/\*\*

\* Employee is an abstract class for all type of employee classes.

\* By implementing this class, an application can create following data:

\* <ul>

\* <li>First Name of the employee</li>

\* <li>Last Name of the employee</li>

\* <li>SSN of the employee</li>

\* </ul>

\* @author zhenhua.yang.1

\* @version 1.0

\*/

public abstract class Employee {

// instance variables

private String firstName;

private String lastName;

private String ssn;

//constructor

public Employee( String fn, String ln, String s ){

firstName = fn;

lastName = ln;

ssn = s;

}

//Accesser methods:

public String getFirstName(){

return firstName;

}

public String getLastName(){

return lastName;

}

public String getSSN(){

return ssn;

}

// mutator methods

public void setFirstName( String newFirstName ){

firstName = newFirstName;

}

public void setLastName( String newLastName ){

firstName = newLastName;

}

// abstract gerEarnings() method

public abstract double getEarnings();

public String toString(){

return "Full Name: " + firstName +" " + lastName + "\tSSN: " + ssn;

}

/\*\*

\*

\* @param o

\* @return

\*/

@Override

public boolean equals( Object o ){

if( ! (o instanceof Employee) )

return false;

else{

Employee objE = (Employee)o;

return objE.firstName.equals(firstName)

&& objE.lastName.equals(lastName)

&& objE.ssn.equals(ssn);

}

}

}

import java.text.DecimalFormat;

public class HourlyEmployee extends Employee {

// create instance variables

private double wage;

private int hours;

// create DecimalFormat object currency

DecimalFormat currency = new DecimalFormat("$0.00");

/\*\*

\* constructor that create an HourlyEMployee object

\* @param newFirstName

\* @param newLastName

\* @param newSSN

\* @param newWage

\* @param newHours

\*/

public HourlyEmployee(String newFirstName, String newLastName,

String newSSN, double newWage, int newHours){

super( newFirstName, newLastName, newSSN);

wage = newWage;

hours = newHours;

}

// accessor methods

public double getWage(){

return wage;

}

public int getHours(){

return hours;

}

// mutator methods

public void setWage( double newWage ){

if(newWage>=0){

wage = newWage;

}else{

throw new IllegalArgumentException( "the new Wage should not be negative." );

}

}

public void setHours( int newHours ){

if( newHours >= 0 ){

hours = newHours;

}else{

throw new IllegalArgumentException("the number of hours should not be negative");

}

}

/\*\*

\* getEearning() method that returns the employee's weekly income.

\* @return a double value that represents total earning during the week

\*/

@Override

public double getEarnings(){

if( getHours() > 40 )

return ( getWage() \* 40 ) + ( getWage() \* ( getHours() - 40) \* 1.5 );

else

return getWage() \* getHours();

}

@Override

public String toString(){

return super.toString() + "\tEmployee Type: Hourly" + ", \tWorking Hours: " + getHours()

+ "\twage: " + currency.format(getWage()) + "\tEarnings: " + currency.format(this.getEarnings());

}

@Override

public boolean equals( Object o ){

if( !( o instanceof HourlyEmployee ) )

return false;

else{

HourlyEmployee objH = (HourlyEmployee) o;

return super.equals(objH)

&& wage == objH.wage

&& hours == objH.hours;

}

}

}

import java.text.DecimalFormat;

/\*\*

\*

\* @author zhenhua.yang.1

\*/

public class SalariedEmployee extends Employee {

// instance variable

private double weeklySalary;

DecimalFormat currency = new DecimalFormat("$0.00");

/\*\*

\* constructor that allow the application create an SalariedEmployee object.

\* @param newFirstName

\* @param newLastName

\* @param newSSN

\* @param newWeeklySalary

\*/

public SalariedEmployee(String newFirstName, String newLastName, String newSSN,

double newWeeklySalary ){

super( newFirstName, newLastName, newSSN);

weeklySalary = newWeeklySalary;

}

// accessor method

public double getWeeklySalary(){

return weeklySalary;

}

// mutator method

public void setWeeklySalary( double newWeeklySalary ){

if( newWeeklySalary >= 0 )

weeklySalary = newWeeklySalary;

else

throw new IllegalArgumentException("the weekly salary should not be negative");

}

/\*\*

\*

\* @return

\*/

@Override

public String toString(){

return super.toString() + "\tEmployee Type: Salaried" + ", \tWeekly Salary: " +

currency.format(getWeeklySalary()) + "\tEarnings: " + currency.format(this.getEarnings());

}

@Override

public boolean equals( Object o ){

if( ! ( o instanceof SalariedEmployee ))

return false;

else{

SalariedEmployee objS = (SalariedEmployee)o;

return super.equals(objS)

&& this.getWeeklySalary() == objS.getWeeklySalary();

