JOB SHEET VIII Stack

From:

AL AZHAR RIZQI RIFA'I FIRDAUS

Class:

1 I

Absence:

01

Student Number Identity:

2241720263

Department:

Information Technology

Study Program:

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 top: int data[]: int
Stack(size: int) IsEmpty(): boolean IsFull(): boolean push(): void pop(): void peek(): void print(): void 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**

```
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

1.2.2. Result

Check if the result match with following image:

```
run:
Stack content:
13
27

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

BUILD SUCCESSFUL (total time: 0 seconds)
```

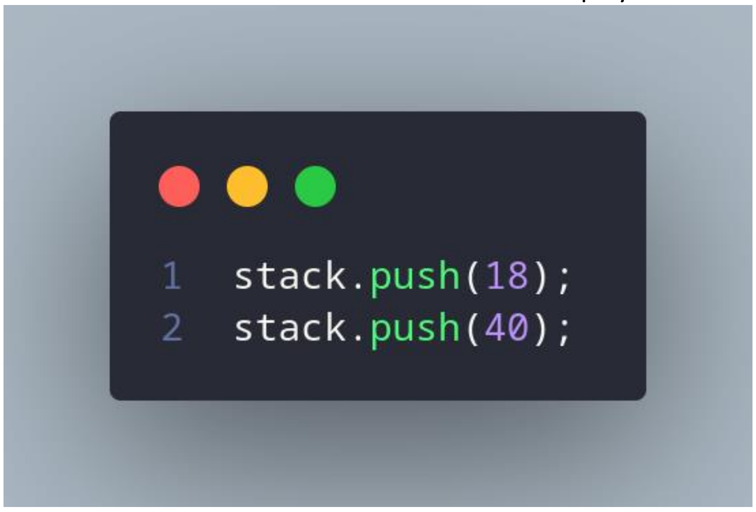
1.2.3. Questions

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

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

   - The number 5 represents the capacity of the stk object that is being created using the Stack class.

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



```
stack.push(18);
stack.push(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 capacity of the stack is set to only 5. When we push 18 using .push() method the IsFull() method willl be return true it means the stack 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.

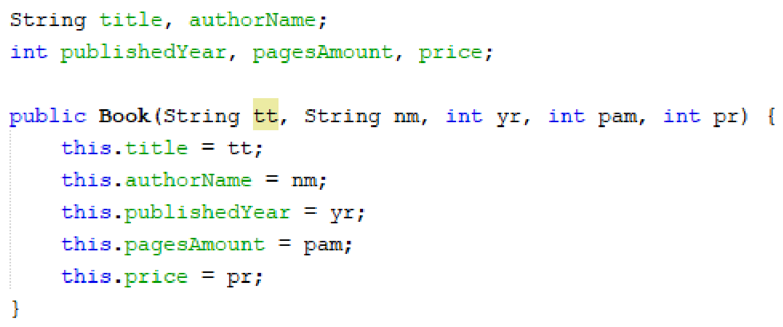
1.3.1. Steps

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

Book
title: String authorName: String publishedYear: int pagesAmount: int 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.



```java
String title, authorName;
int publishedYear, pagesAmount, price;

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

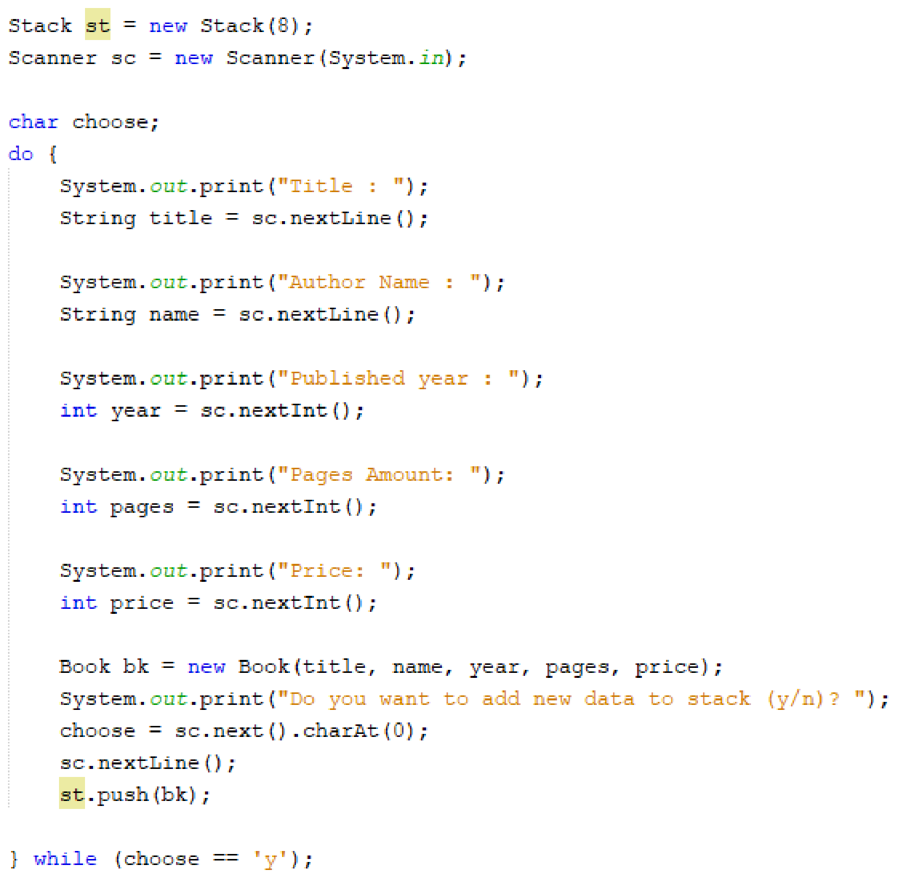
6. We will need to change the **print, pop, and peek method** as well since the data that are going to be printed is not only a string, but an object consists of some information (title, authorName, etc.).

