# Data Structure and Algorithm Practicum
# Stack


**Name**
Muhammad Baihaqi Aulia Asy'ari

**NIM**
2241720145

**Class**
1I

**Department**
Information Technology

**Study Program**
D4 Informatics Engineering

## 1.1 Learning Objective

After finishing this practicum session, students will be able to:

- Define the Stack Data Structure
- Create and implement Stack Data Structure
- Implement Stack data Structure with arrays

## 1.2 Lab Activities

In this practicum, we will implement **Stack** class

### 1.2.1 Steps

1. Take a look at this following class diagram for **Stack** class:

| **Stack** |
| --- |
| size: int<br>top: int<br>data[]: int |
| Stack(size: int)<br>IsEmpty(): boolean<br>IsFull(): boolean<br>push(): void<br>pop(): void<br>peek(): void<br>print(): void<br>clear(): void |

   Based on class diagram above, we will create the **Stack** class in Java program.

2. Create a new project named **Jobsheet7**. Create a new package with name **Practicum1**. Then, create a new class named **Stack**.

3. Create new attributes size, top, and data as follows:

```
int size;
int top;
int data[];
```

4. Add a constructor with parameter as written below:

```
public Stack(int size) {
    this.size = size;
```

```
        data = new int[size];
        top = -1;
    }
```

5. Create a method **isEmpty** with Boolean as its return type to check whether the stack is empty or not.

```
public boolean isEmpty() {
    if (top == 1) {
        return true;
    } else {
        return false;
    }
}
```

6. Create a method **isFull** with Boolean as its return type to check whether the stack is filled completely or not.

```
public boolean isFull() {
    if (top == size - 1) {
        return true;
    } else {
        return false;
    }
}
```

7. Create method **push** with void as its return type to add new stack element with parameter **dt**. This dt variable is in form of integer

```
public void push(int dt) {
    if (!isFull()) {
        top++;
        data[top] = dt;
    } else {
        System.out.println("Stack is full");
    }
}
```

8. Create method **pop** with void as its return type to remove an element from the stack

```
public void pop() {
    if (!isEmpty()) {
        int x = data[top];
        top--;
```

```
            System.out.println("Remove data : " + x);
        } else {
            System.out.println("Stack is empty");
        }
    }
```

9. Create method **peek** with void as its return type to check the top element of the stack

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

10. Create method **print** with void as its return type to display the content of the stack

```
public void print() {
    System.out.println("Stack content: ");
    for (int i = top; i >= 0; i--) {
        System.out.println(data[i] + " ");
    }
    System.out.println("");
}
```

11. Create method **clear** with void as its data type to remove all elements and make the stack empty

```
public void clear() {
    if (!isEmpty()) {
        for (int i = top; i >= 0; i--) {
            top--;
        }
        System.out.println("Stack is now empty");
    } else {
        System.out.println("Failed ! Stack is still empty");
    }
}
```

12. Next up, we create a new class named **StackMain** inside the package **Practicum1**. Create a main function and make object instantiation with name is **stk**

```
public class StackMain {
    public static void main(String[] args) {
        Stack stk = new Stack(5);
```

```
    }
}
```

13. Fill the stack object by calling method **push**, the data is being inserted accordingly

```
stk.push(15);
stk.push(27);
stk.push(13);
```

14. Display the data that we've inserted in previous step by calling method **print**

```
stk.print();
```

15. Repeat the insertion process twice, then call pop **method** to remove an element. We can also check the top data with **peek** method. Finally, display all the data by calling method **print**

```
stk.push(11);
stk.push(34);
stk.pop();
stk.peek();
stk.print();
```

