Linnaeus University

1DV532 – Starting out with Java
Assignment 4 Report

Student: Evan Huynh
Student ID: eh223im@student.lnu.se

Table of Contents

Report

Exercise 1: BankAccount

The exercise requires verifying user inputs and give them another chance to enter again, until an expected input is received. It is pretty easy to solve this problem, by using a while loop or do-while loop, we can force the problem to run until the expected output is received.

Since this problem can be solved with only one method, it will be solved in one method. Of course, this is bad practice but since time is constricted, this is the best solution.

The first check with customer ID, we require users to input a string starting with Latin characters and ends with 3 digits. We can solve this using either if loop or a regular expression, in this case, starting with any number of English characters so [A-z]+ and ends with 3 digits so it will be [0-9]{3}.

The second check will be account number, with 7 digits. This time we will requires users to input a 7-digit integer, meaning that leading zeros are not acceptable (to simplify the solution). This one also has a try…catch block in case users enters a non-integer input or enters a number with less than 7 digits (not including those with leading zeros). In those cases, it will simply print out the error and force the users to enter the information again.

The third check is balance, which is an integer larger than 1000. Similar to account number check, this one also checks if the input is larger than 1000 and is an integer in try...catch and if block inside and works similarly as the account number check.

At the end of this program are some methods and fields needed for exercise 2.

Exercise 2: Stack

Since I have done this one before, this will be just another attempt but with changing in of the Object. Since exercise 1 can be finished with one method, the blanket constructor and getStr method has been specifically created for exercise 2 (to simplify the testing and nothing else).

In the stack, at first, we initialize an array with length 5 (because it is half of 10) and size of the stack is 0 at first. If a push is requested, it will check if the array is full, double the array size if that is the case, move everything of the array to the next position and replace the first element with the one needed pushing in.

For the pop method, it will throw an Exception if the stack is empty, otherwise it will return the first element in the array, copy everything from the second element to the previous position and decrease the size.

For the peek method, it will just return the first element in the stack without removing them or throw an exception if it is empty.

For the StackMain class, we have empty test, add 1 then empty the stack test, add 100 then pop them all test with all methods available from the StackImpl class.

Exercise 3: Polygon

In this exercise it requires some default methods such as getPerimeter, returns the value of getSideLength times getNumSides, getInteriorAngle returns (getNumSides-2)*180, getTotalSides returns the total number of sides from a given RegularPolygon array.

Since most of the necessary methods are coded in the interface, in the child classes such as EquilateralTriangle and Square, we only need one constructor of side length, set the number of sides to 3 for triangle, 4 for square and one getter method. Of course, this will also check that if the side length is larger than 0 then throw an exception, because we cannot have negative side length, otherwise proceed as normal.

The RegularPolygonMain is just a test with all methods from RegularPolygon.

Exercise 4: Average

This method requires users to enter the number of numbers, enter them in order, store each of the valid number in the array and then calculate the average.

For the first check, how many numbers, we just ask users to enter a number to calculate. By default, we set the default to 0. If users enter anything invalid, the line "Number must be greater than 0" will always be written (because of the default value).

For the next step, the program will ask users to enter as many numbers as stated. If users enter something else than integers, the program will roll back one step (decreasing i by 1, because it is Java, not Python). This method works because at the end of the for loop, Java increases i by 1, thus we have done nothing to the index if the users try to break it. This is also more convenient since we do not have to require users to enter all the input starting from the beginning, just starts where they left off.

At the end of the program, we simply calculate the average and format the output get print the result.

Source code

Here are my source codes for all the exercise.

