Converting Pseudocode to Java

When software engineers move from design to implementation, they convert designs (presented in pseudocode or some other design tool) to programming code in a specific language. This handout presents many examples of pseudocode converted to the programming language Java. As you read through the conversion examples, you will notice how similar the pseudocode and Java code can be. Keep in mind that each language has its own syntax or set of rules. The following notes introduce you to some of the Java syntax requirements.

Some Java Syntax Notes to Get Started:

a. Comments start with //

b. Each variable used **must** be declared first! Each variable declaration should include a comment to explain how the variable is used.

int - for variables that are declared in pseudocode as Integer (have no decimal point)

```
int count; // counts the number of items
int num; // number entered by user
int sum = 0; // sum of the values entered (note: sum is initialized to 0)
```

float - for variables that are declared in pseudocode as Real (have a decimal point)

```
float battingAverage; // player's batting average
float average = 0.0; // average of all the values in the array
```

double - for variables that are declared in pseudocode as Real (have a decimal point; higher precision than float)

```
double myAverage; // my average score
double sum;    // sum of hourly pay rates
```

char - for variables that are declared in pseudocode as Character (contain a single character)

```
char firstInitial; // employee's first initial
char middleInitial; // employee's middle initial
char lastInitial; // employee's last initial
```

c. Variable names must start with a lowercase letter.

d. **{ }** are braces and they surround the entire class code and any internal structure.

e Every programming statement ends with a semi-colon **;** (see the examples in this handout). You will notice that the beginning of the while loop and if-then structures do not end with a semi-colon (;). This is a special case in the Java language. Pay close attention to the examples.

f. ALL reserved words (while, if, main, int, float, char…) are lowercase in Java.

g. **`System.out.println()`** is the command for output in Java. In pseudocode we used Display.

h. Any program that inputs something uses the Scanner class. The Scanner class must be imported using the following statement, which appears before the class definition:

```
import java.util.Scanner;
```

Then an object of the scanner class must be declared:

```
Scanner scan = new Scanner(System.in);
```

Then values can be input using the Scanner object:

```
int inputValue;
inputValue = scan.nextInt();
```

i. Every opening brace **{** MUST have a corresponding closing brace **}**

This table presents small "code segments" presented first in pseudocode and then in Java	
Pseudocode	**Java**

Declare Integer num
Set num = 5
Display num
	`int num; // number to print`
	`num = 5;`
	`System.out.println (num);`

Declare Real num
Set num = 5.3
Display "Number is: ", num
	`double num;  // number to print`
	`num = 5.3;`
	`System.out.println ("Number is: " + num);`

Declare Integer num
Set num = 5
Constant Integer MAX = 10
If num < MAX Then
  Display num * MAX
End If
	`int num = 5; // number to multiply`
	`final int MAX = 10; // value to multiply`
	`if (num < MAX)`
	`{`
	`  System.out.println (num * MAX);`
	`}`

Declare Integer num = 20
Constant Integer MAX = 10
If num < MAX Then
  Display num * MAX
Else
  Display num / MAX
End If
	`int num = 20; // number to manipulate`
	`final int MAX = 10; // value to multiply or divide`
	`if (num < MAX)`
	`{`
	`  System.out.println (num * MAX);`
	`}`
	`else`
	`{`
	`  System.out.println (num / MAX);`
	`}`

Declare Integer count = 0
Constant Integer MAX = 10
While count < MAX
  Display "count: ", count
  Set count = count + 1
End While
	`int count = 0; // counter`
	`final int MAX = 10; // number of times to count`
	`while (count < MAX)`
	`{`
	`  System.out.println ("count: " + count);`
	`  count = count + 1;`
	`}`

Declare Real average
Declare Real sum = 0
Declare Real addAmt
Declare Integer count = 0
Display “Enter numbers to add.”
Display “Enter 0 to quit.”
Input addAmt
While addAmt != 0
  Set sum = sum + addAmt
  Set count = count + 1
  Input addAmt
End While
Set average = sum / count
Display “Sum is: ”, sum
Display “Average is:”, average
	`import java.util.Scanner;`
	`double average; // average of numbers entered`
	`double sum = 0; // sum of numbers entered`
	`double addAmt; // number entered to add to sum`
	`int count = 0; // counts number of values entered`
	`Scanner scan = new Scanner (System.in);`
	`System.out.println ("Enter numbers to add.");`
	`System.out.println ("Enter 0 to quit.");`
	`addAmt = scan.nextDouble();`
	`while (addAmt != 0)`
	`{`
	`  sum = sum + addAmt;`
	`  count = count + 1;`
	`  addAmt = scan.nextDouble();`
	`}`
	`average = sum / count;`
	`System.out.println ("Sum is: " + sum);`
	`System.out.println ("Average is: " + average);`