16. Compile and run the program, check the result

```
package Practicum1;

public class Stack {
    int size;
    int top;
    int data[];

    public Stack(int size) {
        this.size = size;
        data = new int[size];
        top = -1;
    }

    public boolean isEmpty() {
        if (top == 1) {
            return true;
        } else {
            return false;
        }
```

```
    }

    public boolean isFull() {
        if (top == size - 1) {
            return true;
        } else {
            return false;
        }
    }

    public void push(int dt) {
        if (!isFull()) {
            top++;
            data[top] = dt;
        } else {
            System.out.println("Stack is full");
        }
    }

    public void pop() {
        if (!isEmpty()) {
            int x = data[top];
            top--;
            System.out.println("Remove data : " + x);
        } else {
            System.out.println("Stack is empty");
        }
    }

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

    public void print() {
        System.out.println("Stack content: ");
        for (int i = top; i >= 0; i--) {
            System.out.println(data[i] + " ");
        }
        System.out.println("");
    }

    public void clear() {
```

```
            if (!isEmpty()) {
                for (int i = top; i >= 0; i--) {
                    top--;
                }
                System.out.println("Stack is now empty");
            } else {
                System.out.println("Failed ! Stack is still empty");
            }
        }
    }
```

```
    package Practicum1;

    public class StackMain {
        public static void main(String[] args) {
            Stack stk = new Stack(5);

            stk.push(15);
            stk.push(27);
            stk.push(13);

            stk.print();

            stk.push(11);
            stk.push(34);
            stk.pop();
            stk.peek();
            stk.print();
        }
    }
```

### 1.2.2 Result

```
PS D:\Kuliah>  & 'C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\ASUS\AppData\Roaming\Code\User\workspaceStorage\
↪  ce3fcb236261368a6cbd019dc8ddda8b\redhat.java\
↪  jdt_ws\Kuliah_28156aa7\bin' 'Practicum1.StackMain'
Stack content:
13
27
15

```

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

### 1.2.3 Questions

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

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

   Answer:
   the number 5 is used to set the size of the stack

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

   ```
   package Practicum1;

   public class StackMain {
       public static void main(String[] args) {
           Stack stk = new Stack(5);

           stk.push(15);
           stk.push(27);
           stk.push(13);

           stk.print();

           stk.push(11);
           stk.push(34);
           stk.pop();
           stk.peek();
           stk.print();

           stk.push(18);
           stk.push(40);
       }
   }
   ```

```
PS D:\Kuliah>  & 'C:\Program
↪  Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\
↪  80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\
↪  Kuliah_28156aa7\bin' 'Practicum1.StackMain'
Stack content:
13
27
15

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

Stack is full
```

3. In previous number, the data inserted in to the stack is only 18 and 40 is not inserted. Why is that?
   Answer:
   Because after pushing the number 18, the stack is already full.

## 1.3 2nd Lab Activities

In this practicum, we will create a program to illustrate a bunch of books that are stored in Stack. Since the book has some information on it, the stack implementation is done using array of object to represent each element.

### 1.3.1 Steps

1. This class diagram is used for creating a program code written in Java programming language

| Book |
| --- |
| title: String<br>authorName: String<br>publishedYear: int<br>pagesAmount: int<br>price: int |
| Book(title: String, author: String, publishedYear: int, pagesAmount: int, price: int) |

2. Create a new package named **Practicum2**, then create a new class named **Book**.

3. Add attributes in that class, and add the constructor as well.

```
String title, authorName;
int publishedYear, pageAmount, price;

public Book(String tt, String nm, int yr, int pam, int pr) {
    this.title = tt;
    this.authorName = nm;
    this.publishedYear = yr;
    this.pageAmount = pam;
    this.price = pr;
}
```

4. Copy the program code for Stack class in **Practicum1** to be used again in here. Since the data stored in Stack in **Practicum1** is integer array, and in **Practicum2** we use objects, we will need to modify some parts in that class.

5. Modify the Stack class by changing the data type of **int data[]** to **Book data[]**. This time we will need to save the data in stack in objects. In addition, we will need to change the **attributes, constructor, method push**, and **method pop**

```
int size, top;
Book data[];

public Stack(int size) {
    this.size = size;
    data = new Book[size];
    top = -1;
}
```

```
public void push(Book dt) {
    if (!isFull()) {
        top++;
        data[top] = dt;
    } else {
        System.out.println("Stack is full");
    }
}
```

6. We will need to change the **print, pop, and peek method** as well since the data that are going to be printed is not only a string, but an object consists of some information (title, authorName, etc.).

```
public void pop() {
    if (!isEmpty()) {
        Book x = data[top];
        top--;
        System.out.println("Remove data : " + x.title + " " +
        ↪  x.authorName + " " + x.publishedYear + " " +
        ↪  x.publishedYear + " " + x.pageAmount + " " +
        ↪  x.price);
    } else {
        System.out.println("Stack is empty");
    }
}

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

public void print() {
    System.out.println("Stack content: ");
    for (int i = top; i >= 0; i--) {
        System.out.println(data[i].title + " " +
        ↪  data[i].authorName + " " + data[i].publishedYear + "
        ↪  " + data[i].pageAmount + " " + data[i].price);
    }
    System.out.println("");
}
```