```
public void pop(){
    if(!isEmpty()){
        Book x = data[top];
        top--;
        System.out.println("Removed data : " + x.title + " " +
                x.authorName + " " + x.publishedYear + " " +
                x.pagesAmount + " " + 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].pagesAmount + " " + 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**

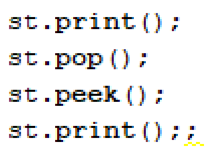
8. Declare the **Scanner** object with name **sc**

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:

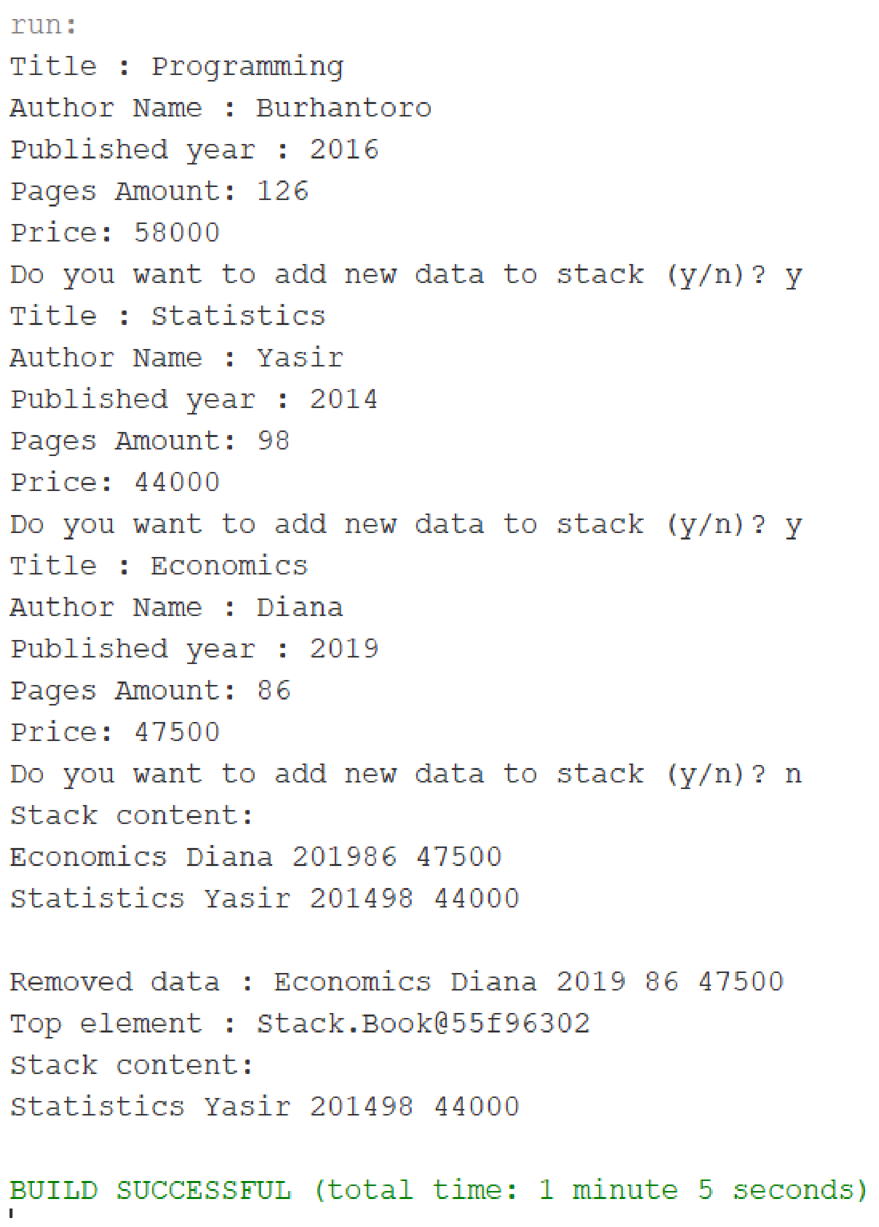


```
st.print();
st.pop();
st.peek();
st.print();;
```

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

1.3.2. Result

Check if the result match with following image:



```
run:
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 201986 47500
Statistics Yasir 201498 44000

Removed data : Economics Diana 2019 86 47500
Top element : Stack.Book@55f96302
Stack content:
Statistics Yasir 201498 44000

BUILD SUCCESSFUL (total time: 1 minute 5 seconds)
```