1

BankAccount.java

```java
package eh223im_assign4;

import java.util.InputMismatchException;
import java.util.Scanner;

public class BankAccount {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Welcome to the Bank of Students, By the
Students, for the Students!");
        System.out.print("Enter customer ID: ");
        String a;
        a = s.next();
        while (!a.matches("^[A-z]+[0-9]{3}")) { // Regex because it
sounds fun, any chars in the beginning, 3 digits at the end.
            System.out.println("Incorrect Customer ID. The Customer ID
must start with a letter followed by three digits.\nInput again");
            System.out.print("Enter customer ID: ");
            a = s.next();
        }
        int b = 0;
        do {
            try {
                System.out.print("Enter account no: ");
                b = s.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("Incorrect Account No. The Account
number must be of seven digits.\nInput again.");
                s.nextLine();
            }
            if (b<=999999 || b > 9999999) {
                System.out.println("Incorrect Account No. The Account
number must be of seven digits.\nInput again.");
            }
        } while (b<=999999 || b > 9999999);

        int c = 0;
        do {
            try {
                System.out.print("Enter balance: ");
                c = s.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("Initial balance must be equal to or
higher than 1000.\nInput again.");
                s.nextLine();
            }
            if (c < 1000) {
                System.out.println("Initial balance must be equal to or
higher than 1000.\nInput again.");
            }
        } while (c < 1000);
        System.out.println("Congratulations, your account has been create
successfully!");
```

```java
        System.out.println("ID: "+a+" | Account no: "+b+" | Balance: 
"+c);
    }

    // For exercise 2 only
    private String str;
    public BankAccount(String str) {
        this.str = str;
    }

    public String getStr() {
        return str;
    }
}
```

2

Stack.java

```java
package eh223im_assign4.stack;

import eh223im_assign4.BankAccount;

public interface Stack {
    int size(); // returns current number of elements (accounts) in the 
stack
    boolean isEmpty(); // true if stack is empty
    void push(BankAccount account); // Adds an account at the top of the 
stack.
    BankAccount pop(); // returns and removes an account at top, throws 
an exception if stack is empty
    BankAccount peek(); // returns (without removing) top element, throws 
an exception if stack is empty.
}
```

StackImpl.java

```java
package eh223im_assign4.stack;

import eh223im_assign4.BankAccount;

public class StackImpl implements Stack {
    // Since I have done this before, as always
    // Here is the proper citation for the code, just in case.
    /**
     * Author(s) name: My2ndAngelic
     * Date: 2017 Dec
     * Title of program/source code: StackUtils
     * Code version: N/A
     * Type: computer program
     * Web address: 
https://github.com/My2ndAngelic/1DV506/blob/master/src/main/java/eh223im_
assign4/stack/StackUtils.java
     */

    BankAccount[] bAss = new BankAccount[5];
    int size = 0;

    @Override
    public int size() {
        return size;
    }

    @Override
    public boolean isEmpty() {
        return size == 0;
    }

    @Override
    public void push(BankAccount bankAccount) {
        if (size == bAss.length) {
            BankAccount[] temp = new BankAccount[size * 2];
            System.arraycopy(bAss, 0, temp, 0, size);
            bAss = temp;
        }
        System.arraycopy(bAss,0,bAss,1,size);
        bAss[0] = bankAccount;
        size++;
    }

    @Override
    public BankAccount pop() {
        if (size == 0) {
            throw new IndexOutOfBoundsException("Empty stack.");
        } else {
            BankAccount temp = bAss[0];
            System.arraycopy(bAss, 1, bAss, 0, size - 1);
            size--;
            bAss[size] = null;
            return temp;
        }
    }

    @Override
    public BankAccount peek() {
        if (size == 0) {
            throw new IndexOutOfBoundsException("Empty stack.");
        } else {
            return bAss[0];
        }
    }
}
```

StackMain.java

```
package eh223im_assign4.stack;

import eh223im_assign4.BankAccount;

public class StackMain {
    public static void main(String[] args) {
        Stack s = new StackImpl();
        // Empty stack
        try {
            s.pop();
        } catch (Exception e) {
            e.getStackTrace();
            System.out.println("Empty stack here.");
        }

        // Add 1 then empty
        s.push(new BankAccount("str"));
        System.out.println("Peek: "+s.peek().getStr());
        System.out.println("Pop: "+s.pop().getStr());
        System.out.println("Is empty: "+s.isEmpty());
```

```java
        // Add 100 then pop all
        for (int i = 0; i < 100; i++) {
            s.push( new BankAccount("str"+(i)));
        }
        System.out.println("Size after pushing: "+s.size());
        for (int i = 0; i < 100; i++) {
            System.out.println(s.pop().getStr());
        }
        System.out.println("Size after popping: "+s.size());
    }
}
```