7. Next, we have to create a new class called **StackMain** in **Practicum2**. Create a main function and instantiate an object with named **st**

8. Declare the **Scanner** object with name **sc**

9. Insert these lines of codes to receive **Book** data input, alongside with its information to be stored in stack

```
Stack st = new Stack(8);
Scanner sc = new Scanner(System.in);

char choose;
do {
    System.out.print("Title : ");
    String title = sc.nextLine();

    System.out.print("Author Name : ");
    String name = sc.nextLine();

    System.out.print("Published year : ");
    int year = sc.nextInt();

    System.out.print("Pages Amount : ");
    int pages = sc.nextInt();

    System.out.print("Price : ");
    int price = sc.nextInt();

    Book bk = new Book(title, name, year, pages, price);
    System.out.print("Do you want to add new data to Stack (y/n)?
    ↪  ");
    choose = sc.next().charAt(0);
    sc.nextLine();
    st.push(bk);

} while (choose == 'y');
```

10. Call print, pop, and peek method accordingly as follows:

```
st.print();
st.pop();
st.peek();
st.print();
```

11. Compile and run **StackMain**, and observe the result

```
package Practicum2;

public class Book {
```

```
    String title, authorName;
    int publishedYear, pageAmount, price;

    public Book(String tt, String nm, int yr, int pam, int pr) {
        this.title = tt;
        this.authorName = nm;
        this.publishedYear = yr;
        this.pageAmount = pam;
        this.price = pr;
    }
}
```

```
package Practicum2;

public class Stack {
    int size, top;
    Book data[];

    public Stack(int size) {
        this.size = size;
        data = new Book[size];
        top = -1;
    }

    public boolean isEmpty() {
        if (top == 1) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isFull() {
        if (top == size - 1) {
            return true;
        } else {
            return false;
        }
    }

    public void push(Book dt) {
        if (!isFull()) {
```

```
            top++;
            data[top] = dt;
        } else {
            System.out.println("Stack is full");
        }
    }

    public void pop() {
        if (!isEmpty()) {
            Book x = data[top];
            top--;
            System.out.println("Remove data : " + x.title + " " +
            ↪  x.authorName + " " + x.publishedYear + " " +
            ↪  x.publishedYear + " " + x.pageAmount + " " +
            ↪  x.price);
        } else {
            System.out.println("Stack is empty");
        }
    }

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

    public void print() {
        System.out.println("Stack content: ");
        for (int i = top; i >= 0; i--) {
            System.out.println(data[i].title + " " +
            ↪  data[i].authorName + " " + data[i].publishedYear
            ↪  + " " + data[i].pageAmount + " " +
            ↪  data[i].price);
        }
        System.out.println("");
    }

    public void clear() {
        if (!isEmpty()) {
            for (int i = top; i >= 0; i--) {
                top--;
            }
            System.out.println("Stack is now empty");
        } else {
```

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

```
package Practicum2;

import java.util.Scanner;

public class StackMain {
    public static void main(String[] args) {
        Stack st = new Stack(8);
        Scanner sc = new Scanner(System.in);

        char choose;
        do {
            System.out.print("Title : ");
            String title = sc.nextLine();

            System.out.print("Author Name : ");
            String name = sc.nextLine();

            System.out.print("Published year : ");
            int year = sc.nextInt();

            System.out.print("Pages Amount : ");
            int pages = sc.nextInt();

            System.out.print("Price : ");
            int price = sc.nextInt();

            Book bk = new Book(title, name, year, pages, price);
            System.out.print("Do you want to add new data to
            ↪  Stack (y/n)? ");
            choose = sc.next().charAt(0);
            sc.nextLine();
            st.push(bk);

        } while (choose == 'y');

        st.print();
        st.pop();
```

```
            st.peek();
            st.print();

            sc.close();
        }
    }
```