Declare Integer x = 5
If x < 10 Then
  If x > 0 Then
  Display x, “ from 0 to 10"
  End If
End If
	`int x = 5; // value to check`
	`if (x < 10)`
	`{`
	`  if (x > 0)`
	`  {`
	`  System.out.println (x + " from 0 to 10");`
	`  } // closes inner if`
	`} // closes outer if`

Example_1: A sample program with explanations for each line.

Here is a sample Java program followed by explanations for each numbered statement or set of statements. The ***numbers*** by the statements are included for reference purposes only and are **NOT** included in actual programs.

```
1.  /**
2.   *  This program outputs the numbers from 0 through 9.
3.   */
4.  class OutputNums
5.  {
6.     public static void main(String[ ] args)
7.     {
8.        int count = 0; // counter for values being output
9.        final int MAX = 10; // determines when to stop
10.       while (count < 10)
11.       {
12.          System.out.println (count);
13.          count = count + 1;
14.       }
15.    }//end of main
16. } // end of class definition
```

Explanation for the numbered statements in Example_1:

1. This begins a javadoc comment, which is a special comment used in Java programs that can be used to generate a web page that provides documentation for other programmers. All classes, attributes, and methods should be commented using a javadoc comment. The javadoc comment starts with /** and ends with */
2. Continuation of javadoc comment.
3. End of javadoc comment.
4. Beginning of class definition for OutputNums class. Note that all methods in a Java program are inside a class; even the main() method. Class names begin with an uppercase letter. This line does NOT have a semi-colon at the end.
5. This is the opening brace for the class definition. This line does NOT have a semi-colon at the end.
6. The is the header for the main() method. Note that the main() method in Java is ALWAYS declared **`public static void`**. The static keyword is used because an object of the class is not declared before calling the main() method, since it is called by the operating system when the program is launched. Always use **`String[] args`** between the parentheses as the parameter for main(). This allows a program to accept arguments from the command line. It is not something that we will use this semester, but you must include it every time you create a main program. This line does NOT have a semi-colon at the end.
7. This is an opening brace to begin the main function. This line does NOT have a semi-colon at the end.
8. count is an integer. All variables MUST be declared, providing the data type and the variable name. This is often done at the top/beginning of the method. This statement assigns an initial value of 0 to the variable count.
9. MAX is a constant with a value of 10. Notice constants in Java are indicated by using the keyword **final**.
10. while loop structure. ALWAYS lower case in the Java programming language. The condition for the while loop is ALWAYS inside of parentheses. This line does NOT have a semi-colon at the end.
11. The opening brace { is needed to surround the while loop statements. This line does NOT have a semi-colon at the end.
12. The statements inside the while loop are indented to show that they are the body of the loop. Each statement ends in a semi-colon. **`System.out.println`** displays a line on the screen.
13. This statement adds 1 to the value of count, and then stores the result in the variable named count, thus increasing the value of count by 1.
14. The closing brace } ends the while loop structure. It does NOT have a semi-colon at the end.
15. This closing brace } ends the main program and returns control to the operating system. The comment (//) delimiters are just there for documentation. Comments are not seen by the compiler. This line does NOT have a semi-colon at the end.
16. This closing brace } ends the class definition. The comment (//) delimiters are just there for documentation. This line does NOT have a semi-colon at the end.

Example_2: Hello world program example

Specification: Write a program that prints out "Hello, world!"

