Software Development I – Exercises **Winter Term 2012/2013**

Übungen zu Softwareentwicklung 1 **Assignment 2**

Name: Andrii Dzhyrma Teaching Assistant: ________________

Student ID (Matr.Nr.): k1157448 Points (max. 24): ________________

Group: G2 Deadline: Wed, Oct. 31st, 2012 12:00 noon

Instructor: ____________________________ Editing time: ____________________

Question 1: Stock Value

a) Idea:
We should read all input information using Input.java class. For the stock price and the commission we can use float type, and for the number of shares we can use integer. Assuming that we have ideal case and there will be no input errors, after reading we should make a calculation of share value, commission and how many net proceeds. Then we will just print result on a screen.

StockValue.java:

```java
package uebung2.question1;

import io.Input;

/**
 * @author Andrii Dzhyrma
 */
public class StockValue {

	/**
	 * Request constant string for the stock price
	 */
	private static final String REQUEST_STOCK_PRICE = "Enter stock price: € ";
	/**
	 * Request constant string for the share number
	 */
	private static final String REQUEST_SHARE_NUMBER = "Enter number of shares: ";
	/**
	 * Request constant string for the commission rate
	 */
	private static final String REQUEST_COMMISSION_RATE = "Enter commission rate (as a percentage): ";

	/**
	 * Output format string for the summary
	 */
	private static final String SUMMARY_FORMAT_STRING = "%nSUMMARY%nStock price: € %.2f%nNumber of
	shares: %d%nCommission rate: %.2f%n";
	/**
	 * Output format string for the result
	 */
	private static final String RESULT_FORMAT_STRING = "%nValue of share: € %.2f%nCommission: €
	%.2f%nNet proceeds: € %.2f%n";

	/**
	 * Stock price
	 */
```

```
        private static float stockPrice;
        /**
         * Commission rate
         */
        private static float commissionRate;
        /**
         * Value of share
         */
        private static float shareValue;
        /**
         * Commission
         */
        private static float commission;
        /**
         * Net proceeds
         */
        private static float netProceeds;
        /**
         * Number of shares
         */
        private static int shareNumber;

        /**
         * @param args
         *            - no arguments will evaluate
         */
        public static void main(String[] args) {
                // Reading all the necessary information
                System.out.print(REQUEST_STOCK_PRICE);
                stockPrice = Input.readFloat();
                System.out.print(REQUEST_SHARE_NUMBER);
                shareNumber = Input.readInt();
                System.out.print(REQUEST_COMMISSION_RATE);
                commissionRate = Input.readFloat();

                // Printing summary information
                System.out.printf(SUMMARY_FORMAT_STRING, stockPrice, shareNumber, commissionRate);

                // Calculating all the result values
                shareValue = stockPrice * shareNumber;
                commission = shareValue * commissionRate / 100;
                netProceeds = shareValue - commission;

                // Printing the result
                System.out.printf(RESULT_FORMAT_STRING, shareValue, commission, netProceeds);
        }
}
```

Test plan for StackValue.java and program output (no maximum or minimum types are in the list, because Input.java will handle them):

Aim	Input	Expected output	Program output
Stock price value is < 0.	-10.1	Not a valid stock price, please re-enter. Enter stock price: €	Enter number of shares:
Stock price value is 0.	0	Enter number of shares:	Enter number of shares:
Stock price value is a string value.	qwe	Not a valid float, please try again:	Not a valid float, please try again:
Stock price value is > 0. Number of shares is < 0.	10.125 -10	Not a valid entry for number of shares, please re-enter. Enter number of shares:	Enter commission rate (as a percentage):
Stock price value is > 0. Number of shares is 0.	10.125 0	Enter commission rate (as a percentage):	Enter commission rate (as a percentage):