### 1.3.2 Result

```
PS D:\Kuliah>  d:; cd 'd:\Kuliah'; & 'C:\Program
↪  Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\ASUS\AppData\Roaming\Code\User\workspaceStorage\ce3fcb236261368a6cbd019
↪  'Practicum2.StackMain'
Title : Programming
Author Name : Burhantoro
Published year : 2016
Pages Amount : 126
Price : 58000
Do you want to add new data to Stack (y/n)? y
Title : Statistics
Author Name : Yasir
Published year : 2014
Pages Amount : 98
Price : 44000
Do you want to add new data to Stack (y/n)? y
Title : Economics
Author Name : Diana
Published year : 2019
Pages Amount : 86
Price : 47500
Do you want to add new data to Stack (y/n)? n
Stack content:
Economics Diana 2019 86 47500
Statistics Yasir 2014 98 44000
Programming Burhantoro 2016 126 58000

Remove data : Economics Diana 2019 2019 86 47500
Top element : Practicum2.Book@16b98e56
Stack content:
Statistics Yasir 2014 98 44000
Programming Burhantoro 2016 126 58000
```

### 1.3.3 Questions

1. In class StackMain, when calling **push** method, the argument is **bk**. What information is included in the **bk** variable?
   `Answer:`
   The **bk** variable contain the title of the book, the author's name, year the book published, the amount of pages in the book, and the price of the book.

2. Which of the program that its usage is to define the capacity of the stack ?
   `Answer:`
   The capacity of the stack is defined in the instantiation of the stack in the **StackMain** class as **st**, which refer to the **Stack** class.

3. What is the function of do-while that is exist in **StackMain** class?
   `Answer:`
   The do-while is used to enter the necessary data for the Book object to be inserted into the stack. After which the user is given the option to add more data or to stop. In this situation the do-while is used to ask the user to atleast input one object, in which after that they are given the option to add more or stop.

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!
   `Answer:`

```
package Practicum2;

import java.util.Scanner;

public class StackMain {
    static Stack st = new Stack(8);
    static Scanner sc = new Scanner(System.in);
    static void callPush() {
        System.out.print("Title : ");
        String title = sc.nextLine();

        System.out.print("Author Name : ");
        String name = sc.nextLine();

        System.out.print("Published year : ");
        int year = sc.nextInt();

        System.out.print("Pages Amount : ");
        int pages = sc.nextInt();
```

```
        System.out.print("Price : ");
        int price = sc.nextInt();

        Book bk = new Book(title, name, year, pages, price);
        st.push(bk);
    }
    public static void main(String[] args) {
        char choose;
        do {
            System.out.print("Title : ");
            String title = sc.nextLine();

            System.out.print("Author Name : ");
            String name = sc.nextLine();

            System.out.print("Published year : ");
            int year = sc.nextInt();

            System.out.print("Pages Amount : ");
            int pages = sc.nextInt();

            System.out.print("Price : ");
            int price = sc.nextInt();

            Book bk = new Book(title, name, year, pages, price);
            System.out.print("Do you want to add new data to
            ↪  Stack (y/n)? ");
            choose = sc.next().charAt(0);
            sc.nextLine();
            st.push(bk);

        } while (choose == 'y');

        /*
        st.print();
        st.pop();
        st.peek();
        st.print();
        */

        Boolean done = false;
```

```
            while (!done) {
                System.out.println("Which operation would you like to
                ↪ do?");
                System.out.println("1. push");
                System.out.println("2. pop");
                System.out.println("3. peek");
                System.out.println("4. print");
                System.out.print("Menu: ");
                char menu = sc.nextLine().charAt(0);
                switch (menu) {
                    case '1':
                        callPush();
                        break;
                    case '2':
                        st.pop();
                        break;
                    case '3':
                        st.peek();
                        break;
                    case '4':
                        st.print();
                        break;
                    default:
                        System.out.println("Please insert number as
                        ↪ displayed in the menu!");
                        break;
                    }
                System.out.println("Would you like to do another
                ↪ operation?");
                System.out.print("Menu (Y/n): ");
                char option = sc.nextLine().charAt(0);
                if (option == 'n' || option == 'N') {
                    done = true;
                }
            }
            sc.close();
        }
    }
```

## 1.4 3rd Lab Activities

In this practicum, we will create program to convert infix notation into postfix notation

### 1.4.1 Steps

1. We will use class diagram to create **Postfix** class in Java program

| **Postfix** |
| --- |
| n: int<br>top: int<br>stack: char[] |
| Postfix(total: int)<br>push(c: char): void<br>pop(): void<br>IsOperand(c: char): boolean<br>IsOperator(c: char): boolean<br>degree(c: char): int<br>convert(Q: String): string |