Pseudocode:

Module main()
    Display "Hello, world!"
End Module

Note: The Java program below MUST be saved in a file named HelloWorld.java

```
/**
 * The HelloWorld class displays "Hello, world!" on the screen.
 * @author  Dr. K. Presnell
 */
class HelloWorld
{   // Opening and closing curly braces are used to surround a block of
    // code, including a class and a method.

    public static void main(String[] args)// In Java, you MUST have String[]
                                          // args as an argument for main.
    {
      System.out.println ("Hello, world!");  // This prints one line. Notice the ;
    }

}
```

Example_3: Employee program

Specification: Write a program that tracks the name and hourly pay rate for employees, and calculates the total pay when the number of hours worked is input.

```
Class Employee
    Private String empName
    Private Real hourlyPayRate

    Public Module Employee()
        Set empName = " "
        Set hourlyPayRate = 0
    End Module

    Public Module Employee(String newEmpName, Real newHourlyPayRate)
        Set empName = newEmpName
        Set hourlyPayRate = newHourlyPayRate
    End Module

    Public Module calculatePay(Real hoursWorked)
        Declare Real totalPay

        Set totalPay = hoursWorked * hourlyPayRate
        Display "Total pay amount is: ", totalPay
    End Module

    Public Funtion Real getHourlyPayRate()
        Return hourlyPayRate
    End Function
End Class

Module main()
    Declare Real hoursWorked
    Declare Employee john
    Set john = New Employee("John Jones", 12.50)

    Display "Please enter number of hours employee worked: "
    Input hoursWorked
    Call john.calculatePay(hoursWorked)
    Display “Pick up the check at payroll”
End Module
```

Note: The class below MUST be saved in a file named Employee.java

```
/**
 * The Employee class tracks the hourly pay rate and the name of an employee,
 * and it will calculate and print the total pay amount.
 * @author Dr. K. Presnell
 */
import java.text.NumberFormat;

class Employee
{
    /**
     * hourly pay rate for the employee
     */
    private double hourlyPayRate;

    /**
     * name of the employee
     */
    private String empName;

    /**
     * Default constructor
     */
    public Employee()
    {
      empName = " ";
      hourlyPayRate = 0;
    }

    /**
     * Constructor that initializes the data members
     * @param newEmpName - name of the employee
     * @param newHourlyPayRate - hourly pay rate for the employee
     */
    public Employee(String newEmpName, double newHourlyPayRate)
    {
      empName = newEmpName;
      hourlyPayRate = newHourlyPayRate;
    }

    /**
     * Calculates the total pay amount
     * @param hoursWorked - number of hours the employee worked this week
     */
    public void calculatePay(double hoursWorked)
    {
      double totalPay;  // total pay amount
      NumberFormat currency = NumberFormat.getCurrencyInstance(); // for formatting

      totalPay = hoursWorked * hourlyPayRate;
      System.out.println ("Total pay amount is: " + currency.format(totalPay));
    }
    /**
     * Returns the hourly pay rate
     * @return hourly pay rate
     */
    public double getHourlyPayRate()
    {
      return hourlyPayRate;
    }
}
```

Note: The program below MUST be saved in a file named EmployeePay.java

```
/**
 * The EmployeePay class creates and initializes an employee, inputs the
 * number of hours worked, and displays the total pay amount.
 * @author Dr. K. Presnell
 */

import java.util.Scanner;

class EmployeePay
{
    public static void main(String[] args)
    {
      double hoursWorked;  // number of hours worked
      Employee john = new Employee("John Jones", 12.50);
                                              // create the object for john
      Scanner scan = new Scanner(System.in);
                                // create a scan object to be used for input

      System.out.print ("Please enter number of hours employee worked: ");
      hoursWorked = scan.nextDouble();
      john.calculatePay(hoursWorked);
      System.out.println ("Pick up the check at payroll");
    }
}
```

Output:

```
Please enter number of hours employee worked: 20
Total pay amount is: $250.00
Pick up the check at payroll
```

Example_4: If/Else (Selection)