Stock price value is > 0. Number of shares is float.	10.125 .5	Not a valid int, please try again:	Not a valid int, please try again:
Stock price value is > 0. Number of shares is string.	10.125 qwe	Not a valid int, please try again:	Not a valid int, please try again:
Stock price value is > 0. Number of shares is >0. Commission rate is <0.	10.125 11 -10	Not a valid commission rate, please re-enter. Enter commission rate (as a percentage):	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: -10.00 Value of share: € 111.38 Commission: € -11.14 Net proceeds: € 122.51
Stock price value is > 0. Number of shares is >0. Commission rate is >100.	10.125 11 110	Not a valid commission rate, please re-enter. Enter commission rate (as a percentage):	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 110.00 Value of share: € 111.38 Commission: € 122.51 Net proceeds: € -11.14
Stock price value is > 0. Number of shares is >0. Commission rate is float.	10.125 11 .5	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 0.50 Value of share: € 111.38 Commission: € 0.56 Net proceeds: € 110.82	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 0.50 Value of share: € 111.38 Commission: € 0.56 Net proceeds: € 110.82
Stock price value is > 0. Number of shares is >0. Commission rate is string.	10.125 11 qwe	Not a valid float, please try again:	Not a valid float, please try again:
Stock price value is > 0. Number of shares is >0. Commission rate >=0 and <=100. (Correctness)	10.125 11 1.5	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 1.50 Value of share: € 111.38 Commission: € 1.67 Net proceeds: € 109.70	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 1.50 Value of share: € 111.38 Commission: € 1.67 Net proceeds: € 109.70

b) Idea:

We should read all input information using Input.java class. For the stock price and the commission we can use float type, and for the number of shares we can use integer. Input.java will handle all errors with types, so we should handle only errors with boundaries. We will ask to input stock price while it will not be more or equal to 0. The same situation is for number of shares. For commission rate we should also check that value is less or equal to 100. After reading we should make a calculation of share value, commission and how many net proceeds. Then we will just print result on a screen.

StockValueModified.java:

```java
package uebung2.question1;

import io.Input;

/**
 * @author Andrii Dzhyrma
 */
public class StockValueModified {

	/**
	 * Request constant string for the stock price
	 */
	private static final String REQUEST_STOCK_PRICE = "Enter stock price: € ";
	/**
	 * Request constant string for the share number
	 */
	private static final String REQUEST_SHARE_NUMBER = "Enter number of shares: ";
	/**
	 * Request constant string for the commission rate
	 */
	private static final String REQUEST_COMMISSION_RATE = "Enter commission rate (as a percentage): ";

	/**
	 * Request constant error string for the stock price
	 */
	private static final String REQUEST_STOCK_PRICE_ERROR = "Not a valid stock price, please re-
	enter.";
	/**
	 * Request constant error string for the number of shares
	 */
	private static final String REQUEST_SHARE_NUMBER_ERROR = "Not a valid entry for number of shares,
	please re-enter.";
	/**
	 * Request constant error string for the commission rate
	 */
	private static final String REQUEST_COMMISSION_RATE_ERROR = "Not a valid commission rate, please
	re-enter.";

	/**
	 * Output format string for the summary
	 */
	private static final String SUMMARY_FORMAT_STRING = "%nSUMMARY%nStock price: € %.2f%nNumber of
	shares: %d%nCommission rate: %.2f%n";
	/**
	 * Output format string for the result
	 */
	private static final String RESULT_FORMAT_STRING = "%nValue of share: € %.2f%nCommission: €
	%.2f%nNet proceeds: € %.2f%n";

	/**
	 * Stock price
	 */
	private static float stockPrice;
	/**
	 * Commission rate
	 */
	private static float commissionRate;
	/**
	 * Value of share
	 */
	private static float shareValue;
	/**
	 * Commission
	 */
	private static float commission;
	/**
```

```
	 * Net proceeds
	 */
	private static float netProceeds;
	/**
	 * Number of shares
	 */
	private static int shareNumber;

	/**
	 * @param args
	 *            - no arguments will evaluate
	 */
	public static void main(String[] args) {
		// Reading all the necessary information
		System.out.print(REQUEST_STOCK_PRICE);
		stockPrice = Input.readFloat();
		while (stockPrice < 0) {
			System.out.println(REQUEST_STOCK_PRICE_ERROR);
			System.out.print(REQUEST_STOCK_PRICE);
			stockPrice = Input.readFloat();
		}
		System.out.print(REQUEST_SHARE_NUMBER);
		shareNumber = Input.readInt();
		while (shareNumber < 0) {
			System.out.println(REQUEST_SHARE_NUMBER_ERROR);
			System.out.print(REQUEST_SHARE_NUMBER);
			shareNumber = Input.readInt();
		}
		System.out.print(REQUEST_COMMISSION_RATE);
		commissionRate = Input.readFloat();
		while (commissionRate < 0 || commissionRate > 100) {
			System.out.println(REQUEST_COMMISSION_RATE_ERROR);
			System.out.print(REQUEST_COMMISSION_RATE);
			commissionRate = Input.readFloat();
		}

		// Printing summary information
		System.out.printf(SUMMARY_FORMAT_STRING, stockPrice, shareNumber, commissionRate);

		// Calculating all the result values
		shareValue = stockPrice * shareNumber;
		commission = shareValue * commissionRate / 100;
		netProceeds = shareValue - commission;

		// Printing the result
		System.out.printf(RESULT_FORMAT_STRING, shareValue, commission, netProceeds);
	}

}
```