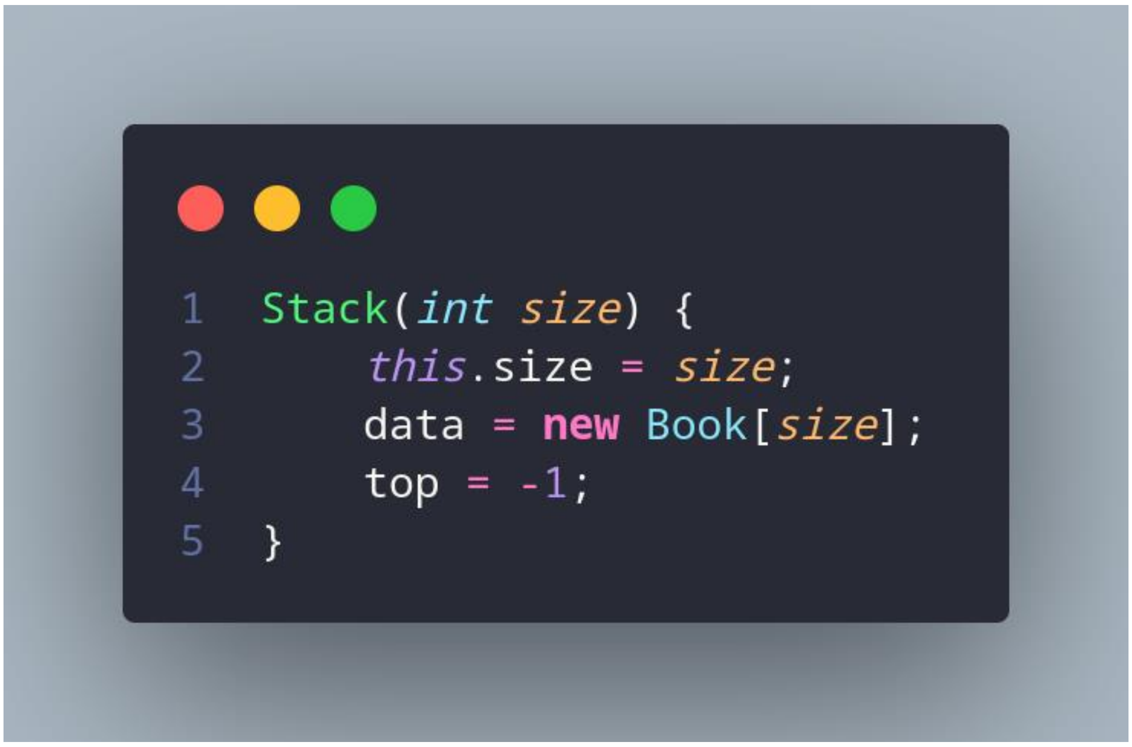
1.3.3. Questions

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

   - bk variable is an instance of the Book class, created with the following parameters passed to its constructor title, name, year, pages, and price.

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

   - in the Stack.class



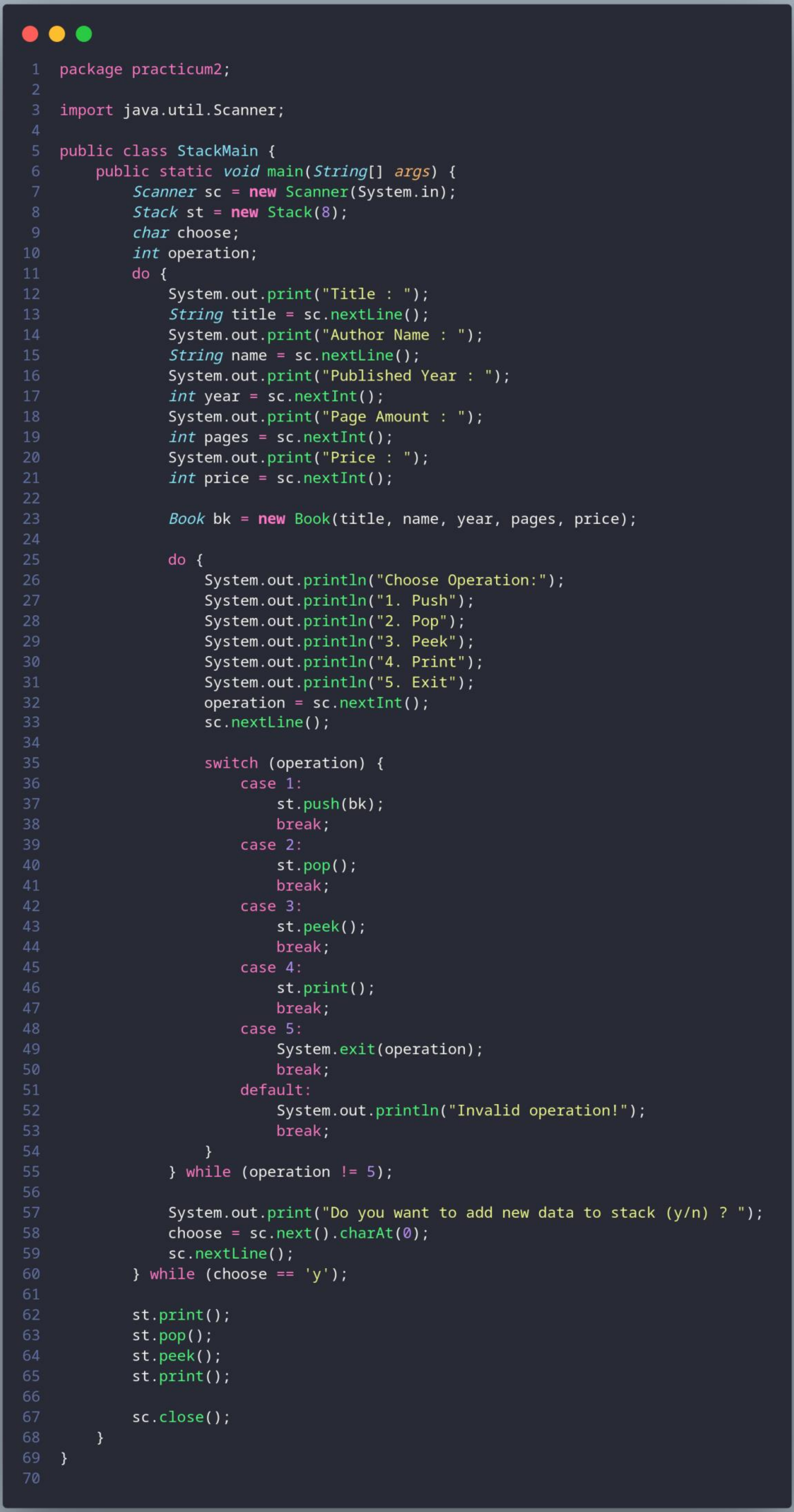
```java
Stack(int size) {
    this.size = size;
    data = new Book[size];
    top = -1;
}
```

3. What is the function of do-while that is exist in **StackMain** class?

   - To do itteration that contains input book data then every single itteration data will be stored at bk variable, and then will be push with .push() method.

