Data Structure and Algorithm Practicum
Stack

Name
Dicha Zelianivan Arkana

NIM
2241720002

Class
1i

Department
Information Technology

Study Program
D4 Informatics Engineering

1 Lab Activity

1.1 Questions

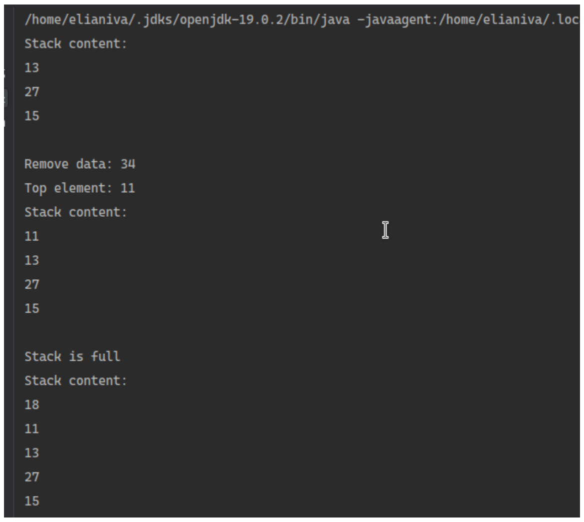
1. In class `StackMain`, what is the usage of number 5 in this following code?

   ```
   Stack stack = new Stack(5);
   ```

   It's used to define the size of the stack. There can only be 5 items in the stack.

2. Add 2 more data in the stack with 18 and 40. Display the result!

   ```
   public static void main(String[] args) {
       Stack stack = new Stack(5);
       stack.push(15);
       stack.push(27);
       stack.push(13);
       stack.print();
       stack.push(11);
       stack.push(34);
       stack.pop();
       stack.peek();
       stack.print();
       stack.push(18);
       stack.push(40);
       stack.print();
   }
   ```



```
/home/elianiva/.jdks/openjdk-19.0.2/bin/java -javaagent:/home/elianiva/.loc
Stack content:
13
27
15

Remove data: 34
Top element: 11
Stack content:
11
13
27
15

Stack is full
Stack content:
18
11
13
27
15
```

Figure 1: The output after adding 18 and 40

3. In previous number, the data inserted in to the stack is only 18, and 40 is not inserted. Why is that?

   Because the maximum capacity is 5 and 40 is the 6th item which means that it can no longer contain the new item.

2 Lab Activity

2.1 Questions

1. In class `StackMain`, when calling `push()` method, the argument is `bk`. What information is included in the `bk` variable?

   The information included is the Book class that was constructed before passing it to the push method

2. Which of the program that its usage is to define the capacity of the stack?

   It's in the constructor of the Stack, the constructor receives an integer that defines the size of the stack.

3. What is the function of do-while that exists in `StackMain` class?

   It is used to repeat the process of inserting the book into the stack until the user no longer wants to continue.

4. Modify the program in `StackMain`, so that the user may choose which operation (push, pop, peek, print) to do in stack from program menu!

3 Lab Activity

3.1 Questions

1. Please explain the flow of method in `Postfix` class!

   - `push` - add an item to the top of the stack, and increment the top counter
   - `pop` - removes the top-most item from the stack, and decrement the top counter
   - `degree` - gets the order of precedence for each operator using switch case
   - `isOperand` - checks if its an operand or not by checking if the character is within the range of A-Z, a-z, and 0-9
   - `isOperator` - checks if its an operator or not by checking if the character is a mathematical operator

- `convert`

  It walks through every character of the input, every operand is inserted into the postfix string. Everytime it encounters an operator, it pushes it to the stack. However, if there is an operator with higher precedence in the stack, it will pop that operator first and then put the curren toperator into the stack. If it encounters a closing brackets, it will pop every items inside the stack until it found the opening brace.

2. What is the function of this program code?

   ```
   c = Q.charAt(i);
   ```

   It is used to get the character at position i

3. Execute the program again, how's the result if we insert **3*5^(8-6)%3** for the expression?

   The result is **3586-^3%***

4. In 2nd number, why the braces are not displayed in conversion result? Please explain

   Because the braces are not necessary since the order of the operand and the operator is already correct so there is no need to include them in the postfix notation.

4 Assignments

1. Create a program with Stack implementation to insert a sentence and display the reversed version of the sentence as a result!

   CharStack.java

   ```
   public class CharStack {
       int top = -1;
       int size;
       char[] data;

       public CharStack(int size) {
           data = new char[size];
           this.size = size;
       }

       boolean isFull() {
           return top == size - 1;
       }
   ```

```
    boolean isEmpty() {
        return top == -1;
    }

    void push(char c) {
        if (isFull()) {
            System.out.println("Stack is already full");
            return;
        }

        top++;
        data[top] = c;
    }

    char pop() {
        if (isEmpty()) {
            System.out.println("Stack is already empty");
            return '.';
        }

        char item = data[top];
        top--;
        return item;
    }
}
```

Assignment1.java

```
import java.util.Scanner;

public class Assignment1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Insert the sentence: ");
        String sentence = sc.nextLine();

        CharStack charStack = new CharStack(sentence.length());
        for (int i = 0; i < sentence.length(); i++) {
            charStack.push(sentence.charAt(i));
        }

        System.out.print("Reversed sentence: ");
        while (!charStack.isEmpty()) {
            System.out.write(charStack.pop());
        }
        System.out.flush();
    }
}
```