Specification: Add a method to the Employee class that determines the amount of an employee's bonus, based on the hourly pay rate. If the hourly pay rate is less than $8.50, then the bonus will be $100.00; if the hourly pay rate is $8.50 or more, but less than $12.00, then the bonus will be $150.00; if the hourly pay rate is $12.00 or more, but less than $18.00, then the bonus will be $200.00; and if the hourly pay rate is $18.00 or more, the bonus will be $250.00.

```
Class EmployeeBonus
    Private String empName
    Private Real hourlyPayRate

    Public Module EmployeeBonus()
        Set empName = " "
        Set hourlyPayRate = 0
    End Module

    Public Module EmployeeBonus(String newEmpName, Real newHourlyPayRate)
        Set empName = newEmpName
        Set hourlyPayRate = newHourlyPayRate
    End Module

    Public Module calculatePay(Real hoursWorked)
        Declare Real totalPay

        Set totalPay = hoursWorked * hourlyPayRate
        Display "Total pay amount is: ", totalPay
    End Module

    Public Module calculateBonus()
        Declare Real bonus

        If hourlyPayRate < 8.50 Then
            Set bonus = 100.00
        Else If hourlyPayRate < 12.00 Then
            Set bonus = 150.00
        Else If hourlyPayRate < 18.00 Then
            Set bonus = 200.00
        Else
            Set bonus = 250.00
        End If
        Display "Total bonus amount is:  ", bonus
    End Module

    Public Function Real getHourlyPayRate()
        Return hourlyPayRate
    End Function
End Class

Module main()
    Declare Real hoursWorked
    Declare EmployeeBonus john
    Set john = New EmployeeBonus("John Jones", 12.50)

    Display "Please enter number of hours employee worked: "
    Input hoursWorked
    Call john.calculatePay(hoursWorked)
    Call john.calculateBonus()
    Display "Pick up the check at payroll"
End Module
```

Note: The class below MUST be saved in a file named EmployeeBonus.java

```
/**
 * The EmployeeBonus class tracks the hourly pay rate and the name of an
 * employee, and it will calculate and print the total pay amount and the
 * amount of the bonus.
 * @author Dr. K. Presnell
 */
import java.text.NumberFormat;

class EmployeeBonus
{
    /**
     * hourly pay rate for the employee
     */
    private double hourlyPayRate;

    /**
     * name of the employee
     */
    private String empName;

    /**
     * Default constructor
     */
    public EmployeeBonus()
    {
      empName = " ";
      hourlyPayRate = 0;
    }

    /**
     * Constructor that initializes the data members
     * @param newEmpName - name of the employee
     * @param newHourlyPayRate - hourly pay rate for the employee
     */
    public EmployeeBonus(String newEmpName, double newHourlyPayRate)
    {
      empName = newEmpName;
      hourlyPayRate = newHourlyPayRate;
    }
    /**
     * Calculates the total pay amount
     * @param hoursWorked - number of hours the employee worked this week
     */
    public void calculatePay(double hoursWorked)
    {
      double totalPay;  // total pay amount
      NumberFormat currency = NumberFormat.getCurrencyInstance();  // for formatting

      totalPay = hoursWorked * hourlyPayRate;
      System.out.println ("Total pay amount is: " + currency.format(totalPay));
    }
```

```
    /**
     * Calculates the amount of the employee bonus
     */
    public void calculateBonus()
    {
      double bonus;  // amount of employee bonus
      NumberFormat currency = NumberFormat.getCurrencyInstance();
                                                              // for formatting

      if (hourlyPayRate < 8.50)
         bonus = 100.00;
      else if (hourlyPayRate < 12.00)
         bonus = 150.00;
      else if (hourlyPayRate < 18.00)
         bonus = 200.00;
      else
         bonus = 250.00;
      System.out.println ("Total bonus amount is: " + currency.format(bonus));
    }

    /**
     * Returns the hourly pay rate
     * @return hourly pay rate
     */
    public double getHourlyPayRate()
    {
      return hourlyPayRate;
    }
}
```

Note: The program below MUST be saved in a file named EmployeeBonusPay.java

```
/**
 * The EmployeeBonusPay class creates and initializes an employee, inputs the
 * number of hours worked, and displays the total pay amount and the bonus.
 * @author Dr. K. Presnell
 */

import java.util.Scanner;

class EmployeeBonusPay
{
    public static void main(String[] args)
    {
      double hoursWorked;  // number of hours worked
      EmployeeBonus john =
         new EmployeeBonus("John Jones", 12.50);
                                              // create the object for john
      Scanner scan = new Scanner(System.in);
                             // create a scan object to be used for input

      System.out.print ("Please enter number of hours employee worked: ");
      hoursWorked = scan.nextDouble();
      john.calculatePay(hoursWorked);
      john.calculateBonus();
      System.out.println ("Pick up the check at payroll");
    }
}
```

Output:

```
Please enter number of hours employee worked: 20
Total pay amount is: $250.00
Total bonus amount is: $200.00
Pick up the check at payroll
```

Example_5: Case (Selection)