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!

   Code :



```java
package practicum2;

import java.util.Scanner;

public class StackMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Stack st = new Stack(8);
        char choose;
        int operation;
        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("Page Amount : ");
            int pages = sc.nextInt();
            System.out.print("Price : ");
            int price = sc.nextInt();

            Book bk = new Book(title, name, year, pages, price);

            do {
                System.out.println("Choose Operation:");
                System.out.println("1. Push");
                System.out.println("2. Pop");
                System.out.println("3. Peek");
                System.out.println("4. Print");
                System.out.println("5. Exit");
                operation = sc.nextInt();
                sc.nextLine();

                switch (operation) {
                    case 1:
                        st.push(bk);
                        break;
                    case 2:
                        st.pop();
                        break;
                    case 3:
                        st.peek();
                        break;
                    case 4:
                        st.print();
                        break;
                    case 5:
                        System.exit(operation);
                        break;
                    default:
                        System.out.println("Invalid operation!");
                        break;
                }
            } while (operation != 5);

            System.out.print("Do you want to add new data to stack (y/n) ? ");
            choose = sc.next().charAt(0);
            sc.nextLine();
        } while (choose == 'y');

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

        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 top: int stack: char[]
Postfix(total: int) push(c: char): void pop(): void IsOperand(c: char): boolean IsOperator(c: char): boolean degree(c: char): int 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.
3. Add a constructor with parameter as follows:

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

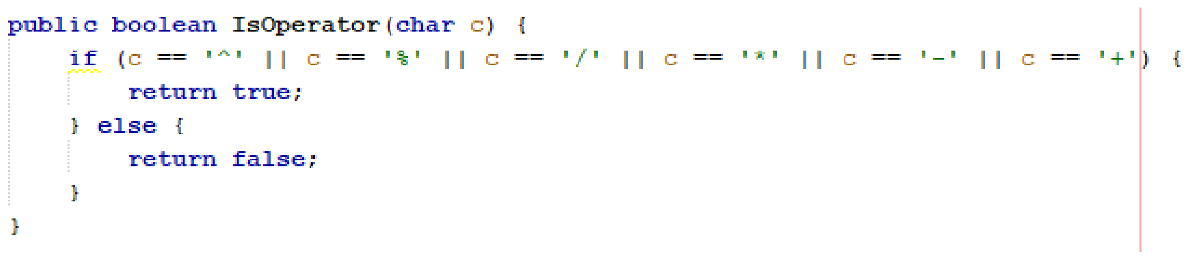
4. Create method **push** and **pop** with void as its return type

```java
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

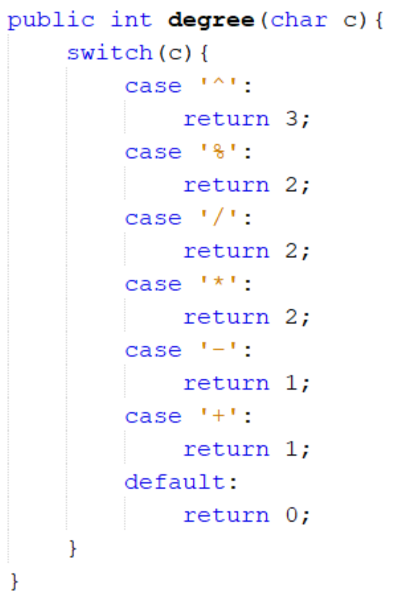
```java
public boolean IsOperand(char c) {
    if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') ||
            (c >= '0' && c <= '9') || c == ' ' || c == '.') {
        return true;
    } else {
        return false;
    }
}
```

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



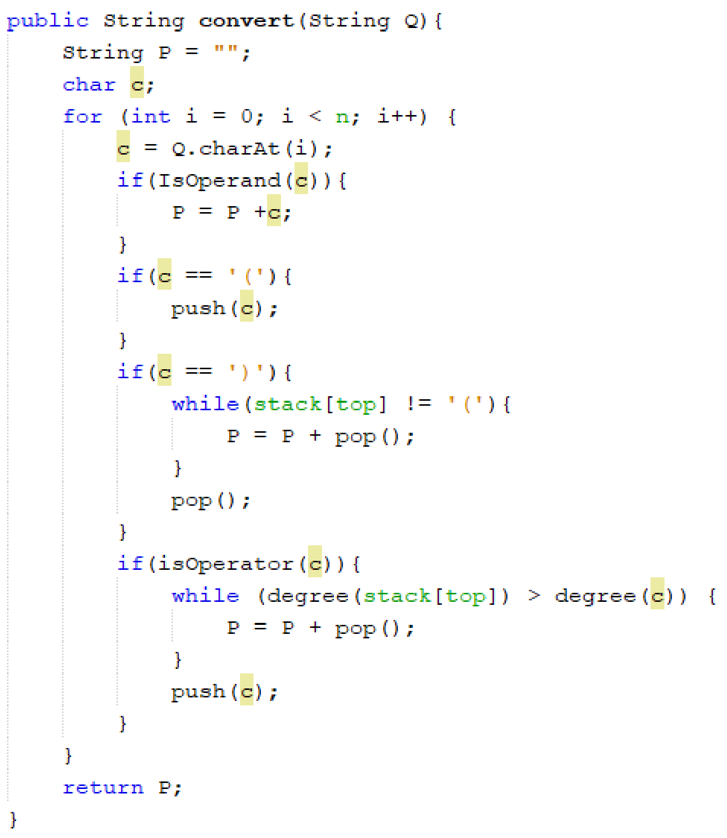
```java
public boolean IsOperator(char c) {
    if (c == '^' || c == '%' || c == '/' || c == '*' || c == '-' || c == '+') {
        return true;
    } else {
        return false;
    }
}
```