3

RegularPolygon.java

```java
package eh223im_assign4.polygons;

public interface RegularPolygon {
    public abstract int getNumSides();
    public abstract int getSideLength();
    public default int getPerimeter() {
        return getSideLength()*getNumSides();
    }
    public default int getInteriorAngle () {
        return (getNumSides()-2)*180;
    }

    public static int totalSides(RegularPolygon[] pol) {
        int k = 0;
        for (RegularPolygon regularPolygon : pol) {
            k += regularPolygon.getNumSides();
        }
        return k;
    }
}
```

EquilateralTriangle.java

```java
package eh223im_assign4.polygons;

import java.util.InputMismatchException;

public class EquilateralTriangle implements RegularPolygon {
    private int sideLength;

    public EquilateralTriangle(int sideLength){
        if (sideLength > 0) {
            this.sideLength = sideLength;
        } else throw new InputMismatchException("Invalid side length.");
    }

    @Override
    public int getNumSides() {
        return 3;
    }

    @Override
    public int getSideLength() {
        return sideLength;
    }
}
```

Square.java

```java
package eh223im_assign4.polygons;

import java.util.InputMismatchException;

public class Square implements RegularPolygon {
    private int sideLength;

    public Square(int sideLength){
        if (sideLength > 0) {
            this.sideLength = sideLength;
        } else throw new InputMismatchException("Invalid side length.");
    }

    @Override
    public int getNumSides() {
        return 4;
    }

    @Override
    public int getSideLength() {
        return sideLength;
    }
}
```

RegularPolygonMain.java

```java
package eh223im_assign4.polygons;

public class RegularPolygonMain {
    public static void main(String[] args) throws Exception {
        RegularPolygon a = new EquilateralTriangle(5);
        System.out.println("Triangle");
        System.out.println("Number of sides: "+a.getNumSides());
        System.out.println("Side length: "+a.getSideLength());
        System.out.println("Interior angele: "+a.getInteriorAngle());
        System.out.println("Perimeter: "+a.getPerimeter());
        RegularPolygon b = new Square (5);
        System.out.println();
        System.out.println("Square");
        System.out.println("Number of sides: "+b.getNumSides());
        System.out.println("Side length: "+b.getSideLength());
        System.out.println("Interior angele: "+b.getInteriorAngle());
        System.out.println("Perimeter: "+b.getPerimeter());
        RegularPolygon[] rp = new RegularPolygon[10];

        for (int i = 0; i < rp.length; i++) {
            rp[i] = new Square(5);
        }
        System.out.println();
        System.out.println("Total number of sides of 10 squares: 
"+RegularPolygon.totalSides(rp));
    }
}
```

4

Average.java

```java
package eh223im_assign4;

import java.text.DecimalFormat;
import java.util.Arrays;
import java.util.InputMismatchException;
import java.util.Scanner;

public class Average {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int a = 0;
        while (a <= 0) {
            System.out.println("How many numbers do you want to enter?");
            try {
                a = s.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("You entered a non-numeric value which
is not allowed. Please enter the number again.");
                s.nextLine();
            }
            if (a <= 0) {
                System.out.println("Number must be greater than 0.");
            }
        }

        int[] b = new int[a];

        for (int i = 0; i < a; i++) {
            try {
                System.out.print("Enter number " + (i+1) +": ");
                b[i] = s.nextInt();
            } catch (Exception e) {
                System.out.println("Enter a valid number.");
                s.nextLine();
                i--;
            }
        }
        System.out.println(Arrays.toString(b));

        double c = 0;
        for (int value : b) {
            c += value;
        }
        c /= b.length;
        String format = new DecimalFormat("#.##").format(c);

        System.out.println("Average of the entered numbers is "+ format);
    }
}
```