2. Create a package named **Practicum3**. Then, we create a new class named **Postfix**. Add attributes **n, top, and stack** based on class diagram above.

```
package Practicum3;

public class Postfix {
    int n, top;
    char[] stack;
}
```

3. Add a constructor with parameter as follows:

```
public Postfix(int total) {
    n = total;
    top = -1;
    stack = new char[n];
    push('(');
}
```

4. Create method **push** and **pop** with void as its return type

```
public void push(char c) {
    top++;
    stack[top] = c;
```

```
}

public char pop() {
    char item = stack[top];
    top--;
    return item;
}
```

5. Create method **isOperand** as Boolean that will be used to check if the element is operand or not

```
public boolean isOperand(char c) {
    if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') || (c >=
    ↪  '0' && c <= '9') || c == ' ' || c == '.') {
        return true;
    } else {
        return false;
    }
}
```

6. Create method **isOperator** as booelan that will be used to check if the element is operator or not

```
public boolean isOperator(char c) {
    if (c == '^' || c == '%' || c == '/' || c == '*' || c == '-'
    ↪  || c == '+') {
        return true;
    } else {
        return false;
    }
}
```

7. Create method **degree** as integer to define the degree of the operator

```
public int degree(char c) {
    switch (c) {
        case '^':
            return 3;
        case '%':
            return 2;
        case '/':
            return 2;
        case '*':
            return 2;
```

```
            case '+':
                return 1;
            case '-':
                return 1;
            default:
                return 0;
        }
    }
```

8. Create method **convert** to convert infix notation to postfix notation by checking the element one by one in data element.

```
public String convert(String Q) {
    String p = "";
    char c;
    for (int i = 0; i < n; i++) {
        c = Q.charAt(i);
        if (isOperand(c)) {
            p = p + c;
        }
        if (c == '(') {
            push(c);
        }
        if (c == ')') {
            while (stack[top] != '(') {
                p = p + pop();
            }
            pop();
        }
        if (isOperator(c)) {
            while (degree(stack[top]) > degree(c)) {
                p = p + pop();
            }
            push(c);
        }
    }
    return p;
}
```

9. Next, we will need create a class named **PostfixMain**. After creating the main function, we create a variable P and Q. P variable will be used to store the final result of converted postfix notation, while Q variable is used to store user input in the form mathematical expression written in infix notation. Instantiate

the Scanner object with **sc** variable, then call build-in **trim** method to remove spaces within a string.

```
Scanner sc = new Scanner(System.in);
String P, Q;
System.out.println("Insert mathematical expression (infix) : ");
Q = sc.nextLine();
Q = Q.trim();
Q = Q + ")";
```

We need to add string "**)**" to ensure all symbol/ characters that are exist in the stack will be retrieved and moved in postfix.

10. Create a **total** variable to calculate how many characters in variable Q

```
int total = Q.length();
```

11. Instantiate object **post** with **total** as the argument. Then, call **convert** method to change the infix notation in Q string to postfix notation P

```
Postfix post = new Postfix(total);
P = post.convert(Q);
System.out.println("Postfix : " + P);
```

12. Compile and run **StackMain**, and observe the result

```
package Practicum3;

public class Postfix {
    int n, top;
    char[] stack;

    public Postfix(int total) {
        n = total;
        top = -1;
        stack = new char[n];
        push('(');
    }

    public void push(char c) {
        top++;
        stack[top] = c;
    }

    public char pop() {
```

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

    public boolean isOperand(char c) {
        if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') ||
        ↪  (c >= '0' && c <= '9') || c == ' ' || c == '.') {
            return true;
        } else {
            return false;
        }
    }

    public boolean isOperator(char c) {
        if (c == '^' || c == '%' || c == '/' || c == '*' || c ==
        ↪  '-' || c == '+') {
            return true;
        } else {
            return false;
        }
    }

    public int degree(char c) {
        switch (c) {
            case '^':
                return 3;
            case '%':
                return 2;
            case '/':
                return 2;
            case '*':
                return 2;
            case '+':
                return 1;
            case '-':
                return 1;
            default:
                return 0;
        }
    }
```