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



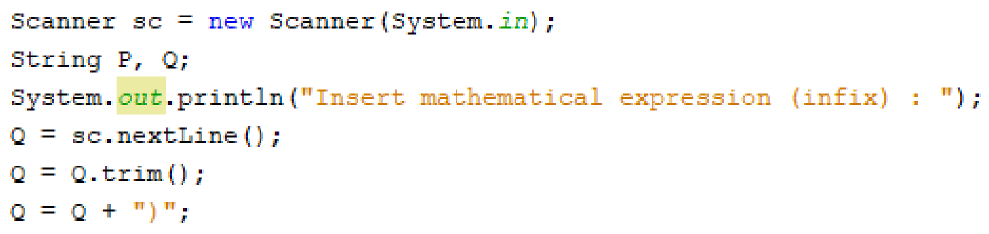
```java
public int degree(char c){
    switch(c){
        case '^':
            return 3;
        case '%':
            return 2;
        case '/':
            return 2;
        case '*':
            return 2;
        case '-':
            return 1;
        case '+':
            return 1;
        default: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

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



```java
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;
}
```

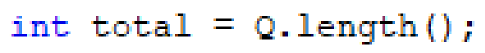
9. Next, we will need create a class named **PostfixMain.** After creating the main function, we create a variable P and Q. P variable will be used to store the final result of converted postfix notation, while Q variable is used to store user input in the form mathematical expression written in infix notation. Instantiate the Scanner object with **sc** variable, then call build-in **trim** method to remove spaces within a string.



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

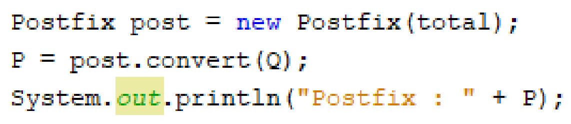
   We need to add string **")"** to ensure all symbol/ characters that are exist in the stack will be retrieved and moved in postfix.

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



```java
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

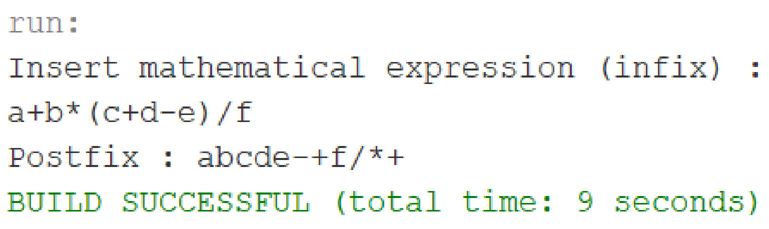


```java
Postfix post = new Postfix(total);
P = post.convert(Q);
System.out.println("Postfix : " + P);
```

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

12.1.1. Result

Check if the result match with following image:



```
run:
Insert mathematical expression (infix) :
a+b*(c+d-e)/f
Postfix : abcde-+f/*+
BUILD SUCCESSFUL (total time: 9 seconds)
```

12.1.2. Questions

1. Please explain the flow of method in **Postfix** class
   Constructor Postfix(int total):

   Initializes the n instance variable to the total parameter value.
   Initializes the top instance variable to -1, indicating the stack is initially empty.
   Creates a new char array stack with size n.
   Calls the push() method to push a left parenthesis character '(' onto the stack.

   Method push(char c):

   Increments the top instance variable by 1 to prepare for the new character to be added.
   Adds the character c to the stack array at the top index.

   Method pop():

Gets the character at the top index of the stack array and assigns it to the item variable.
Decrements the top instance variable by 1 to remove the character from the stack.
Returns the item variable containing the removed character.

Method isOperand(char c):

Returns true if the character c is an operand (i.e., a letter, digit, space or period) and false otherwise.

Method isOperator(char c):

Returns true if the character c is an operator (i.e., one of '^', '%', '/', '*', '-', or '+') and false otherwise.

Method degree(char c):

Returns the degree of precedence of the operator c. The operator with the highest degree of precedence has the highest integer value returned.

Method convert(String Q):

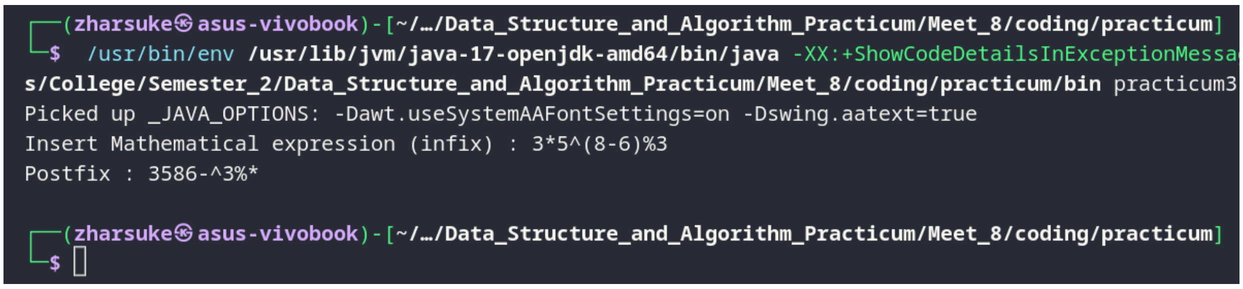
Initializes an empty string p to store the postfix expression.
Iterates through each character c in the infix expression Q.
If c is an operand, it is added to the p string.
If c is a left parenthesis, it is pushed onto the stack.
If c is a right parenthesis, all characters in the stack are popped off and added to the p string until a left parenthesis is found and removed.
If c is an operator, all operators with a higher precedence than c are popped off the stack and added to the p string before c is pushed onto the stack.
After all characters in Q have been processed, any remaining operators in the stack are popped off and added to the p string.
The postfix expression p is returned.

