```
//This is the code for the Company() class

//Using subclasses that inherit methods and data from their parent class, Employee(), this program manages a list of up to five salaried or hourly workers.

//It is a looping menu that continually asks the user for input until the condition that the input equals 'f'

//(the quit option) is met. The list allows employees to be added, paychecks to be doled out, hours to changed for hourly workers, raises to be given to specific workers, and

//a sort of toString to be printed out showing the employees currently hired and information about them.


import java.util.Scanner;

import java.text.NumberFormat;


public class Company

{

   //a number of static variables are instantiated so that they can be referred to from any context within the program later

   static NumberFormat money = NumberFormat.getCurrencyInstance();

   static Scanner in = new Scanner(System.in);

   //two arrays, one holding Employees and another holding AdministrativeAssistants(hourly workers). Hourly workers will be added to both arrays but salaried workers will only be added to empArray.

   //AdministrativeAssistants have hoursPerWeek data and hour methods that cant be accessed from the parent class so this array is useful when hour data/methods must be accessed

   static Employee[] empArray = new Employee[5];

   static AdministrativeAssistant[] hourlyEmpArray = new AdministrativeAssistant[5];

   //a temporary Employee object called luckyYou is used to temporarily store the data of certain employees in an array (empArray[i]) that are targeted by the user's input. It is a sort of temporary employee that can be used when a placeholder is useful.

   static Employee luckyYou;

   static String theInput;

   //skipper is a static int set equal to one so that

   static int skipper = 1;
```

```
  // a static method that returns a string.The string must be a single letter. The input given to this method will correspond to option chosen by the user.
  public static String menuLoad()
  {
    String menu = "What would you like to do?\na.  Add an Employee\nb.  List all Employees\nc.  Give an Employee a raise\nd.  Give Paychecks\ne.  Change someone's hours\nf. quit";
    System.out.println(menu);
    String input = in.next();
    String returns;
    String invalid = "That is not a valid option, try again";

if(input.equals("a")||input.equals("A")||input.equals("b")||input.equals("B")||input.equals("c")||input.equals("C")||input.equals("d")||input.equals("D")||input.equals("e")||input.equals("E")||input.equals("f")||input.equals("F"))
    {
      if (input.length()==1)
      {
        returns = input;
      }

      else
      {
        returns = invalid;
      }
    }

    else
    {
      returns = invalid;
```

```
    }
    return returns;
  }


        public static void main(String[] args)
        {
           //a while(true) loop is used to pseudo-infinitely loop the menu and the nested functions inside. Unless the user enters "f" and "F" it will continually ask for input and churn out output.
    while(true)
    {
      //sets the string theInput equal to what is put into the class method menuLoad discussed earlier
      theInput = Company.menuLoad();


      //if conditions check the input. An or operator (||) makes sure this input is not case sensitive)
      if(theInput.equals("a")||theInput.equals("A"))
      {
         //An integer openSlot is created and set equal to 0. Then a for loop iterates through the empArray. For each empty slot, the counter called openSlot goes up by one.
         int openSlot = 0;
         for(int i=0; i<(empArray.length); i++)
         {
            if(empArray[i] == null)
            {
               openSlot++;
            }
            else
            {;}
         }
         //these three are instantiated to store user input
```

```
String newEmpName;
double newEmpSalary;
String empHourlyQ;

//if open slot is equal to or between 1 and 5 that means there is a slot where an employee can be added. The main function of this option begins.