}

}

@Override

public double getEarnings() {

return getWeeklySalary();

}

}

import java.text.DecimalFormat;

/\*\*

\*

\* @author zhenhua.yang.1

\*/

public class CommissionEmployee extends Employee {

// create instance variables

private double grossSales;

private double commissionRate;

// create DecimalFormat object currency

DecimalFormat currency = new DecimalFormat("$0.00");

/\*\*

\* constructor

\* @param newFirstName

\* @param newLastName

\* @param newSSN

\* @param newGrossSales

\* @param newCommissionRate

\*/

public CommissionEmployee(String newFirstName, String newLastName,

String newSSN, double newGrossSales, double newCommissionRate){

super( newFirstName, newLastName, newSSN);

grossSales = newGrossSales;

commissionRate = newCommissionRate;

}

// accessor methods

public double getGrossSales(){

return grossSales;

}

public double getCommissionRate(){

return commissionRate;

}

// mutator methods

public void setGrossSales( double newGrossSales ){

grossSales = newGrossSales;

}

public void setCommissionRate( double newCommissionRate ){

commissionRate = newCommissionRate;

}

public String toString(){

return super.toString() + "\tEmployee Type: Commission" + ", \tGross Sales: " + currency.format(getGrossSales())

+ "\tCommission Rate: " + currency.format(getCommissionRate()) + "\tEarnings: " + currency.format(this.getEarnings());

}

@Override

public boolean equals( Object o ){

if( ! ( o instanceof CommissionEmployee ))

return false;

else{

CommissionEmployee objC = (CommissionEmployee)o;

return super.equals(objC)

&& this.getGrossSales() == objC.getGrossSales()

&& this.getCommissionRate() == objC.getCommissionRate();

}

}

public double getEarnings(){

return commissionRate \* grossSales;

}

}

import java.text.DecimalFormat;

public class BasePlusCommissionEmployee extends CommissionEmployee {

// instance variable

private double baseSalary;

// create DecimalFormat object currency

DecimalFormat currency = new DecimalFormat("$0.00");

/\*\*

\* constructor that create an BasePlusCommissionEmployee object

\* @param newFirstName

\* @param newLastName

\* @param newSSN

\* @param newGrossSales

\* @param newCommissionRate

\* @param newBaseSalary

\*/

public BasePlusCommissionEmployee(String newFirstName, String newLastName, String newSSN,

double newGrossSales, double newCommissionRate, double newBaseSalary){

super( newFirstName, newLastName, newSSN, newGrossSales, newCommissionRate );

baseSalary = newBaseSalary;

}

// accessor methods

public double getBaseSalary(){

return baseSalary;

}

public void setBaseSalary( double newBaseSalary ){

if( newBaseSalary >= 0 )

baseSalary = newBaseSalary;

else

throw new IllegalArgumentException( "the base salary should not be negative" );

}

/\*\*

\* getEearning() method that returns the employee's earning.

\* @return a double value that represents total earning

\*/

public double getEarnings(){

return (getCommissionRate() \* getGrossSales()) + baseSalary;

}

public String toString(){

return super.toString() + "\tEmployee Type: Base+Commission" + ", \tBase Salary: " +

currency.format(getBaseSalary()) + "\tEarnings: " + currency.format(this.getEarnings());

}

public boolean equals( Object o ){

if( !( o instanceof BasePlusCommissionEmployee ))

return false;

else{

BasePlusCommissionEmployee objH = (BasePlusCommissionEmployee) o;

return super.equals(objH)

&& this.getEarnings() == objH.getEarnings()

&& baseSalary == objH.baseSalary;

}

}

}

/\*\*

\*

\* @author zhenhua.yang.1

\* @version 1.0

\*/

import java.util.ArrayList;

import java.util.Random;

public class EmployeeClient {

public static void main( String [] args ){

// create ArrayList object employees and implement generateEmployees()

// method to generate ten employees

ArrayList<Employee> employees = generateEmployees();

Employee worker = new HourlyEmployee( "Aaron", "Yang", "5079", 60, 40);

// use a for loop to print all the content of the emplyees.

for ( int i = 0; i < employees.size(); i++ ){

System.out.println(employees.get(i).toString());

}

}

// static method that generates ten employee objects and store in a ArrayList.

public static ArrayList<Employee> generateEmployees(){

// creare a ArrayList object list

ArrayList<Employee> list = new ArrayList<>();

// instantiate a Randome object rand

Random rand = new Random();

// create String arrays for ten first names, then last names and tem SSNs.