Test plan for StackValueModified.java and program output:

Aim	Input	Expected output	Program output
Stock price value is < 0.	-10.1	Not a valid stock price, please re-enter. Enter stock price: €	Not a valid stock price, please re-enter. Enter stock price: €
Stock price value is 0.	0	Enter number of shares:	Enter number of shares:
Stock price value is a string value.	qwe	Not a valid float, please try again:	Not a valid float, please try again:
Stock price value is > 0. Number of shares is < 0.	10.125 -10	Not a valid entry for number of shares, please re-enter. Enter number of shares:	Not a valid entry for number of shares, please re-enter. Enter number of shares:
Stock price value is > 0. Number of	10.125	Enter commission rate (as	Enter commission rate (as

shares is 0.	0	a percentage):	a percentage):
Stock price value is > 0. Number of shares is float.	10.125 .5	Not a valid int, please try again:	Not a valid int, please try again:
Stock price value is > 0. Number of shares is string.	10.125 qwe	Not a valid int, please try again:	Not a valid int, please try again:
Stock price value is > 0. Number of shares is >0. Commission rate is <0.	10.125 11 -10	Not a valid commission rate, please re-enter. Enter commission rate (as a percentage):	Not a valid commission rate, please re-enter. Enter commission rate (as a percentage):
Stock price value is > 0. Number of shares is >0. Commission rate is >100.	10.125 11 110	Not a valid commission rate, please re-enter. Enter commission rate (as a percentage):	Not a valid commission rate, please re-enter. Enter commission rate (as a percentage):
Stock price value is > 0. Number of shares is >0. Commission rate is float.	10.125 11 .5	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 0.50 Value of share: € 111.38 Commission: € 0.56 Net proceeds: € 110.82	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 0.50 Value of share: € 111.38 Commission: € 0.56 Net proceeds: € 110.82
Stock price value is > 0. Number of shares is >0. Commission rate is string.	10.125 11 qwe	Not a valid float, please try again:	Not a valid float, please try again:
Stock price value is > 0. Number of shares is >0. Commission rate >=0 and <=100. (Correctness)	10.125 11 1.5	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 1.50 Value of share: € 111.38 Commission: € 1.67 Net proceeds: € 109.70	SUMMARY Stock price: € 10.13 Number of shares: 11 Commission rate: 1.50 Value of share: € 111.38 Commission: € 1.67 Net proceeds: € 109.70

Question 2: Vending Machine

c) Idea:

At first we should store in static final fields or important values that we will use in a code. This is first letter from the alphabet in upper case that represents first item in the vending machine, amount cents in 1 euro, array of items (names), array of their prices, sorted array from biggest coin to lowest and other less important constant for calculation or strings for memory optimization. On the initialization part of the program we should check, if length of two arrays with names and prices of an items are the same, if not we should ask to change somehow initialization values. To propose user pick some letter, we should combine queue of alphabetical symbols depends on the amount of items in array. After that we can read input choice and check if it in a range or not, including possibility to input lower case characters. Then we should show to consumer amount of money to insert. After that we will ask user to input any amount of money till the sum will reach the price or will be more than the price. As was mentioned in requirements “Do not worry about the incoming coins”, so we will not check is the input amount is greater or equal to zero or coins can be of float type. Then we will calculate change and amount of coins that we should produce. At the end we will write to user how many coins he will get and what item he will receive. The code for an items, prices and coins should be organized in way that if we will add some new items or coins in constant arrays, no more improvements in code should not be provided.