2. Every Sunday, Dewi shops to a supermarket that is in her residential area. Everytime she finishes, she keeps the receipt of what she has bought in a wardrobe. After 2 months, She had 8 receipts. She plans to trade her 5 receipts in exchange for a voucher. Create a program using stack implementation to store Dewi's receipt. As well as the retrieving of the receipts. The information that are included in a receipt are as follow:

   - Transaction ID
   - Date
   - Quantity of items
   - Total price

   Code

   - `Receipt.java`

```
public class Receipt {
    String transactionId;
    String date;
    int quantity;
    int totalPrice;

    public Receipt(String transactionId, String date, int quantity, int totalPrice
        this.transactionId = transactionId;
        this.date = date;
        this.quantity = quantity;
        this.totalPrice = totalPrice;
    }
}
```

   - `ReceiptStack.java`

```
public class ReceiptStack {
    int size;
    int top;
    Receipt[] data;

    public ReceiptStack(int size) {
        this.size = size;
        data = new Receipt[size];
        top = -1;
```

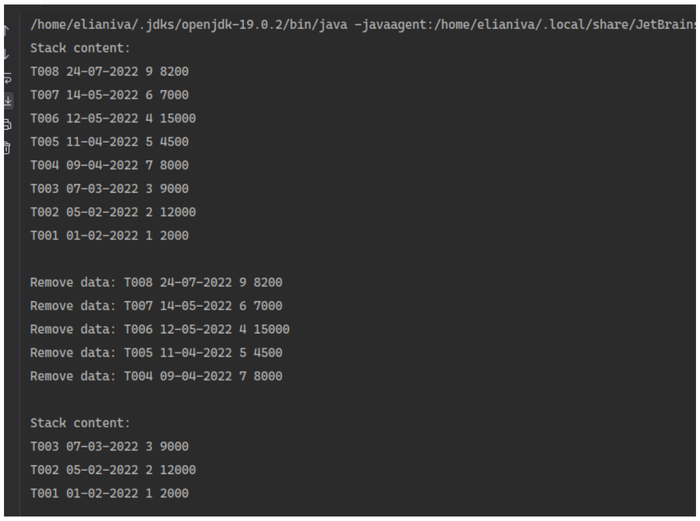
```
}

public boolean isEmpty() {
    return top == -1;
}

public boolean isFull() {
    return top == (size - 1);
}

public void push(Receipt book) {
    if (isFull()) {
        System.out.println("Stack is full");
        return;
    }

    top++;
    data[top] = book;
}

public void pop() {
    if (isEmpty()) {
        System.out.println("Stack is empty");
        return;
    }

    Receipt x = data[top];
    top--;
    System.out.println("Remove data: " + x.transactionId + " " + x.date + " "
}

public void peek() {
    System.out.println("Top element: " + data[top]);
}

public void print() {
    System.out.println("Stack content: ");
    for (int i = top; i > -1; i--) {
        System.out.println(data[i].transactionId + " " + data[i].date + " " +
    }
    System.out.println();
}

public void clear() {
    if (isEmpty()) {
```

```
                System.out.println("Failed! Stack is empty");
                return;
            }

            top = -1;
            System.out.println("Stack is now empty");
        }
    }
```

- ReceiptMain.java

```
public class ReceiptMain {
    public static void main(String[] args) {
        ReceiptStack receiptStack = new ReceiptStack(10);

        // adding 8 receipts
        receiptStack.push(new Receipt("T001", "01-02-2022", 1, 2000));
        receiptStack.push(new Receipt("T002", "05-02-2022", 2, 12000));
        receiptStack.push(new Receipt("T003", "07-03-2022", 3, 9000));
        receiptStack.push(new Receipt("T004", "09-04-2022", 7, 8000));
        receiptStack.push(new Receipt("T005", "11-04-2022", 5, 4500));
        receiptStack.push(new Receipt("T006", "12-05-2022", 4, 15000));
        receiptStack.push(new Receipt("T007", "14-05-2022", 6, 7000));
        receiptStack.push(new Receipt("T008", "24-07-2022", 9, 8200));

        // display items
        receiptStack.print();

        // pop 5 receipts
        for (int i = 0; i < 5; i++) {
            receiptStack.pop();
        }
        System.out.println();

        // display receipts
        receiptStack.print();
    }
}
```

Output



```
/home/elianiva/.jdks/openjdk-19.0.2/bin/java -javaagent:/home/elianiva/.local/share/JetBrains
Stack content:
T008 24-07-2022 9 8200
T007 14-05-2022 6 7000
T006 12-05-2022 4 15000
T005 11-04-2022 5 4500
T004 09-04-2022 7 8000
T003 07-03-2022 3 9000
T002 05-02-2022 2 12000
T001 01-02-2022 1 2000

Remove data: T008 24-07-2022 9 8200
Remove data: T007 14-05-2022 6 7000
Remove data: T006 12-05-2022 4 15000
Remove data: T005 11-04-2022 5 4500
Remove data: T004 09-04-2022 7 8000

Stack content:
T003 07-03-2022 3 9000
T002 05-02-2022 2 12000
T001 01-02-2022 1 2000
```