ASSIGNMENT 1

CPE 102 Introduction to Programming

LIM REAMSOVICHEA

HP053246

FE 12

Date: September 26, 2005

REPORT

Program 1

Purpose: the program is written to calculate the actual cost of buying a car in Singapore.

Program Analysis:

Program inputs:

- the price of the car
- the category of the car

Program outputs:

- the actual cost of buying a car which is included in taxes and so on

Formulas:

- discounted price = price of the car * 80%
- GST = discounted price * 5%
- luxury tax = (discounted price – 100000) * 10% if discounted price > 100000
  = 0 if discounted price < = 100000
- actual cost = discounted price + GST + luxury tax + COE

Program Design:

Initial Algorithm

1. Read the price and category of a car
2. Compute the discounted price, GST, luxury tax, actual cost
3. Print the actual cost

Refinement in Pseudocode

main:

```
SET COE TO 0.0
DO
        READ carPrice
        IF carPrice <= 0.0
                PRINT “Please input the positive price.”
WHILE carPrice <=  0.0


DO
        READ category

        IF category = 1
                SET COE TO 20000.0
        ELSE IF category = 2
                SET COE TO 18000.0
        ELSE IF category = 3
                SET COE TO 10000.0
        ELSE IF category = 4
                SET COE TO 500.0
        ELSE
                PRINT "You chose the wrong category."
        ENDIF

WHILE category < 1 OR category > 4

COMPUTE discountedPrice = carPrice * 0.8
```

```
COMPUTE GST = 0.05 * discountedPrice
IF discountedPrice > 100000.0
        COMPUTE luxuryTax = 0.1 * (discountedPrice - 100000.0)
ELSE
        SET luxuryTax TO 0.0
END IF

COMPUTE actualCost = discountedPrice + GST + luxuryTax + COE
PRINT actualCost
```

Program logic:

- Since we need the check the error whether the user enters the correct price in positive value, we have to use loop. The reason we use "*do-while*" loop is that we need the loop to be done at least once.
- Again, we use "*do-while*" loop to perform the action at least once for the category of the car. Moreover, we use selection "*switch*" rather than "*if-else*" because it is more convenient since the arguments of condition are integers: 1, 2, 3 and 4.
- We use luxury tax to include in the actual cost formula because we already determine the condition to luxury tax no matter the discounted price is over $100,000 or not.

Program listing:

```
/* This is the program to calculate the actual cost of buying a car in Singapore.
 *     Author: Lim Reamsovichea
 *     Date:   23 September 2005
 */


import java.util.Scanner; // imported for class Scanner
import java.text.DecimalFormat; // imported for class DecimalFormat


public class P1
{
        //Start of method main
        public static void main(String[] args)
        {
                // Local Definitions
                Scanner sc = new Scanner(System.in); // create object sc

                DecimalFormat dec = new DecimalFormat(",##0.00"); // create object dec

                int category;

                double carPrice, discountedPrice, actualCost, GST, luxuryTax;

                double COE = 0.0;

                // Statements

                do // this loop is to check whether user enter the appropriate price
                {

                        System.out.print("Please enter the car price: "); // prompt
                        carPrice = sc.nextDouble(); // read the original car price
```

```
            if(carPrice <= 0.0)
                System.out.println("Please enter the positive price.");

        }while(carPrice <= 0.0); // end of loop

        do // this loop is to check whether user chooses the correct category
        {
            System.out.print("Please choose the category: "); // prompt
            category = sc.nextInt(); // read the category of the car


            switch(category)
            {
                case 1: COE = 20000.0; break;
                case 2: COE = 18000.0; break;
                case 3: COE = 10000.0; break;
                case 4: COE = 500.0;   break;

                default:System.out.println("You chose the wrong category.");


            } // end of switch

        }while(category < 1 || category > 4);//end of loop


        discountedPrice = carPrice * 0.8; // car dealer discount 20%

        GST = 0.05 * discountedPrice; // tax of 5% on discounted price

        if(discountedPrice > 100000.0) // luxury tax is accounted

            // tax of 10% on the excess amount over $100,000
            luxuryTax = 0.1 * (discountedPrice - 100000.0);

        else

            // luxury tax is not accounted since discounted price < $100,000
            luxuryTax = 0.0;


        actualCost = discountedPrice + GST + luxuryTax + COE;

        System.out.println();

        System.out.println("The actual cost of buying a car is $" + dec.format(actualCost));


    } // end method main

} // end class P1
```