VendingMachine.java:

```java
package uebung2.question2;

import io.Input;

/**
 * @author Andrii Dzhyrma
 */
public class VendingMachine {

	/**
	 * Constant string that represents separating symbol between options
	 */
	private static final String SEPARATOR = ", ";
	/**
	 * Constant empty string
	 */
	private static final String EMPTY_STRING = "";
	/**
	 * Constant char for the first item symbol in UPPERCASE
	 */
	private static final char A = 'A';
	/**
	 * Constant difference between upper case and lower case
	 */
	private static final int CASE_DIFFERENCE = 'A' - 'a';
	/**
	 * Constant amount of cents in 1 €
	 */
	private static final int CENTS_IN_EURO = 100;
	/**
	 * Constant float value that defines epsilon for float comparison
	 */
	private static final float EPSILON = 0.005f;

	/**
	 * Constant array with the names of items in the vending machine
	 */
	private static final String[] ITEMS = { "MARS", "TWIX", "M & M's", "CHIPS", "MINTS" };
	/**
	 * Constant array with the prices of items in the vending machine
	 */
	private static final float[] PRICES = { .9f, .9f, .9f, 1.2f, .6f };
	/**
	 * Constant array with the equity of coins in the vending machine
	 */
	private static final int[] COINS = { 200, 100, 50, 20, 10 };

	/**
	 * Initialization error string
	 */
	private static final String INITIALIZATION_ERROR = "Amount of the items in vending machine is not
	the same as amount of their prices. Call administrator for a help!";
	/**
	 * Request constant format string for selecting an item
	 */
	private static final String REQUEST_SELECT_ITEM_FORMAT_STRING = "Please select an item (%s):%n";
```

```
/**
 * Request constant format string for the amount of money left, if first
 * input money was not enough
 */
private static final String REQUEST_LEFT_AMOUNT_FORMAT_STRING = "Input:%.2f%nEnter another:
%.2f%n";
/**
 * Output format string for the amount of money left
 */
private static final String REQUEST_AMOUNT_FORMAT_STRING = "Amount due: %.2f€%n";
/**
 * Output format string for the result (first part)
 */
private static final String RESULT1_FORMAT_STRING = "Total Change due: %.2f€%nChange: ";
/**
 * Output format string for the result (last part)
 */
private static final String RESULT2_FORMAT_STRING = "%nOutput: %s%n";
/**
 * Output format string for the amount of coins in change
 */
private static final String CHANGE_FORMAT_STRING = "%d ";
/**
 * Output format string for the € value
 */
private static final String EURO_FORMAT_STRING = "%d€";
/**
 * Output format string for the cents value
 */
private static final String CENT_FORMAT_STRING = "%dc";

/**
 * @param args
 *            - no arguments will evaluate
 */
public static void main(String[] args) {
        // Initialization checking
        if (ITEMS.length != PRICES.length) {
                System.out.println(INITIALIZATION_ERROR);
                return;
        }

        // Making string for possible options
        String options = EMPTY_STRING;
        for (char symbol = A; symbol - A < ITEMS.length; symbol++)
                options += ((symbol == A) ? EMPTY_STRING : SEPARATOR) + symbol;

        // Reading input char for an item while it will be in the range from 'a'
        // to 'e' or from 'A' to 'E'
        char itemChar;
        do {
                System.out.printf(REQUEST_SELECT_ITEM_FORMAT_STRING, options);
                itemChar = Input.readChar();
        } while (((itemChar - A < 0) || (itemChar - A > ITEMS.length))
                        && ((itemChar - A + CASE_DIFFERENCE < 0) || (itemChar - A +
                        CASE_DIFFERENCE > ITEMS.length)));
        // Getting normal integer index for the item
        int itemIndex = ((itemChar - A < 0) || (itemChar - A > ITEMS.length)) ? itemChar - A +
                        CASE_DIFFERENCE : itemChar - A;
        // Printing price for that item
        float moneyLeft = PRICES[itemIndex];
        System.out.printf(REQUEST_AMOUNT_FORMAT_STRING, moneyLeft);

        // Reading input amount of money. No checking for input with minus
        // because of the requirements
        float input = Input.readFloat();
        moneyLeft -= input;
        // Continue asking for more money, until it will be enough
        while (moneyLeft > EPSILON) {
                System.out.printf(REQUEST_LEFT_AMOUNT_FORMAT_STRING, input, moneyLeft);
                input = Input.readFloat();
                moneyLeft -= input;
            }

            // Printing the result and calculating the change
            System.out.printf(RESULT1_FORMAT_STRING, Math.abs(moneyLeft));
            int change = Math.abs(Math.round(moneyLeft * 100));
            int[] changeCoinsCounter = new int[COINS.length];
            for (int i = 0; i < COINS.length; i++) {
                changeCoinsCounter[i] = change / COINS[i];
                change %= COINS[i];
                System.out.printf(CHANGE_FORMAT_STRING, changeCoinsCounter[i]);
                if (COINS[i] >= CENTS_IN_EURO)
                    System.out.printf(EURO_FORMAT_STRING, COINS[i] / CENTS_IN_EURO);
                if (COINS[i] % CENTS_IN_EURO > 0)
                    System.out.printf(CENT_FORMAT_STRING, COINS[i] % CENTS_IN_EURO);
                if (i < COINS.length - 1)
                    System.out.print(SEPARATOR);
            }
            System.out.printf(RESULT2_FORMAT_STRING, ITEMS[itemIndex]);
        }
}
```

