|  |  |
| --- | --- |
|  | |
| Add Employee SubClass PartsWorker | |
| **Updated** | **10.28.2018 2:40 PM** |

# **Problem 1:**

Modify the Payroll system to include an additional Employee subclass PartsWorker that represents an employee whose pay is based on the number of pieces of merchandise produced. For example if the employee earns €5 for each part produced and he/she produces 20 parts then total payment is €100. This category of employee also has a bonus incentive scheme. If the employee produces more than a certain number of pieces then a standard bonus is added to their total payment. Class PartsWorker should contain private variables to store payment per piece, the number of pieces produced, the fixed bonus amount and the number of pieces required before a bonus is applied. Provide a concrete implementation of earnings in class PartsWorker that calculates the employee's earnings appropriately. Add two other methods to the PartsWorker class that you think might be useful. Provide a separate class to test the functionality of the PartsWorker class

# **Approach**

The main approach was to modify the payroll system to include an additional Employee subclass called PartsWorker and correctly calculate the wage and bonus as set out in the question. In addition I add two additional methods to calculate average pieces per day and average wage per day.

# **BankAccount Design**

PartsWorker extends Employee

Declare instance variable wage for payment per piece

Declare instance variable pieces number of pieces produced

Declare instance variable bonusAmount for fixed bonus amount

Declare instance variable bonusPieces for pieces required before a bonus is applied

Create a seven-argument constructor calling super variables

Set the wage method

Get the wage method

Set the pieces method

Get the pieces method

Set the bonusAmount

Get the bonusAmount

Set the bonusPieces

Get the bonusPieces

Calculate weekly earnings; override abstract method earnings in Employee

ADDITIONAL METHOD 1: calculate average pieces produced per day by Parts Worker

ADDITIONAL METHOD 2: calculate average amount earner per day by Parts Worker

Return String representation of PartsWorker object

EXIT

# **Testing**

The program was run several times with different inputs to ensure the code behaves as expected for each possible execution scenario including the input of invalid and out of bound parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **Actual Result** |
| 1 | Create **PartsWorker**  objects, with seven variables | **BankAccount**  Object is created seven variables | **BankAccount**  Object is created seven variables |
| 2 | Create **PartsWorker** object with less that bonus amount of pieces | Correct weekly wage is calculated without bonus amount | Correct weekly wage is calculated without bonus amount |
| 3 | Create **PartsWorker** object with more that bonus amount of pieces | Correct weekly wage is calculated with bonus amount | Correct weekly wage is calculated with bonus amount |
| 4 | Create **PartsWorker** object with less than zero number of pieces | Number of pieces is set to zero | Number of pieces is set to zero |
| 5 | Create **PartsWorker** object with less than zero wage amount | Wage amount is set to $0.0 | Wage amount is set to $0.0 |
| 6 | Create **PartsWorker** object with less than zero bonus pieces amount | Bonus pieces amount is set to zero | Bonus pieces amount is set to zero |
| 7 | Create **PartsWorker** object with less than zero bonus amount | Bonus amount is set to $0.0 | Bonus amount is set to $0.0 |
| 8 | Calculate average pieces produced per day by **PartsWorker** | Returns correct average pieces produced per day by **PartsWorker** | Returns correct average pieces produced per day by **PartsWorker** |
| 9 | Calculate average wage produced per day by **PartsWorker** | Returns correct average wage per day by **PartsWorker** | Returns correct average wage per day by **PartsWorker** |
| 10 | Return string representation of **PartsWorker** object | Correct string representation of **PartsWorker** object returned | Correct string representation of **PartsWorker** object returned |