if(openSlot >= 1 && openSlot <= 5)
{
   //Asks for new employee name and assigns it to newEmpName
   System.out.println("What is the employee's name?");
   in.nextLine();
   newEmpName = in.nextLine();

   //Asks for salary amount and assigns it to newEmpSalary
   System.out.println("What is their salary (yearly or hourly)?");
   //a while loop with a try-catch continually asks for a salary input and asks again if the input is bad, ensuring it is a double
   while(true)
   {
      try
      {
         newEmpSalary = in.nextDouble();
         break;
      }

      catch(Exception e)
      {
         System.out.println("Must use only numbers to define salary, try again");
         in.next();
```

```
            }
        }

        System.out.println("Are they an hourly worker? (Y/N)");
        empHourlyQ = in.next();
        //The user's input is sotred and compared in a while loop. If the answer is not a y,Y (yes) or n,N (no) the loop continues to ask for input until it is
        while(true)
        {

if(empHourlyQ.equals("y")||empHourlyQ.equals("Y")||empHourlyQ.equals("n")||empHourlyQ.equals("N"))
            {
                break;
            }

            else
            {
                System.out.println("Invalid entry, use a single letter: Y/N");
                empHourlyQ = in.next();
            }

        }

        //the user's answer is stored. Depending on whether they answer no or yes, a SoftwareEngineer (salaried) is added to the empArray is added to an AdministrativeAssistant (hourly) is added to both arrays.
        int newEmpHours = 0;
        if(empHourlyQ.equals("Y")||empHourlyQ.equals("y"))
        {
```

```
            System.out.println("How many hours does " + newEmpName + " work?");
            while(true)
            {
               try
               {
                  newEmpHours = in.nextInt();
                  break;
               }

               catch(Exception e)
               {
                  System.out.println("Must use only integers to define hours, try again");
                  in.next();
               }
            }

            AdministrativeAssistant newAA = new AdministrativeAssistant(newEmpName,
newEmpSalary, newEmpHours);

            for(int i=0;i<empArray.length;i++)
            {
               if (empArray[i] == null)
               {
                  empArray[i] = newAA;
                  System.out.println(newEmpName + " has been hired as an administrative assistant!");
                  break;
               }
            }
```

```
      for(int i=0;i<hourlyEmpArray.length;i++)
      {
         if (hourlyEmpArray[i] == null)
         {
            hourlyEmpArray[i] = newAA;
            break;
         }
      }
   }

   else
   {
     SoftwareEngineer newAA = new SoftwareEngineer(newEmpName, newEmpSalary);

      for(int i=0;i<empArray.length;i++)
      {
         if (empArray[i] == null)
         {
            empArray[i] = newAA;
            System.out.println(newEmpName + " has been hired as a software engineer!");
            break;
         }
      }
   }
}
else
{
   System.out.println("There's no room for a new employee.");
}
```

```
}

//if theInput=b the employees in empArray are printed along with their information.
if(theInput.equals("b")||theInput.equals("B"))
{
   //a for loop equal to the length of the array is made to sort through the empArray
  for(int i=0; i<empArray.length;i++)
  {
     int trackAA = i;
     //if the employee at empArray[i] is an object and is not an empty slot (null) this code begins
     if(empArray[i] != null)
     {
        //if the class of empArray[i]'s to string is equal to the string that would equal the two string
o an object of class AdministrativeAssistant then it is known this employee has hour data.
        if(empArray[i].getClass().toString().equals("class AdministrativeAssistant"))
        {
          //another for loop iterates through the hourlyEmpArray and each object in it to the
employee object at empArray[i]
           for(int n=0;n<hourlyEmpArray.length;n++)
           {
              //if they are equal then the object at hourlyEmpArray[n] has its information printed
out, including hours.
              if(empArray[trackAA].equals(hourlyEmpArray[n]))
              {
                 System.out.println("Name: " + hourlyEmpArray[n].name + " " + "Salary: " +
money.format(hourlyEmpArray[n].salary) + " " + "Cash: " + money.format(hourlyEmpArray[n].getCash())
+ " " + "Hours worked per week: " + hourlyEmpArray[n].hoursPerWeek + ", " +  "Administrative
Assistant.");
              }
           }
        }
```

```
                else
                {
                    System.out.println("Name: " + empArray[i].name + " " + "Salary: " +
money.format(empArray[i].salary) + " " + "Cash: " + money.format(empArray[i].getCash()) + " "  +
"Software Engineer.");
                }
            }
        }
    }

    //this code is responsible for giving a raise to an employee
    if(theInput.equals("c")||theInput.equals("C"))
    {

        System.out.println("Enter the name of the employee who will be receiving a raise:");
        in.nextLine();
        String raiseFor = in.nextLine();
        String thisYou;

        //for loop to sort through array
        for(int i=0;i<empArray.length;i++)
        {
            //in case the object at empArray[i] is not null there is something there
            if (empArray[i] != null)
            {
                //string thisyou set equal to that object's name data
                thisYou = empArray[i].name;
```

```
            //if the name the user is searching for equals a name in the array the following code is
executed
            if(raiseFor.equals(thisYou))
            {
               //finds out if employee is hourly. If so the print statement is different. Their raise is set.