2. What is the function of this program code?

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

retrieves the character at index i from the string Q and assigns it to the variable c.

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



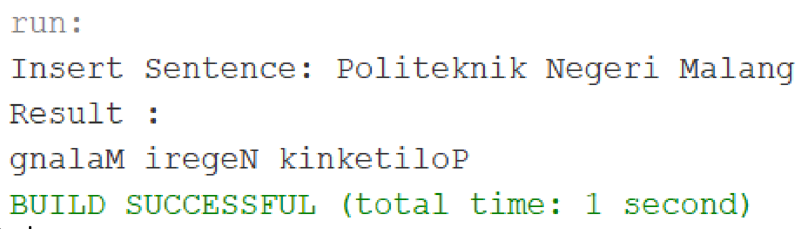
```
┌──(zharsuke㉿asus-vivobook)-[~/…/Data_Structure_and_Algorithm_Practicum/Meet_8/coding/practicum]
└─$  /usr/bin/env /usr/lib/jvm/java-17-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessa
s/College/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_8/coding/practicum/bin practicum3
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Insert Mathematical expression (infix) : 3*5^(8-6)%3
Postfix : 3586-^3%*

┌──(zharsuke㉿asus-vivobook)-[~/…/Data_Structure_and_Algorithm_Practicum/Meet_8/coding/practicum]
└─$ 
```

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

- The reason why the braces are not visible in the converted output is because the Postfix class's convert method solely utilizes the parentheses to establish the order of operations and does not include them in the resultant postfix expression. Whenever the method encounters an opening parenthesis, it places it onto the stack. On the other hand, when it encounters a closing parenthesis, it pops the operators from the stack and adds them to the postfix expression until it locates the opening parenthesis, which is then removed from the stack. Hence, the parentheses serve as a mechanism to ensure the accurate order of operations, and they are not a constituent of the final postfix expression.