String[] firstNames = { "Amy", "Aaron", "Jake", "Ming", "Ann",

"Mike", "Namcy", "Bryn", "James", "Will" };

String[] lastNames = { "Lee", "Smith", "Fan", "Yang", "Zhang",

"Black", "Vogel", "Megol", "Kong" };

String[] ssn = { "0987", "2312", "5092", "2309", "0934",

"0876", "7823", "9820", "0923", "5667" };

// for loop to create ten Employee objects and add each to list.

for( int i = 0; i < 10; i++ ){

// craete a random integer (value) from 0 to 3

int value = rand.nextInt(3);

// if value = 0, create HourlyEmployee object hourly

switch (value) {

case 0:

{

int hour = rand.nextInt(40);

int name = rand.nextInt(9);

Employee hourly = new HourlyEmployee( firstNames[name], lastNames[name], ssn[name],

(Math.random() + 0.5) \* 10, hour);

list.add(hourly);

break;

}

case 1:

{

// if valur = 1, create a SalariedEmployee object salaried

int name = rand.nextInt(9);

Employee salaried = new SalariedEmployee( firstNames[name], lastNames[name], ssn[name],

(Math.random() + 5) \* 100);

list.add(salaried);

break;

}

case 2:

{

// if valur = 2, create a CommissionEmployee object commission

int name = rand.nextInt(9);

Employee commission = new CommissionEmployee( firstNames[name], lastNames[name], ssn[name],

(Math.random() + 0.5) \* 10, (Math.random() + 0.5) \* 10 );

list.add(commission);

break;

}

case 3:

{

// if valur = 3, create a BasePlusCommissionEmployee object basePlusComs

int name = rand.nextInt(9);

Employee basePlusComs = new BasePlusCommissionEmployee( firstNames[name], lastNames[name], ssn[name],

(Math.random() + 0.5) \* 10, (Math.random() + 0.5) \* 10,

(Math.random() + 0.5) \* 10);

list.add(basePlusComs);

break;

}

default:

break;

}

}

return list;

}

}

Run 1:

Full Name: Bryn Megol SSN: 9820 Employee Type: Hourly, Working Hours: 32 wage: $7.92 Earnings: $253.58

Full Name: Namcy Vogel SSN: 7823 Employee Type: Hourly, Working Hours: 18 wage: $7.08 Earnings: $127.52

Full Name: James Kong SSN: 0923 Employee Type: Salaried, Weekly Salary: $552.46 Earnings: $552.46

Full Name: Bryn Megol SSN: 9820 Employee Type: Salaried, Weekly Salary: $539.70 Earnings: $539.70

Full Name: Bryn Megol SSN: 9820 Employee Type: Salaried, Weekly Salary: $576.12 Earnings: $576.12

Full Name: Ming Yang SSN: 2309 Employee Type: Commission, Gross Sales: $11.15 Commission Rate: $5.36 Earnings: $59.81

Full Name: Aaron Smith SSN: 2312 Employee Type: Hourly, Working Hours: 13 wage: $8.68 Earnings: $112.82

Full Name: Bryn Megol SSN: 9820 Employee Type: Commission, Gross Sales: $8.15 Commission Rate: $10.86 Earnings: $88.49

Full Name: Aaron Smith SSN: 2312 Employee Type: Commission, Gross Sales: $11.23 Commission Rate: $13.33 Earnings: $149.73

Full Name: Mike Black SSN: 0876 Employee Type: Salaried, Weekly Salary: $543.26 Earnings: $543.26

BUILD SUCCESSFUL (total time: 0 seconds)

run:

Full Name: Jake Fan SSN: 5092 Employee Type: Hourly, Working Hours: 20 wage: $11.79 Earnings: $235.75

Full Name: Namcy Vogel SSN: 7823 Employee Type: Commission, Gross Sales: $14.64 Commission Rate: $6.71 Earnings: $98.20

Full Name: Aaron Smith SSN: 2312 Employee Type: Commission, Gross Sales: $9.28 Commission Rate: $10.08 Earnings: $93.51

Full Name: Ann Zhang SSN: 0934 Employee Type: Salaried, Weekly Salary: $536.61 Earnings: $536.61

Full Name: Ann Zhang SSN: 0934 Employee Type: Salaried, Weekly Salary: $549.75 Earnings: $549.75

Full Name: Mike Black SSN: 0876 Employee Type: Salaried, Weekly Salary: $579.40 Earnings: $579.40

Full Name: Ming Yang SSN: 2309 Employee Type: Salaried, Weekly Salary: $584.23 Earnings: $584.23

Full Name: James Kong SSN: 0923 Employee Type: Hourly, Working Hours: 14 wage: $5.72 Earnings: $80.14

Full Name: Aaron Smith SSN: 2312 Employee Type: Salaried, Weekly Salary: $567.82 Earnings: $567.82

Full Name: Ann Zhang SSN: 0934 Employee Type: Commission, Gross Sales: $8.26 Commission Rate: $13.18 Earnings: $108.94

BUILD SUCCESSFUL (total time: 0 seconds)