Program Testing:

```
[hp053246@c193 as1Xp]$ java P1
Please enter the car price: 35000
```

```
Please choose the category: 1

The actual cost of buying a car is $49,400.00

[hp053246@c193 as1Xp]$ java P1
Please enter the car price: 150000
Please choose the category: 2

The actual cost of buying a car is $146,000.00

[hp053246@c193 as1Xp]$ java P1
Please enter the car price: 30000
Please choose the category: 3

The actual cost of buying a car is $35,200.00

[hp053246@c193 as1Xp]$ java P1
Please enter the car price: 5000
Please choose the category: 4

The actual cost of buying a car is $4,700.00

[hp053246@c193 as1Xp]$ java P1
Please enter the car price: 0
Please enter the positive price.
Please enter the car price: 10000
Please choose the category: 8
You chose the wrong category.
```

Program 2

Purpose: This is a program to read in a list of non-negative integers and print the total counts of odd and even numbers as well as their averages

Program Analysis:

Program inputs:

- integers

Program outputs:

- the number of even integers
- the number of odd integers
- the average of even integers
- the average of odd integers

Formulas:

- count of even = count of even +1 while input is even
- sum of even = sum of even + integer while input is even
- count of odd = count of odd +1 while input is odd
- sum of odd = sum of odd + integer while input is odd
- average of even = sum of even / count of even
- average of odd = sum of even / count of odd

Program Design:

Initial Algorithm

1. Read the integers
2. Compute the count of even, count of odd, sum of even, sum of odd, average of even and average of odd
3. Print count of even, count of odd, average of even and average of odd

Refinement in Pseudocode

```
main:
        SET countEven, countOdd TO 0
        SET sumEven, sumOdd TO 0

        DO
                READ inputNum

                IF inputNum >= 0
                        IF inputNum % 2 = 0
                                INCREMENT countEven BY 1
                                COMPUTE sumEven = sumEven + inputNum
                        ELSE
                                INCREMENT countOdd BY 1
                                COMPUTE sumOdd = sumOdd + inputNum
                        ENDIF
                ENDIF
                IF inputNum < -1
                        PRINT "Negative integer will be ignored."
                ENDIF

        WHILE inputNum != -1


        IF countEven != 0
                COMPUTE averageEven = sumEven / countEven
        ELSE
                SET averageEven = 0.0
        ENDIF


        IF countOdd != 0
                COMPUTE averageOdd = sumOdd / countOdd
        ELSE
                SET averageOdd = 0.0
        ENDIF
        PRINT countEven, countOdd, averageEven, averageOdd
```

Program logic:

- We use sentinel-controlled loop "*do-while*" because we do not know when the user wants to stop their inputs; moreover, we want the loop to perform the task at least once.
- In the loop, we check the condition whether the input number is non-negative, and we do not take into account for negative integers.
- For the nested "*if*", we check according condition to calculate the count of even and sum of even as well as for odd number.
- For the last part, we use the condition to check whether count of even because if there is no even inputs then average of even become zero divided by zero. To prevent this, we set average of even to zero when count of even is zero. In the same way, it also applies to count of odd.