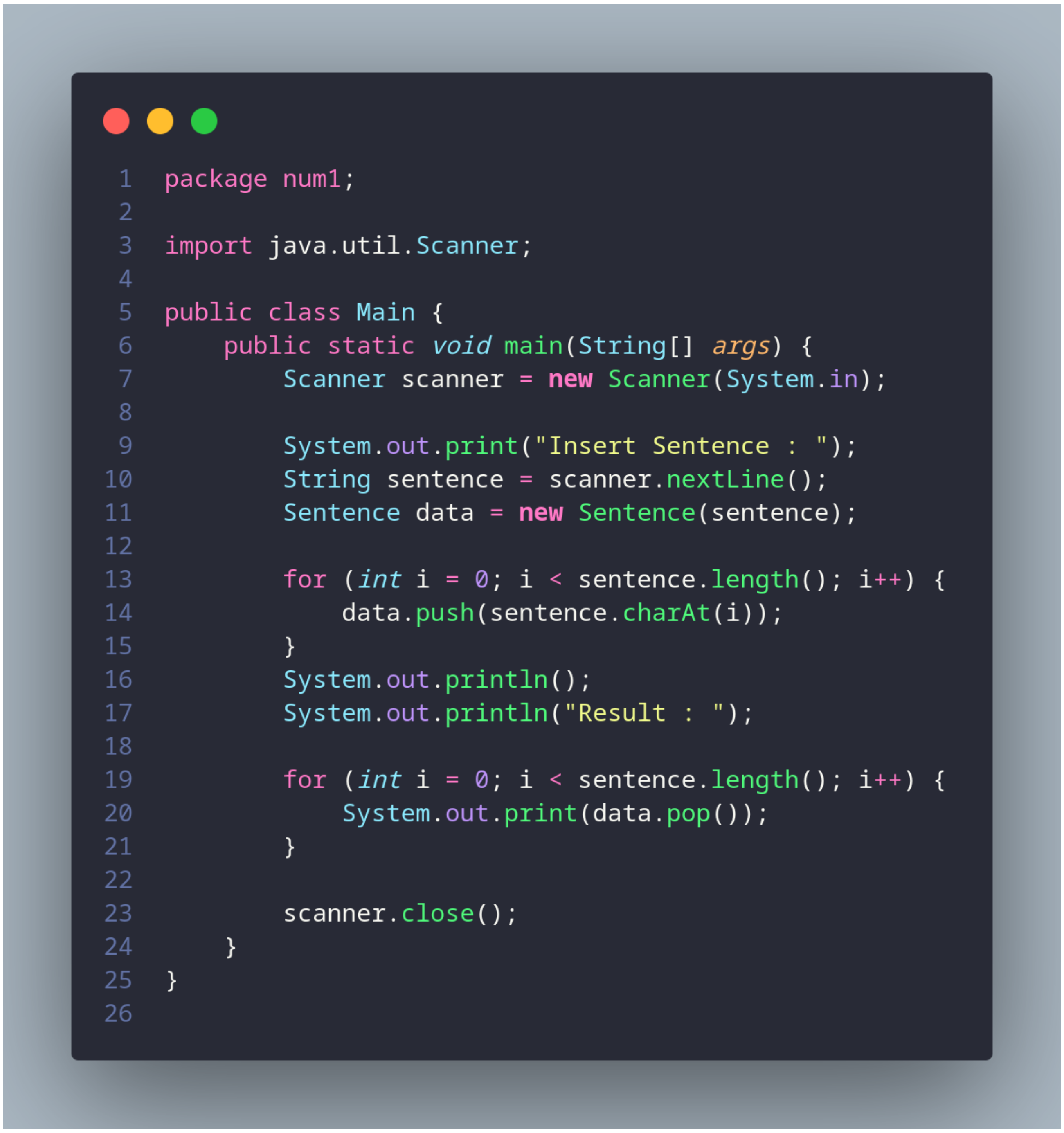
12.2. Assignment

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

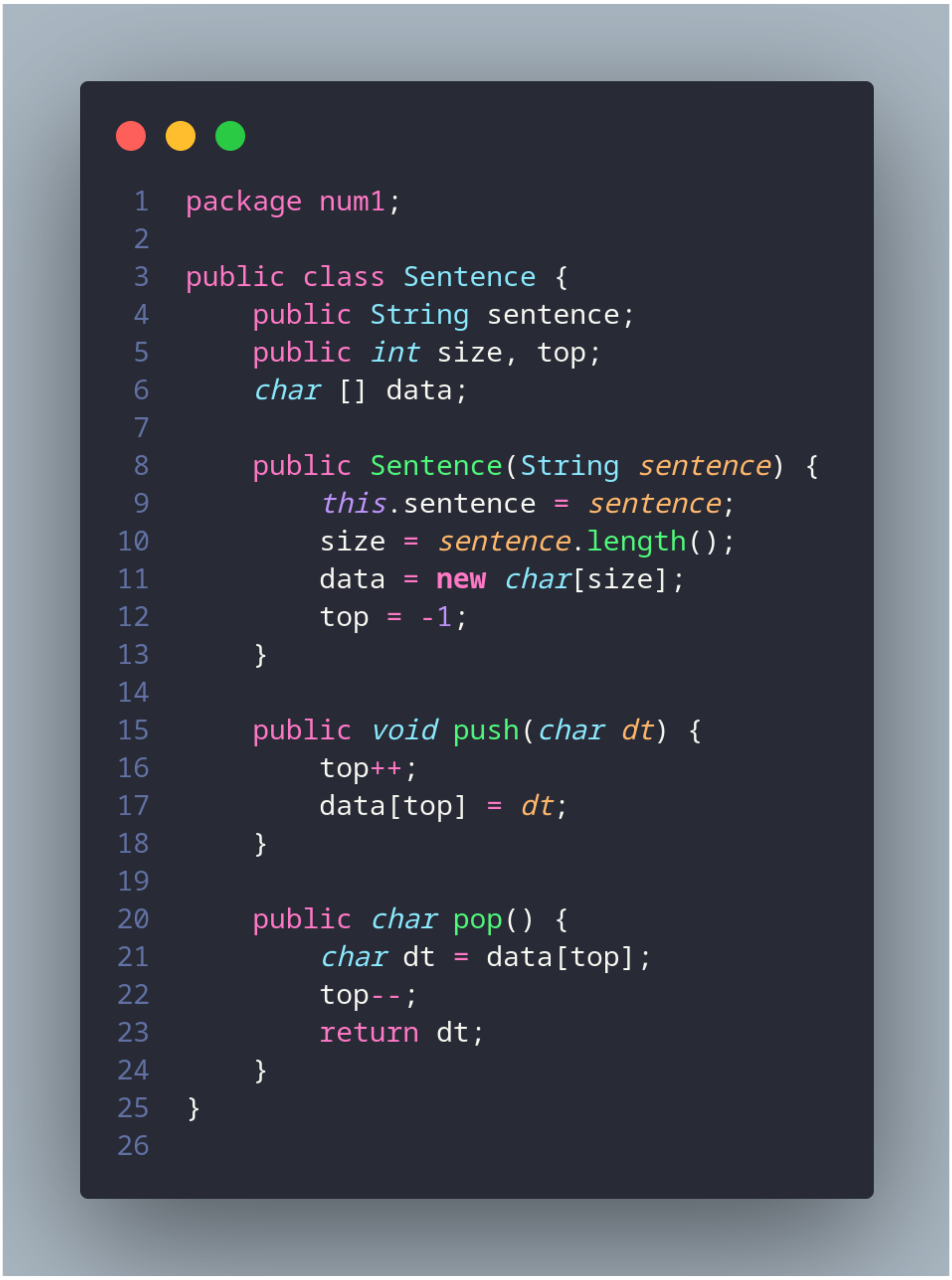


```
run:
Insert Sentence: Politeknik Negeri Malang
Result :
gnalaM iregeN kinketiloP
BUILD SUCCESSFUL (total time: 1 second)
```

Code :

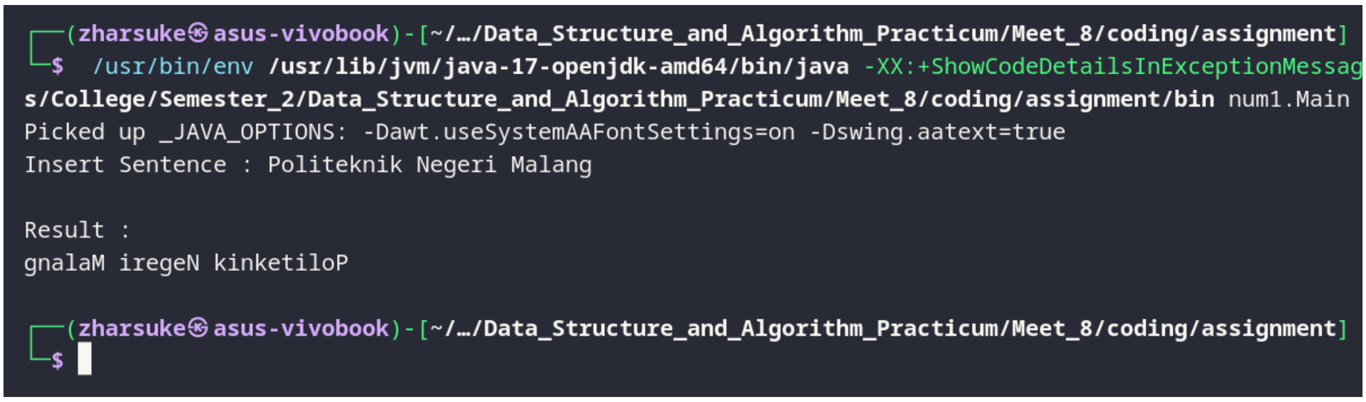


```java
package num1;

import java.util.Scanner;

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

        System.out.print("Insert Sentence : ");
        String sentence = scanner.nextLine();
        Sentence data = new Sentence(sentence);

        for (int i = 0; i < sentence.length(); i++) {
            data.push(sentence.charAt(i));
        }
        System.out.println();
        System.out.println("Result : ");

        for (int i = 0; i < sentence.length(); i++) {
            System.out.print(data.pop());
        }

        scanner.close();
    }
}