               if(empArray[i].getClass().toString().equals("class AdministrativeAssistant"))
               {
                  luckyYou = empArray[i];
                  System.out.println("How much of a raise will " + empArray[i].name + " get? As an
administrative assistant who works by the hour, the number given for the raise will be a flat increase on
the employee's hourly rate.");
                  double raiseForEmployee = in.nextDouble();
                  empArray[i].giveRaise(raiseForEmployee);
                  break;
               }

               //finds out if employee is salaried. If so the print statement is different. Their raise is set.
               else if (empArray[i].getClass().toString().equals("class SoftwareEngineer"))
               {
                  luckyYou = empArray[i];
                  System.out.println("How much of a raise will " + empArray[i].name + " get? As a
SoftwareEngineer with a salary, the number given for the raise will be a percentage by which the
employee's salary is increased.");
                  double raiseForEmployee = in.nextDouble();
                  empArray[i].giveRaise(raiseForEmployee);
                  break;
               }

               //in case name isn't found
               else if(raiseFor.equals(thisYou) == false && i == empArray.length-1 || empArray[i] == null
&& i == empArray.length-1)
```

```
            {
               System.out.println("There is no employee here by that name.");
               break;
            }

            else
            {;}
         }
      }

      //another case in case the employee isnt found under different circumstances
      else if(empArray[i] == null && i == empArray.length-1)
      {
         System.out.println("That is not the name of a current employee here.");
         break;
      }
   }
}

//this code is executed if the user enters d, meaning they want to pay the employees
if(theInput.equals("d")||theInput.equals("D"))
{
   System.out.println("Every employee currently under employ will now receive a paycheck.");

   //loops sorts through emp array
   for(int i=0; i<empArray.length; i++)
   {
      if(empArray[i] == null)
      {
```

```
        }

        else if(empArray[i] != null)
        {
           if(empArray[i].getClass().toString().equals("class AdministrativeAssistant"))
           {
              Employee hourlyAsParent = empArray[i];
              AdministrativeAssistant hourlyWorker;
              //sorts through hourly array to execute AdministrativeAssistant specific getPaid method
              for(int n=0;n<hourlyEmpArray.length;n++)
              {
                 if(hourlyAsParent.equals(hourlyEmpArray[n]))
                 {
                    hourlyEmpArray[n].getPaid(hourlyEmpArray[n].salary);
                    hourlyAsParent.cash = hourlyEmpArray[n].cash;
                 }
              }
           }
           //in case the employee is not hourly the standard get paid executes
           else
           {
              empArray[i].getPaid(empArray[i].salary);
           }
        }
     }
  }

//If the user inputs e or E hours can be changed with the following code
```

```
        if(theInput.equals("e")||theInput.equals("E"))
        {
           System.out.println("Who's hours would you like to change?");
           in.nextLine();
           String hourChangeEmp = in.nextLine();
           String hourlyEmployeeHere;
           Employee possibleSalariedEmp;
           //skipper is used here. skipper equals 1 by default so this condition is addressed first.
           //This checks if the employee being searched for is salaried. If they are then they can't have hours changed.
           if (skipper == 1)
           {
              for(int i=0;i<empArray.length;i++)
              {
                 if(empArray[i] != null)
                 {
                    possibleSalariedEmp = empArray[i];
                    String isThisTheGuy = possibleSalariedEmp.name;

                    if(isThisTheGuy.equals(hourChangeEmp))
                    {
                       //if this condition is met, an error message prints, skipper is set equal to zero, and the loop breaks.