Specification: Add an attribute to the Employee class that indicates the type of employee. An 'M' indicates a manager, an 'S' for a sales person, and a 'C' is for a clerk. Change the calculateBonus method that determines the amount of an employee's bonus, based on the type of employee. If the employee is a clerk the bonus will be $150.00; if the employee is a sales person, then the bonus will be $200.00; and if the employee is a manager, the bonus will be $250.00.

```
Class EmployeeType
    Private String empName
    Private Real hourlyPayRate
    Private Character empType

    Public Module EmployeeType()
        Set empName = " "
        Set hourlyPayRate = 0
        Set empType = ' '
    End Module

    Public Module EmployeeType(String newEmpName, Real newHourlyPayRate, Character newEmpType)
        Set empName = newEmpName
        Set hourlyPayRate = newHourlyPayRate
        Set empType = newEmpType
    End Module

    Public Module calculatePay(Real hoursWorked)
        Declare Real totalPay

        Set totalPay = hoursWorked * hourlyPayRate
        Display "Total pay amount is: ", totalPay
    End Module

    Public Module calculateBonus()
        Declare Real bonus

        Select empType
            Case 'C':
                Set bonus = 150.00
            Case 'S':
                Set bonus = 200.00
            Case 'M':
                Set bonus = 250.00
            Sefault:
                Set bonus = 0.0
                Display "Error in employee type"
        End Select
        Display "Total bonus amount is:  ", bonus
    End Module

    Public Function Real getHourlyPayRate()
        Return hourlyPayRate
    End Function
End Class
```

```
Module main()
    Declare Real hoursWorked
    Declare EmployeeType john
    Set john = New EmployeeType("John Jones", 12.50, 'S')

    Display "Please enter number of hours employee worked: "
    Input hoursWorked
    Call john.calculatePay(hoursWorked)
    Call john.calculateBonus()
    Display "Pick up the check at payroll"
End Module
```

Note: The class below MUST be saved in a file named EmployeeType.java

```
/**
 * The EmployeeType class tracks the hourly pay rate, the name of an employee,
 * and the employee type (C for clerk, S for salesperson, M for manager).
 * It calculates and prints the total pay amount and the amount of the bonus.
 * @author Dr. K. Presnell
 */

import java.text.NumberFormat;

class EmployeeType
{
    /**
     * hourly pay rate for the employee
     */
    private double hourlyPayRate;

    /**
     * name of the employee
     */
    private String empName;

    /**
     * type of employee
     */
    private char empType;

    /**
     * Default constructor
     */
    public EmployeeType()
    {
      empName = " ";
      hourlyPayRate = 0;
      empType = ' ';
    }
```

```
    /**
     * Constructor that initializes the data members
     * @param newEmpName - name of the employee
     * @param newHourlyPayRate - hourly pay rate for the employee
     * @param newEmpType - type of employee, either C, S, or M
     */
    public EmployeeType(String newEmpName, double newHourlyPayRate,
                        char newEmpType)
    {
      empName = newEmpName;
      hourlyPayRate = newHourlyPayRate;
      empType = newEmpType;
    }

    /**
     * Calculates the total pay amount
     * @param hoursWorked - number of hours the employee worked this week
     */
    public void calculatePay(double hoursWorked)
    {
      double totalPay;  // total pay amount
      NumberFormat currency = NumberFormat.getCurrencyInstance();// for formatting
      totalPay = hoursWorked * hourlyPayRate;
      System.out.println ("Total pay amount is: "+ currency.format(totalPay));
    }

    /**
     * Calculates the amount of the employee bonus, based on employee type
     */
    public void calculateBonus()
    {
      double bonus = 25.0; // amount of employee bonus
      NumberFormat currency = NumberFormat.getCurrencyInstance();// for formatting

      switch (empType)
      {
         case 'C':
            bonus = 150.00;
            break;
         case 'S':
            bonus = 200.00;
            break;
         case 'M':
            bonus = 250.00;
            break;
         default:
            bonus = 0.00;
            System.out.println ("Error in employee type: " + empType);
            break;
      }
      System.out.println ("Total bonus amount is: " + currency.format(bonus));
    }

    /**
     * Returns the hourly pay rate
     * @return hourly pay rate
     */
    public double getHourlyRate()
    {
      return hourlyPayRate;
    }
}
```