Test plan for VendingMachine.java and program output:

Aim	Input	Expected output	Program output
Item is not in the list.	Z	Please select an item (A, B, C, D, E):	Please select an item (A, B, C, D, E):
Item is not a char.	123	Not a valid char, please try again:	Not a valid char, please try again:
Item is lower case character.	a	Amount due: 0.90€	Amount due: 0.90€
Item is a char in the list. Input amount is not enough.	E .5	Input:0.50 Enter another: 0.10	Input:0.50 Enter another: 0.10
Item is a char in the list. Input amount is < 0.	E -5	Input:-5.00 Enter another: 5.60	Input:-5.00 Enter another: 5.60
Item is a char in the list. Input amount is 0.	A 0	Input:0.00 Enter another: 0.90	Input:0.00 Enter another: 0.90
Item is a char in the list. Input amount is equal to price.	A .9	Total Change due: 0.00€ Change: 0 2€, 0 1€, 0 50c, 0 20c, 0 10c Output: MARS	Total Change due: 0.00€ Change: 0 2€, 0 1€, 0 50c, 0 20c, 0 10c Output: MARS
Item is a char in the list. Input amount is more than price.	A 2	Total Change due: 1.10€ Change: 0 2€, 1 1€, 0 50c, 0 20c, 1 10c Output: MARS	Total Change due: 1.10€ Change: 0 2€, 1 1€, 0 50c, 0 20c, 1 10c Output: MARS
Item is a char in the list. Input amount is more than price by 5 euro and 85 cents.	A 6.75	Total Change due: 5.85€ Change: 2 2€, 1 1€, 1 50c, 1 20c, 1 10c Output: MARS	Total Change due: 5.85€ Change: 2 2€, 1 1€, 1 50c, 1 20c, 1 10c Output: MARS
Item is a char in the list. Input amount is more than price by 5 euro and 95 cents.	A 6.85	Total Change due: 5.95€ Change: 2 2€, 1 1€, 1 50c, 2 20c, 0 10c Output: MARS	Total Change due: 5.95€ Change: 2 2€, 1 1€, 1 50c, 2 20c, 0 10c Output: MARS