# **Source Code – PartsWorker.java**

|  |
| --- |
| /\* mmcmahon\_wk5\_Assignment\_PartsWorker.java  \* Employee subclass PartsWorker that represents an employee whose pay is based on the number of pieces of merchandise produced.  \* Class PartsWorker contains private variables to store: Payment per piece, Number of pieces produced, Fixed bonus amount  \* Number of pieces required before a bonus is applied. Add two other methods to the PartsWorker class that you think might be useful.  Notes:  NegativePieceWageException? – see slide 15 deck 3  \*/  **public** **class** PartsWorker **extends** Employee {  **private** **double** wage; // declare instance variable wage for payment per piece  **private** **int** pieces; // declare instance variable pieces number of pieces produced  **private** **double** bonusAmount = 100.00; // declare instance variable bonus amount for fixed bonus amount  **private** **int** bonusPieces = 100; // declare instance variable bonuspieces for pieces required before a bonus is applied  // seven-argument constructor  **public** PartsWorker(String first, String last, String ssn, **double** PieceWage,  **int** NumerOfPieces, **double** FixBonusAmount, **int** FixBonusPieces) {  **super**(first, last, ssn);  setWage(PieceWage); // validate and store wage  setPieces(NumerOfPieces); // validate and store pieces per week  setBonusAmount(FixBonusAmount); // validate and store bonus amount  setBonusPieces(FixBonusPieces); // validate and store the amount of pieces per week for bonus  } // end seven-argument PartsWorker constructor  // set wage method  **public** **void** setWage(**double** PieceWage) {  wage = (PieceWage < 0.0) ? 0.0 : PieceWage;  } // end set wage method    // return wage method  **public** **double** getWage()  {  **return** wage;  } // end return wage method  // set pieces method  **public** **void** setPieces(**int** NumerOfPieces) {  pieces = (NumerOfPieces >= 0) ? NumerOfPieces : 0;  } // end set pieces method  // return pieces method  **public** **int** getPieces() {  **return** pieces;  } // end return pieces method    // set bonus amount method  **public** **void** setBonusAmount(**double** FixBonusAmount) {  bonusAmount = (FixBonusAmount < 0.0) ? 0.0 : FixBonusAmount;  } // end set bonus amount method    // return bonus amount method  **public** **double** getBonusAmount() {  **return** bonusAmount;  } // end return bonus amount method    // set bonus pieces method  **public** **void** setBonusPieces(**int** FixBonusPieces) {  bonusPieces = (FixBonusPieces >= 0) ? FixBonusPieces : 0;  } // end set bonus pieces method  // return bonus pieces method  **public** **int** getBonusPieces() {  **return** bonusPieces;  } // end return bonus pieces method      // calculate weekly earnings; override abstract method earnings in Employee  **public** **double** earnings()  {  **if** (getPieces() < 100) // no bonus  **return** getWage() \* getPieces();  **else**  **return** (getPieces() \* getWage()) + bonusAmount; // with bonus  } // end method weekly earnings    //ADDITIONAL METHOD 1: calculate average pieces produced per day by Parts Worker  **public** **int** avgPiecesPerDay()  {  **return** getPieces()/5; //assuming an 5 day week  } // end method average pieces produced per day    //ADDITIONAL METHOD 2: calculate average amount earner per day by Parts Worker  **public** **double** avgWagePerDay()  {  **return** (getWage() \* getPieces())/5; //assuming an 5 day week  } // end method average amount earner per day  // return String representation of PartsWorker object  **public** String toString()  {  **return** String.*format*(  "Parts Worker: %s\n"  + "%s: $%,.2f;\n"  + "%s: %d \n"  + "%s: $%,.2f;\n"  + "%s: %d \n"  + "%s: %d \n"  + "%s: $%,.2f;",    **super**.toString(),  "Payment per piece", getWage(),  "Number of pieces produced per week", getPieces(),  "The Bonus Amount", getBonusAmount(),  "The Number of pieces per week for bonus", getBonusPieces(),  "The Average number of pieces produced per day", avgPiecesPerDay(),  "The Average amount earned per day by Parts Worker", avgWagePerDay());  } // end method toString    } // end class PartsWorker |

# **Source Code – TestPartsWorker.java**

|  |
| --- |
| /\* mmcmahon\_wk5\_Assignment\_PartsWorkerTest.java  \* Provide a separate class to test the functionality of the PartsWorker class.  \*/  **public** **class** PartsWorkerTest {  **public** **static** **void** main(String[] args) {    // create PartsWorker object without bonus  PartsWorker partsWorker1 =  **new** PartsWorker( "John", "Doe", "111-11-1111", 5, 99, 100, 100 );    System.***out***.printf( "%s\nWeekly Wage Earned: $%,.2f\n\n",  partsWorker1, partsWorker1.earnings() );    // create PartsWorker object with bonus  PartsWorker partsWorker2 =  **new** PartsWorker( "Jane", "Doe", "222-22-2222", 5, 100, 100, 100 );  System.***out***.printf( "%s\nWeekly Wage Earned: $%,.2f\n\n",  partsWorker2, partsWorker2.earnings() );    // create PartsWorker object with less than zero amounts  PartsWorker partsWorker3 =  **new** PartsWorker( "John", "Doe", "222-22-2222", -5, -100, -100, -100 );  System.***out***.printf( "%s\nWeekly Wage Earned: $%,.2f\n\n",  partsWorker3, partsWorker3.earnings() );  }  } |

# **References:**

Deitel, H. & Deitel, P. (2010). *Java* (9th ed.). Upper Saddle River, NJ: Prentice Hall.