Program listing:

```
/* This is a program to read in a list of non-negative integers and print
 * the total counts of odd and even numbers as well as their averages.
 *     Author: Lim Reamsovichea
```

```
*     Date: 23 September 2005
*/


import java.util.Scanner; // imported for class Scanner
import java.text.DecimalFormat; // imported for class DecimalFormat

public class P2
{
	// Start of method main
	public static void main(String[] args)
	{
		// Local Definitions

		Scanner sc = new Scanner(System.in); // create object sc

		DecimalFormat dec = new DecimalFormat("0.000"); // create object dec

		int inputNum;
		int countEven = 0, countOdd = 0;
		int sumEven = 0, sumOdd = 0;

		double averageEven, averageOdd;


		// Statements

		do // the loop will ignore the negative input
		{
			System.out.print("Please enter the integer: "); // prompt
			inputNum = sc.nextInt(); // read integer for next input

			if(inputNum >= 0) // accept non-negative integer only
			{

				if(inputNum % 2 == 0) //input number is even
				{
					countEven++;
					sumEven += inputNum;
				}

				else // input number is odd
				{
					countOdd++;
					sumOdd += inputNum;

				} // end nested "if-else"

			} // end "if"

			if(inputNum < -1) System.out.println("Negative integer will be ignored.");


		} while(inputNum != -1); // the loop will be terminated when inputNum = -1


		/* the condition is used because it prevents the case there is no input
		 * of even or odd number which draws the result of zero divided by zero
```

```
                *
                * sumX and countX are needed to
                *declare as double since they are integers
                */


                if(countEven != 0) averageEven = (double)sumEven / countEven;
                else averageEven = 0.0;

                if(countOdd != 0) averageOdd = (double)sumOdd / countOdd;
                else averageOdd = 0.0;


                System.out.println("The number of even integers is " + countEven);
                System.out.println("The number of odd integers is " + countOdd);
                System.out.println("The average of even integers is " + dec.format(averageEven));
                System.out.println("The average of odd integers is " + dec.format(averageOdd));

        }// end method main

}// end class P2
```

Program Testing:

```
[hp053246@c157 assign1]$ java P2
Please enter the integer: 1
Please enter the integer: 3
Please enter the integer: 5
Please enter the integer: -1
The number of even integers is 0
The number of odd integers is 3
The average of even integers is 0.000
The average of odd integers is 3.000

[hp053246@c157 assign1]$ java P2
Please enter the integer: 2
Please enter the integer: 4
Please enter the integer: 6
Please enter the integer: 8
Please enter the integer: 10
Please enter the integer: -1
The number of even integers is 5
The number of odd integers is 0
The average of even integers is 6.000
The average of odd integers is 0.000

[hp053246@c157 assign1]$ java P2
Please enter the integer: 1
Please enter the integer: 5
Please enter the integer: 4
Please enter the integer: 8
Please enter the integer: 24
Please enter the integer: 56
Please enter the integer: 89
Please enter the integer: 76
Please enter the integer: 6
Please enter the integer: -1
The number of even integers is 6
The number of odd integers is 3
The average of even integers is 29.000
The average of odd integers is 31.667
```

Program 3

Purpose: This is a program that asks the user to input four non-negative numbers and draws the corresponding bar chart that is displayed vertically on the screen.

Program Analysis:

Program inputs:

- the four non-negative numbers

Program outputs:

- corresponding bar chart that is vertically displayed on the screen

Formulas:

- find the maximum among the four numbers: if max < num then max = num
- while max >= 1 use charBuilder(num, max)

Program Design:

Initial Algorithm

1. Read the four non-negative numbers
2. Compute the maximum value among the four numbers
3. Print stars or spaces in line by line according to the loop condition

Refinement in Pseudocode

```
main:
    READ num1, num2, num3, num4
    ASSIGN max TO num1
    IF max < num2
        ASSIGN max TO num2
    ENDIF

    IF max < num3
        ASSIGN max TO num3
    ENDIF

    IF max < num4
        ASSIGN max TO num4
    ENDIF

    WHILE max >= 1
        CALL chartBuilder(num1, max)
        CALL chartBuilder(num2, max)
        CALL chartBuilder(num3, max)
        CALL chartBuilder(num4, max)

        PRINT "\n"
        DECREMENT max BY 1
    ENDWHILE

chartBuilder(num, max):

    IF num >= max
        PRINT "**"
    ELSE
        PRINT "  "
    ENDIF
```

Program logic

- We find the maximum value among them because we want to determine the number of the loops which depend on the highest height of the bars
- We use loop to print the stars or spaces from the top line to bottom line. In this case, we use the counter-controlled loop because we know the number of the loops which will be executed.
- In the loop, we use the method chartBuilder because it is convenient and easy to understand as well as efficient in other cases.