                       if(possibleSalariedEmp.getClass().toString().equals("class SoftwareEngineer"))
                       {
                          System.out.println("That employee is paid an annual salary; no hours to adjust.");
                          skipper = 0;
                          break;
                       }
```

```
            }
        }
    }

    //skipper increments. If skipper passed through the first condition without finding the employee to be a
    //software engineer then it will increment to 2 and go through the second condition. If it found the employee to be
    //a SoftwareEngineer then skipper will be set to zero, increment to 1, and finish the loop without going through the next condition.
    skipper++;

    //if the first loop's conditions are passed without finding the employee to be a SoftwareEngineer, skipper increments to 2 then this
    if (skipper == 2)
    {
        for(int i=0;i<hourlyEmpArray.length;i++)
        {
            if(hourlyEmpArray[i] != null)
            {
                hourlyEmployeeHere = hourlyEmpArray[i].name;

                if(hourlyEmployeeHere.equals(hourChangeEmp)&& skipper==2 )
                {
                    System.out.println("What would you like to change this employee's hours to?");
                    while(true)
                    {
                        try
                        {
                            int changedHours = in.nextInt();
```

```
                    hourlyEmpArray[i].setHours(changedHours);
                    skipper = 0;
                    break;
                }

                catch(Exception e)
                {
                    System.out.println("Must use only integers to define hours, try again");
                    in.next();
                }
            }
        }

        else if(hourlyEmployeeHere.equals(hourChangeEmp) == false && i ==
hourlyEmpArray.length-1 || hourlyEmpArray[i] == null && i == hourlyEmpArray.length-1)
        {
            System.out.println("That employee cannot be found.");
            break;
        }

        else
        {;}
    }

    else if(hourlyEmpArray[i] == null && i == hourlyEmpArray.length-1 && skipper == 2)
    {
        System.out.println("There is no employee currently working here under that name.");
        break;
    }
```

```
            }
          }
        }
        else
        {;}
        skipper = 1;


      }


      //this line ends the program if the user enters f or F
      if(theInput.equals("f")||theInput.equals("F"))
      {
        System.out.println("See you later! Ending program...");
        //terminates the program
        System.exit(0);
      }


      //in case the user enters anything other than what's on the menu
      else if(theInput.equals("a") == false && theInput.equals("A") == false && theInput.equals("b") == false && theInput.equals("B") == false && theInput.equals("c") == false && theInput.equals("C") == false && theInput.equals("d") == false && theInput.equals("D") == false && theInput.equals("e") == false && theInput.equals("E") == false && theInput.equals("f") == false && theInput.equals("F") == false)
      {
        System.out.println("That is in an invalid input, please try again");
      }
       }
  }
}
```

```
//this is the code for the Employee() class
abstract class Employee
{
   protected String name;
   protected double salary;
   protected double cash;

   public double getCash()
   {
      return cash;
   }

   public void getPaid(double salary)
   {
      cash += salary/26;
   }

   public abstract void giveRaise(double percentage);

}
```

```
//This is the code for the SoftwareEngineer() class
class SoftwareEngineer extends Employee
{
   SoftwareEngineer(String engrName, double engrSalary)
   {
      this.name = engrName;
      this.salary = engrSalary;
      this.cash = 0;
   }

   public void giveRaise(double percentage)
   {
      this.salary = (salary * (percentage * .01)) + salary;
   }

   public double getCash()
   {
      return cash;
   }
}
```

```
//This is the code for the AdministrativeAssistant class
class AdministrativeAssistant extends Employee implements Hourly
{
   public int hoursPerWeek;

   AdministrativeAssistant(String adminName, double adminSalary, int adminHour)
   {
      this.name = adminName;
      this.salary = adminSalary;
      this.hoursPerWeek = adminHour;
   }

   public int getNumOfHours()
   {
      return this.hoursPerWeek;
   }

   public void setHours(int newHours)
   {
      this.hoursPerWeek = newHours;
   }

   public void giveRaise(double increase)
   {
      this.salary += increase;
   }

   public void getPaid(double salary)
   {
```

```
        cash += salary * (this.hoursPerWeek * 2);
    }


}
```

```
//this is the code for the Hourly interface
interface Hourly
{
   public int getNumOfHours();

   public void setHours(int newHours);

   public void giveRaise(double raise);
}
```