```
    public String convert(String Q) {
        String p = "";
        char c;
        for (int i = 0; i < n; i++) {
            c = Q.charAt(i);
            if (isOperand(c)) {
                p = p + c;
            }
            if (c == '(') {
                push(c);
            }
            if (c == ')') {
                while (stack[top] != '(') {
                    p = p + pop();
                }
                pop();
            }
            if (isOperator(c)) {
                while (degree(stack[top]) > degree(c)) {
                    p = p + pop();
                }
                push(c);
            }
        }
        return p;
    }
}
```

```
package Practicum3;

import java.util.Scanner;

public class PostfixMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String P, Q;
        System.out.println("Insert mathematical expression
        ↪  (infix) : ");
        Q = sc.nextLine();
        Q = Q.trim();
        Q = Q + ")";
```

```
            int total = Q.length();

            Postfix post = new Postfix(total);
            P = post.convert(Q);
            System.out.println("Postfix : " + P);

            sc.close();
        }
    }
```

### 1.4.2 Result

```
PS D:\Kuliah>  d:; cd 'd:\Kuliah'; & 'C:\Program
↪  Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\
↪  80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\
↪  Kuliah_28156aa7\bin' 'Practicum3.PostfixMain'
Insert mathematical expression (infix) :
a+b*(c+d-e)/f
Postfix : abcde-+f/*+
```

### 1.4.3 Questions

1. Please explain the flow of method in Postfix class
   Answer:
   The algorithm scan for the problem from the first character. If the algorithm detects an operand, it will put it in the stack. If the algorithm detects an open parentheses, it will put it in the stack. If the algorithm detects a closed parentheses, it will pop trough the stack until it found the open parentheses. If the algorithm detects an operator, then it will check if the stack is empty. If the stack is empty, it will push the operator to the stack. If there is an operator in the stack, it will check if the degree of the operator is bigger than the top of the stack. If it is bigger, the operator will be push. If not, then the stack will be popped until the top of the stack degree is lower than the current operator. If all of the equation has been read. then it will pop everything in the stack. Everytime the stack is popped it will push it to the postfix except for the parentheses.

2. What is the function of this program code?

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

Answer:
The varibale c is used to store the character the for loop is currently itterating trough the string that has been inputed.

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

```
PS D:\Kuliah>  & 'C:\Program
↪  Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\
↪  80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\
↪  Kuliah_28156aa7\bin' 'Practicum3.PostfixMain'
Insert mathematical expression (infix) :
3*5^(8-6)%3
Postfix : 3586-^3%*
```

4. In 2nd number, why the braces are not displayed in conversion result? Please explain
Answer:
Because in the postfix algorithm, the closed parentheses are only used as limiter to pop trough the stack when the open parentheses is founded. In which both do not get printed.

## 1.5 Assignment

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

```
run:
Inser Sentence: Politeknik Negeri Malang
Result :
gnalaM iregeN kinketiloP
BUILD SUCCESSFUL (total time: 1 second)
```

```
package Assignment1;

public class Stack {
    int size;
    int top;
    char data[];

    public Stack(int size) {
```

```
        this.size = size;
        data = new char[size];
        top = -1;
    }

    public boolean isEmpty() {
        if (top == 1) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isFull() {
        if (top == size - 1) {
            return true;
        } else {
            return false;
        }
    }

    public void push(char dt) {
        if (!isFull()) {
            top++;
            data[top] = dt;
        } else {
            System.out.println("Stack is full");
        }
    }

    public void pop() {
        if (!isEmpty()) {
            char x = data[top];
            top--;
            System.out.println("Remove data : " + x);
        } else {
            System.out.println("Stack is empty");
        }
    }

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

```
    }

    public void print() {
        System.out.println("Result: ");
        for (int i = top; i >= 0; i--) {
            System.out.print(data[i]);
        }
    }

    public void clear() {
        if (!isEmpty()) {
            for (int i = top; i >= 0; i--) {
                top--;
            }
            System.out.println("Stack is now empty");
        } else {
            System.out.println("Failed ! Stack is still empty");
        }
    }
}
```

```
package Assignment1;

import java.util.Scanner;

public class StackMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Insert Sentence: ");
        String sentence = sc.nextLine();

        Stack stk = new Stack(sentence.length());
        for (int i = 0; i < sentence.length(); i++) {
            stk.push(sentence.charAt(i));
        }
        stk.print();

        sc.close();
    }
}
```

```
PS D:\Kuliah>  d:; cd 'd:\Kuliah'; & 'C:\Program
↪  Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\
↪  80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\
↪  Kuliah_28156aa7\bin' 'Assignment1.StackMain'
Insert Sentence: Politeknik Negeri Malang
Result:
gnalaM iregeN kinketiloP
```