Note: The class below MUST be saved in a file named EmployeeTypePay.java

```
/**
 * The EmployeeTypePay class creates and initializes an employee, inputs the
 * number of hours worked, and displays the total pay amount and bonus amount.
 * @author Dr. K. Presnell
 */

import java.util.Scanner;

class EmployeeTypePay
{
    public static void main(String[] args)
    {
      double hoursWorked;  // number of hours worked
      EmployeeType john =
         new EmployeeType("John Jones", 12.50, 'S');// create object for john
      Scanner scan = new Scanner(System.in); // create a scan object to be
                                             // used for input

      System.out.print ("Please enter number of hours employee worked: ");
      hoursWorked = scan.nextDouble();
      john.calculatePay(hoursWorked);
      john.calculateBonus();
      System.out.println ("Pick up the check at payroll");
      return;
    }
}
```

Output:

```
Please enter number of hours employee worked: 50
Total pay amount is: $625.00
Total bonus amount is: $200.00
Pick up the check at payroll
```

Example_6: While (Repetition)

Specification: Write a program that tracks the id number, last name, first name, and hourly rate, and last weeks production for personnel. The program must print labels for an employee, printing 10% more labels than the employee produced last week.

```
Class Personnel
    Private String idNumber
    Private String lastName
    Private String firstName
    Private Real hourlyRate
    Private Integer lastWeeksProduction

    Public Module Personnel()
        Set idNumber = " "
        Set lastName = " "
        Set firstName = " "
        Set hourlyRate = 0
        Set lastWeeksProduction = 0
    End Module

    Public Module Personnel (String newIdNumber, String newLastName, String newFirstName,
                             Real newHourlyRate, Integer newLastWeeksProduction)
        Set idNumber = newIdNumber
        Set lastName = newLastName
        Set firstName = newFirstName
        Set hourlyRate = newHourlyRate
        Set lastWeeksProduction = newLastWeeksProduction
    End Module

    Public Module initialize ()
        Display "Enter the employee's id number:"
        Input idNumber
        Display "Enter the employee's last name:"
        Input lastName
        Display "Enter the employee's first name:"
        Input firstName
        Display "Enter the employee's hourly pay rate:"
        Input hourlyRate
        Display "Enter the number of items the employee produced last week:"
        Input lastWeeksProduction
    End Module

    Public Function String getIdNumber()
        Return idNumber
    End Function

    Public Module setIdNumber (string newIdNumber)
        Set idNumber = newIdNumber
    End Module

    Public Function String getLastName()
        Return lastName
    End Function
```

```
    Public Module setLastName (string newLastName)
        Set lastName = newLastName
    End Module

    Public Function String getFirstName()
        Return firstName
    End Function

    Public Module setFirstName (string newFirstName)
        Set firstName = newFirstName
    End Module

    Public Function Real getHourlyRate()
        Return hourlyRate
    End Function

    Public Module setHourlyRate (numeric newHourlyRate)
        Set hourlyRate = newHourlyRate
    End Module

    Public Function Integer getLastWeeksProduction()
        Return lastWeeksProduction
    End Function

    Public Module setLastWeeksProduction (numeric newLastWeeksProduction)
        Set lastWeeksProduction = newLastWeeksProduction
    End Module
End Class


Module main()
    Declare Personnel aWorker
    Set Worker = New Personnel()
    Call aWorker.initialize()
    Call createLabels (aWorker)
End Module

Module createLabels(Personnel per)
    Declare Integer num
    Declare Integer labelsToPrint
    Set num = 0
    Set labelsToPrint = per.getLastWeeksProduction() * 1.10
    While num !=  labelsToPrint
        Display "Made for you personally by ", per.getFirstName()
        Set num = num + 1
    End While
End Module
```

