package employee;

/\*\*

\* @author Andrew

\* @since 3-28-2019 I pledge that this program represents my own program code. I

\* received code from and shared my code with no one.

\*/

public class HourlyEmployee extends FullTimeEmployee implements Employee {

public final static int MAX\_REGULAR\_HOURS = 40;

public final static double OVERTIME\_FACTOR = 1.5;

protected int hoursWorked;

protected double payRate, regularPay, overtimePay;

/\*\*

\* Default constructor that initializes everything

\*/

public HourlyEmployee() {

hoursWorked = 0;

payRate = 0.00;

regularPay = 0.00;

overtimePay = 0.00;

} // default constructor

/\*\*

\* @param name - the specified name.

\* @param hoursWorked - the specified hours worked.

\* @param payRate - the specified pay rate.

\*

\*/

public HourlyEmployee(String name, int hoursWorked, double payRate) {

this.name = name;

this.hoursWorked = hoursWorked;

this.payRate = payRate;

if (hoursWorked <= MAX\_REGULAR\_HOURS) {

regularPay = hoursWorked \* payRate;

overtimePay = 0.00;

} // if

else {

regularPay = MAX\_REGULAR\_HOURS \* payRate;

overtimePay = (hoursWorked - MAX\_REGULAR\_HOURS)

\* (payRate \* OVERTIME\_FACTOR);

} // else

grossPay = regularPay + overtimePay;

} // 3-parameter constructor

/\*\*

\* @return the hours worked by this full-time HourlyEmployee object.

\*/

public int getHoursWorked() {

return hoursWorked;

} //getHoursWorked()

/\*\*

\* @return the pay rate this full-time HourlyEmployee object.

\*/

public double getPayRate() {

return payRate;

} //getPayRate()

/\*\*

\* @return the regular pay this full-time HourlyEmployee object.

\*/

public double getRegularPay() {

return regularPay;

} //getRegularPay()

/\*\*

\* @return the overtime pay this full-time HourlyEmployee object.

\*/

public double getOvertimePay() {

return overtimePay;

} //getOvertimePay()

/\*\*

\* @return a String representation of this full-time HourlyEmployee object.

\*/

public String toString() {

return super.toString() + " HOURLY";

} //toString()

}

package employee;

import java.util.\*;

import java.io.\*;

/\*\*

\* @author Andrew

\* @since 3-28-2019 I pledge that this program represents my own program code. I

\* received code from and shared my code with no one.

\*/

public class Company {

public static void main(String[] args) throws FileNotFoundException {

new Company().run();

} //main()

/\*\*

\* Determines and prints out the best paid of the full-time employees

\* scanned in from a specified file.

\*

\*/

public void run() throws FileNotFoundException {

final String INPUT\_PROMPT = "Please enter the path for the file of employees: ";

final String BEST\_PAID\_MESSAGE

= "\n\nThe best-paid employee (and gross pay) is ";

final String NO\_INPUT\_MESSAGE

= "\n\nError: There were no employees scanned in.";

String fileName;

System.out.print(INPUT\_PROMPT);

fileName = new Scanner(System.in).nextLine();

Scanner sc = new Scanner(new File(fileName));

FullTimeEmployee bestPaid = findBestPaid(sc);

if (bestPaid == null) {

System.out.println(NO\_INPUT\_MESSAGE);

} else {

System.out.println(BEST\_PAID\_MESSAGE + bestPaid.toString());

}

} //run()

/\*\*

\* @param sc – the Scanner object used to scan in the employees.

\*

\* @return the best paid of all the full-time employees scanned in, or null

\* there were no employees scanned in.

\*/

public FullTimeEmployee findBestPaid(Scanner sc) {

FullTimeEmployee full,

bestPaid = new FullTimeEmployee();

while (sc.hasNext()) {

full = getNextEmployee(sc);

if (full.getGrossPay() > bestPaid.getGrossPay()) {

bestPaid = full;

}

} //while

if (bestPaid.getGrossPay() == 0.00) {

return null;

}

return bestPaid;

} //findBestPaid()

/\*\*

\* @param sc – the Scanner object over the file.

\*

\* @return the next full-time employee scanned in from sc.

\*/

protected FullTimeEmployee getNextEmployee(Scanner sc) {

Scanner lineScanner = new Scanner(sc.nextLine());

String name = lineScanner.next();

double grossPay = lineScanner.nextDouble();

return new FullTimeEmployee(name, grossPay);

} //getNextEmployee()

}