)  
 protected String jobTitle; //a string to hold the job title of administrative  
 protected float basePay;//a float which holds the base pay of administrative  
 protected float bonus;//creating a bonus variable the holds the employee's bonus  
 protected float COLA;//a float to hold COLA of boston  
 protected float totalSalary;  
  
 //Getter methods for jobTitle, basePay, bonus, COLA, and totalSalary  
 public String getJobTitle() {  
 return jobTitle;  
 }  
 public float getBasePay() {  
 return basePay;  
 }  
 public float getBonus() {  
 return bonus;  
 }  
 public float getCOLA() {  
 return COLA;  
 }  
 public float getTotalSalary() {  
 return totalSalary;  
 }  
  
 public BostonMedicalSupport(String employeeName, String employeeSSN, String employeeRace, int specialEmployee) { //constructor for BostonAdministrative  
 this.employeeName = employeeName;  
 this.employeeSSN = employeeSSN;  
 this.employeeRace = employeeRace;  
 this.specialEmployee = specialEmployee;  
  
 //if else chain to determine payRate dependent on specialEmployee value  
 if(specialEmployee==1){  
 basePay = 45000;  
 }else{  
 basePay = 35000;  
 }  
  
 //if else chain to assign jobTitle  
 if(specialEmployee==1){  
 jobTitle="Support Type 1";  
 }else{  
 jobTitle="Support Type 2";  
 }  
 COLA = (float)(0.15\*basePay);//calculating cola of boston  
 }//end of BostonMedicalSupport constructor  
 public String toString(){ //changing the format for when we want to print out an object  
 return employeeName+"\n"+"Resident Boston Hospital\n"+"Race: "+employeeRace+"\nSSN: "+employeeSSN+"\nSpecial Integer: "+specialEmployee+  
 "\nJob Title: "+jobTitle+"\nBase Salary: $"+basePay  
 +"\nCOLA: $"+COLA+"\nSpecial Bonus: $"+bonus;  
 }  
}//end of BostonMedicalSupport class  
  
public class Test1 {  
 public static void main(String[] args){  
 WLFUManager<RegionalCitiesBoston> bostonEmployees = new WLFUManager<RegionalCitiesBoston>();//creating a new Boston Array List  
  
 BostonDoctor doc1 = new BostonDoctor("IM Bones","455657890","AA",100);  
 //creating a new object of BostonDoctor  
 doc1.calcSalary(doc1.getBasePay(),doc1.getCOLA(),doc1.getBonus(),doc1.getTotalSalary()); //using the calcSalary function  
 bostonEmployees.Add(doc1); //adding employee to ArrayList  
  
 BostonNurse nurse1 = new BostonNurse("UR Temp","789302345","CA",3);  
 //creating a new object of BostonNurse  
 nurse1.calcSalary(nurse1.getBasePay(),nurse1.getCOLA(),nurse1.getBonus(),nurse1.getTotalSalary()); //using the calcSalary function  
 bostonEmployees.Add(nurse1); //adding employee to ArrayList  
  
 BostonDoctor doc2 = new BostonDoctor("DVM Frakes","786456712","CA",120);  
 //creating a new object of BostonDoctor  
 doc2.calcSalary(doc2.getBasePay(),doc2.getCOLA(),doc2.getBonus(),doc2.getTotalSalary()); //using the calcSalary function  
 bostonEmployees.Add(doc2); //adding employee to ArrayList  
  
 BostonAdministrative admin1 = new BostonAdministrative("IM Boss","543126787","HS", 1);  
 //creating a new object of BostonAdministrative  
 admin1.calcSalary(admin1.getBasePay(),admin1.getCOLA(),admin1.getBonus(),admin1.getTotalSalary()); //using the calcSalary function  
 bostonEmployees.Add(admin1); //adding employee to ArrayList  
  
 for(int i =0; i < bostonEmployees.size(); i++){ //for loop is used to iterate through the ArrayList and print out each object  
 System.*out*.println(bostonEmployees.Get(i));  
 System.*out*.println("");  
 }//end of for loop  
 }//end of main method  
} //end of Test1

5. continued (code output)

IM Bones

Resident Boston Hospital

Race: AA

SSN: 455657890

Special Integer: 100

Job Title: Doctor

Base Salary: $155000.0

COLA: $23250.0

Special Bonus: $38750.0

UR Temp

Resident Boston Hospital

Race: CA

SSN: 789302345

Special Integer: 3

Job Title: Hospital Administrative Nurse

Base Salary: $65000.0

COLA: $9750.0

Special Bonus: $13000.0

DVM Frakes

Resident Boston Hospital

Race: CA

SSN: 786456712

Special Integer: 120

Job Title: Doctor

Base Salary: $155000.0

COLA: $23250.0

Special Bonus: $46500.0

IM Boss

Resident Boston Hospital

Race: HS

SSN: 543126787

Special Integer: 1

Job Title: Senior Executive

Base Salary: $400000.0

COLA: $60000.0

Special Bonus: $80000.0

Process finished with exit code 0