Program listing:

```
/* This is a program that asks the user to input four non-negative numbers
 * and draws the corresponding bar chart that is displayed vertically on
 * the screen.
 *
 *     Author: Lim Reamsovichea
 *     Date:   23 September 2005
 */


import java.util.Scanner; // imported for class Scanner

public class P3
{
        // Start of method main
        public static void main(String[] args)
        {
                // Local Definitions

                Scanner sc = new Scanner(System.in); // creat object sc

                int num1, num2, num3, num4; // the four input numbers
                int max; // maximum of the four input numbers


                // Statements

            System.out.print("Please enter the four positive numbers: "); // prompt

            num1 = sc.nextInt();
            num2 = sc.nextInt();
            num3 = sc.nextInt();
            num4 = sc.nextInt(); // read the positive integers from users

            // find the maximum amoung num1, num2, num3, and num4

            max = num1;

                if (max < num2) max = num2;
                if (max < num3) max = num3;
                if (max < num4) max = num4;

                //this loop is used to print the lines from the top to bottom
                for( ; max >= 1; max--)
                {

                        chartBuilder(num1, max);
                        chartBuilder(num2, max);
```

```
                chartBuilder(num3, max);
                chartBuilder(num4, max); // to print each columns in the same line

                System.out.println(); // enter the new line when one line is completed



            } // end of loop

        } // end method main


        // method chartBuilder is to print stars or spaces in a column of one line
        public static void chartBuilder(int mum, int max)
        {
                if (mum >= max) System.out.print("**");
                else            System.out.print("  ");

        }//end method chartBuilder

}// end class P3
```

Program Testing:

```
[hp053246@c157 assign1]$ java P3
Please enter the four positive number: 5 4 3 2
**
****
******
********
********

[hp053246@c157 assign1]$ java P3
Please enter the four positive number: 2 3 4 5
      **
    ****
  ******
********
********

[hp053246@c157 assign1]$ java P3
Please enter the four positive number: 5 3 4 2
**
**  **
******
********
********

[hp053246@c157 assign1]$ java P3
Please enter the four positive number: 2 4 5 3
    **
  ****
  ******
********
********
```

Program 4

Purpose: This is the program that accepts a number and after that it reverses the digits on either side of the middle number.

Program Analysis:

Program inputs:

- a non-negative integer

Program outputs:

- an integer which reverses the digits on either side of the middle number

Formulas:

- use loop to count the number of digits of an integer
- the half end of an integer = value % 10 powered (number of digits / 2)
- the half front of an integer = value / 10 powered (number of digits / 2) if digits is even
  = value / 10 powered (number of digits / 2 +1) if digits is odd
- call another method to reverse the first and last digits of half front and half end
- middle digit = 0 if digits is even
  = (value / 10 powered (number of digits / 2)) % 10
- value = (halfFront * powTenFront) + (middleDigit * powTenEnd) + halfEnd

Program Design:

Initial Algorithm

1. Read an non-negative integer
2. Separate the input number into two parts: half front and half end. Find middle number if the number of digits of the input number is odd
3. Create a method that exchanges the first and last digits of a number
4. Call the exchanged method to change the half front and half end
5. Combine the new half front and half end to form the new number
6. Print the new number

Refinement in Pseudocode

```
main:
    READ value
    COMPUTE result = reverseSideDigits(value)
    PRINT result

reverseSideDigits(value):

    SET digits TO 0
    SET powTenFront TO 1
    SET powTenEnd TO 1

    ASSIGN temp = value

    WHILE temp != 0

        INCREMENT digits BY 1
        COMPUTE temp = temp / 10

        IF digits % 2 = 0
            COMPUTE powTenEnd = powTenEnd * 10
        ELSE
            COMPUTE powTenFront = powTenFront * 10
        ENDIF
```

```
    ENDWHILE

    IF digits <= 3
        RETURN value
    ELSE
        COMPUTE halfEnd = value % powTenEnd
        COMPUTE halfEnd = firstLastExchange(halfEnd, digits / 2, powTenEnd / 10)

        COMPUTE halfFront = value / powTenFront
        COMPUTE halfFront = firstLastExchange(halfFront, digits / 2, powTenEnd / 10)

        IF digits % 2 = 0
            SET middleDigit TO 0
        ELSE
            COMPUTE middleDigit = (value / powTenEnd) % 10
        ENDIF
        RETURN value
    ENDIF


firstLastExchange( n, digits, powTen ):

    COMPUTE first = n % 10
    COMPUTE last = n / powTen

    COMPUTE n = (n % powTen) / 10

    COMPUTE n = first * powTen + 10 * n + last

    RETURN n
```

Program logic:

- In reverseSidesDigits method, we use loop to count the number of digits of input value. Meanwhile, we try to find the power 10 in order to separate the half front and half end of the value. Finding the powTenFront and powTenEnd simultaneously in the loop is the best way to cope with the recombining of the new value.
- Since purpose is to reverse the digits on either side of the middle number, then we have to separate the input into two parts, but since we do not know whether the user enters the odd or even number of digits of integer, we use condition to determine the middle digit.
- After we separate the input into two parts, namely half front and half end, we call the firstLastExchange method to exchange the first and last digits of the half front and half end. Next, we combine the new half front and half end into a new number.