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 the receipts. The information that are included in a receipt are as follows:

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

```java
package Assignment2;

public class Receipts {
    int transactionID, quantity;
    String date;
    double transactionTotal;

    public Receipts(int transactionID, String date, int quantity,
    ↪  double transactionTotal) {
        this.transactionID = transactionID;
        this.date = date;
        this.quantity = quantity;
        this.transactionTotal = transactionTotal;
    }
}
```

```java
package Assignment2;
```

```
public class Stack {
    int size, top;
    Receipts data[];

    public Stack(int size) {
        this.size = size;
        data = new Receipts[size];
        top = -1;
    }

    public boolean isEmpty() {
        if (top == 1) {
            return true;
        } else {
            return false;
        }
    }

    public boolean isFull() {
        if (top == size - 1) {
            return true;
        } else {
            return false;
        }
    }

    public void push(Receipts dt) {
        if (!isFull()) {
            top++;
            data[top] = dt;
        } else {
            System.out.println("Stack is full");
        }
    }

    public void pop() {
        if (!isEmpty()) {
            Receipts x = data[top];
            top--;
            System.out.println("Remove data : " + x.transactionID
            ↪  + " " + x.date + " " + x.quantity + " " +
            ↪  x.transactionTotal);
```

```
        } else {
            System.out.println("Stack is empty");
        }
    }

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

    public void print(int limit) {
        System.out.printf("last %d data: \n", limit);
        for (int i = top; i >= (size - limit); i--) {
            System.out.println(data[i].transactionID + " " +
            ↪  data[i].date + " " + data[i].quantity + " " +
            ↪  data[i].transactionTotal);
        }
        System.out.println("");
    }

    public void clear() {
        if (!isEmpty()) {
            for (int i = top; i >= 0; i--) {
                top--;
            }
            System.out.println("Stack is now empty");
        } else {
            System.out.println("Failed ! Stack is still empty");
        }
    }
}
```

```
package Assignment2;

public class StackMain {
    public static void main(String[] args) {
        Stack stack = new Stack(8);

        int[] transactionIDs = {1, 2, 3, 4, 5, 6, 7, 8};
        String[] date = {" 7 February 2023", "14 February 2023",
        ↪  "21 February 2023", "28 February 2023", " 7 March
        ↪  2023", "14 March 2023", "21 March 2023", "28 March
        ↪  2023"};
```

```
            int[] quantity = {5, 7, 8, 6, 7, 7, 5, 6};
            double[] transactionTotal = {85_000, 115_000, 150_000,
            ↪  100_000, 115_000, 115_000, 85_000, 100_000};

            for (int i = 0; i < stack.size; i++) {
                System.out.printf("%d %s %d %,.2f\n",
                ↪  transactionIDs[i], date[i], quantity[i],
                ↪  transactionTotal[i]);
            }

            System.out.println();

            for (int i = 0; i < stack.size; i++) {
                Receipts receipts = new Receipts(transactionIDs[i],
                ↪  date[i], quantity[i], transactionTotal[i]);
                stack.push(receipts);
            }

            int requestedReceipts = 5;
            stack.print(requestedReceipts);
        }
    }
```

```
PS D:\Kuliah>  d:; cd 'd:\Kuliah'; & 'C:\Program
↪  Files\Java\jdk-18.0.2.1\bin\java.exe'
↪  '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
↪  'C:\Users\G4CE-PC\AppData\Roaming\Code\User\workspaceStorage\
↪  80d97a47d24665dc0bce7ab1e048ecbd\redhat.java\jdt_ws\
↪  Kuliah_28156aa7\bin' 'Assignment2.StackMain'
1  7 February 2023 5 85,000.00
2 14 February 2023 7 115,000.00
3 21 February 2023 8 150,000.00
4 28 February 2023 6 100,000.00
5  7 March 2023 7 115,000.00
6 14 March 2023 7 115,000.00
7 21 March 2023 5 85,000.00
8 28 March 2023 6 100,000.00

last 5 data:
8 28 March 2023 6 100000.0
7 21 March 2023 5 85000.0
6 14 March 2023 7 115000.0
```

```
15  5  7 March 2023 7 115000.0
16  4 28 February 2023 6 100000.0
```