Note: The class below MUST be saved in a file named Personnel.java

```
/**
 * The Personnel class stores data for an employee and initializes an
 * employee with user input.
 * @author Dr. K. Presnell
 */
import java.util.Scanner;

class Personnel
{
    /**
     * id number for employee
     */
    private String idNumber;

    /**
     * last name for the employee
     */
    private String lastName;

    /**
     * first name for the employee
     */
    private String firstName;

    /**
     * hourly pay rate for the employee
     */
    private double hourlyRate;

    /**
     * number of items employee produced last week
     */
    private int lastWeeksProduction;


    /**
     * Default constructor
     */
    public Personnel()
    {
      idNumber = "";
      lastName = "";
      firstName = "";
      hourlyRate = 0;
      lastWeeksProduction = 0;
    }
```

```
/**
 * Constructor that initializes the data members
 * @param newIdNumber - id number of the employee
 * @param newLastName - last name of the employee
 * @param newFirstName - first name of the employee
 * @param newHourlyRate - hourly pay rate for the employee
 * @param newLastWeeksProduction - number of items employee produced last week
 */
public Personnel (String newIdNumber, String newLastName, String newFirstName,
                  double newHourlyRate, int newLastWeeksProduction)
{
  idNumber = newIdNumber;
  lastName = newLastName;
  firstName = newFirstName;
  hourlyRate = newHourlyRate;
  lastWeeksProduction = newLastWeeksProduction;
}


/**
 * Initializes all the private data members by getting input from the user
 */
public void initialize ()
{
  Scanner scan = new Scanner(System.in);

  System.out.print ("Enter the employee's id number:  ");
  idNumber = scan.nextLine();
  System.out.print ("Enter the employee's last name:  ");
  lastName = scan.nextLine();
  System.out.print ("Enter the employee's first name:  ");
  firstName = scan.nextLine();
  System.out.print ("Enter the employee's hourly pay rate:  ");
  hourlyRate = scan.nextDouble();
  System.out.print
        ("Enter the number of items the employee produced last week:  ");
  lastWeeksProduction = scan.nextInt();
}


/**
 *  Returns the id number of the employee
 *  @return id number of employee
 */
public String getIdNumber()
{
  return idNumber;
}


/**
 *  Sets the id number of the employee
 *  @param newIdNumber - new id number of employee
 */
public void setIdNumber (String newIdNumber)
{
  idNumber = newIdNumber;
}
```

```
/**
 *  Returns the last name of the employee
 *  @return last name of employee
 */
public String getLastName()
{
  return lastName;
}


/**
 *  Sets the last name of the employee
 *  @param newLastName - new last name of employee
 */
public void setLastName (String newLastName)
{
  lastName = newLastName;
}


/**
 *  Returns the first name of the employee
 *  @return first name of employee
 */
public String getFirstName()
{
  return firstName;
}


/**
 *  Sets the first name of the employee
 *  @param newFirstName - new first name of employee
 */
public void setFirstName (String newFirstName)
{
  firstName = newFirstName;
}


/**
 *  Returns the hourly pay rate for the employee
 *  @return hourly pay rate for employee
 */
public double getHourlyRate()
{
  return hourlyRate;
}


/**
 *  Sets the hourly pay rate for the employee
 *  @param newHourlyRate - new hourly pay rate for employee
 */
public void setHourlyRate (double newHourlyRate)
{
  hourlyRate = newHourlyRate;
}
```

```
    /**
     *  Returns the number of items the employee produced last week
     *  @return number of items employee produced last week
     */
    public int getLastWeeksProduction()
    {
      return lastWeeksProduction;
    }


    /**
     *  Sets the number of items the employee produced last week
     *  @param newLastWeeksProduction - new number of items employee produced
     *                                  last week
     */
    public void setLastWeeksProduction (int newLastWeeksProduction)
    {
      lastWeeksProduction = newLastWeeksProduction;
    }
}
```

Note: The class below MUST be saved in a file named CreateLabels.java

```
/**
 * The CreateLabels class prints labels for one worker. It prints 10% more
 * labels than the worker needed last week.
 * @author Dr. K. Presnell
 */
class CreateLabels
{
    public static void main(String[] args)
    {
      Personnel aWorker = new Personnel();
      aWorker.initialize();
      createLabels (aWorker);
    }


    public static void createLabels(Personnel per)
    {
      int num;
      int labelsToPrint;
      num = 0;
      labelsToPrint = (int)(per.getLastWeeksProduction() * 1.10);
      System.out.println();
      System.out.println ("Printing " + labelsToPrint + " labels:");
      while (num != labelsToPrint)
      {
         System.out.println ("Made for you personally by " + per.getFirstName());
         num = num + 1;
      }
    }
}
```

Output:

```
Enter the employee's id number:  12345
Enter the employee's last name:  Jones
Enter the employee's first name:  John
Enter the employee's hourly pay rate:  54.25
Enter the number of items the employee produced last week:  10

Printing 11 labels.
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
Made for you personally by John
```