Program listing:

```
/* This is the program that accepts a number and after that it reverses the
 * digits on either side of the middle number.
 *     Author: Lim Reamsovichea
 *     Date:   23 September 2005
 */
```

```
import java.util.Scanner;//imported for class Scanner

public class P4
{
	// Start of method main
	public static void main(String[] args)
	{
		// Local Defitions

		Scanner sc = new Scanner(System.in);// create object sc

		long value, result;

		// Statements

		System.out.print("Please enter a non-positive integer: ");//prompt
		value = sc.nextLong();//read integer from user

		result = reverseSideDigits(value); // call the method to reverse

		System.out.println("The reverse of " + value + " is " + result);


	}//end method main


	// this method will reverse the digits on either side of the middle number
	public static long reverseSideDigits(long value)
	{
		// Local Definitions
		int digits = 0;
		long halfFront, halfEnd, middleDigit;
		long temp = value; // value is reserved for returning purpose
		long powTenEnd = 1; // power ten to separate the half end of value
		long powTenFront = 1; // power ten to separate the half front of value

		// find the number of digits of input value
		while(temp != 0)
		{
			digits++;

			temp /= 10;


			/* powTenEnd is 10 powered (digits/2) times to separate the
			 * half end of value. powTenEnd and powTenFront are the same
			 * when the final digits is even.
			 */

			if (digits % 2 == 0) powTenEnd *= 10;
			else powTenFront *= 10;

		}// end of while


		if (digits <= 3) return value;// return the same value
```

```
        else // number of digits of value are more than 3 digits
        {
                halfEnd = value % powTenEnd;//take out the half end of digits

                // reverse half end of value by exchanging the first and last digits
                halfEnd = firstLastExchange(halfEnd, digits / 2, powTenEnd / 10);


                halfFront = value / powTenFront; // take out the half front of digits

                // reverse half front of value by exchanging the first and last digits

                halfFront = firstLastExchange(halfFront, digits / 2, powTenEnd / 10);



                /* if the number of digits of value is even, then there is no middle
                 * digit; hence we set the middle digit offset to zero.
                 */

                if (digits % 2 == 0) middleDigit = 0; // there is no middle number

                else middleDigit = (value / powTenEnd) % 10; // extract middle digit


                // combine the new half front and half end to form new value

                value = (halfFront * powTenFront) + (middleDigit * powTenEnd) + halfEnd;

                return value;// return new value to reverseSideDigits in long


        }//end "if" to check number of digits of value is more than one digit

}//end method reserseSideDigits



// this method will exchaneg the first and last digits of a number
public static long firstLastExchange(long n, int digits, long powTen)
{
        long first, last;


        first = n % 10; // the first digit of new n
        last = n / powTen; // the last digit of new n

        n = (n % powTen) / 10; // middle part of original n

        //the new value of n
        n = first * powTen + 10 * n + last;


        return n;
```

```
    }//end method firstLastExchange

}//end class P4
```

Program Testing:

```
[hp053246@c157 assign1]$ java P4
Please enter a positive integer: 1
The reverse of 1 is 1

[hp053246@c157 assign1]$ java P4
Please enter a positive integer: 45
The reverse of 45 is 45

[hp053246@c193 as1Xp]$ java P4
Please enter a non-positive integer: 4589
The reverse of 4589 is 5498

[hp053246@c157 assign1]$ java P4
Please enter a positive integer: 45689
The reverse of 45689 is 54698

[hp053246@c157 assign1]$ java P4
Please enter a positive integer: 456789
The reverse of 456789 is 654987
```


Email: singachea@pmail.ntu.edu.sg
Website: http://singachea.atspace.biz

Singachea