```



```java
package num1;

public class Sentence {
    public String sentence;
    public int size, top;
    char [] data;

    public Sentence(String sentence) {
        this.sentence = sentence;
        size = sentence.length();
        data = new char[size];
        top = -1;
    }

    public void push(char dt) {
        top++;
        data[top] = dt;
    }

    public char pop() {
        char dt = data[top];
        top--;
        return dt;
    }
}

```

Result :

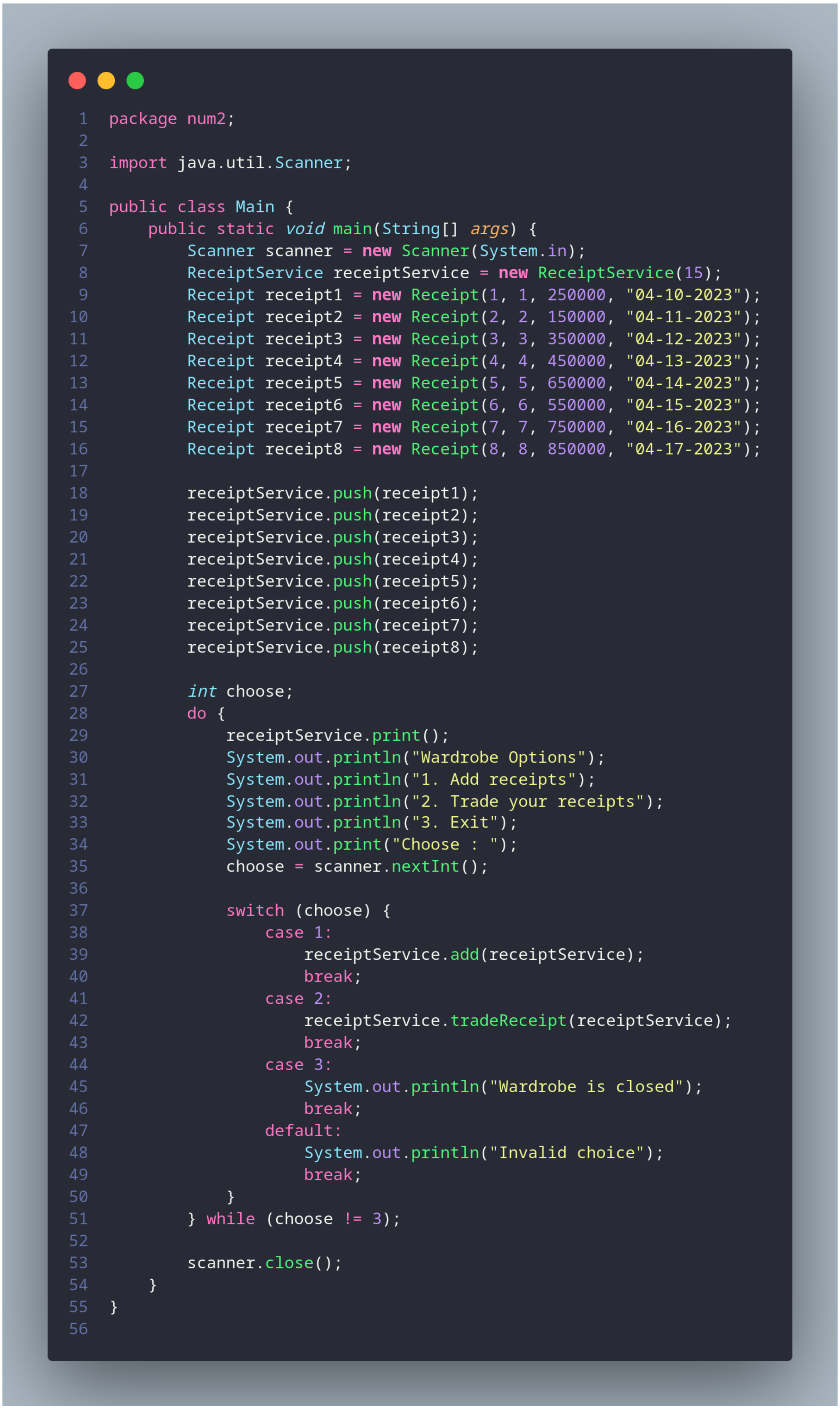


```
┌──(zharsuke㉿asus-vivobook)-[~/…/Data_Structure_and_Algorithm_Practicum/Meet_8/coding/assignment]
└─$  /usr/bin/env /usr/lib/jvm/java-17-openjdk-amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessag
s/College/Semester_2/Data_Structure_and_Algorithm_Practicum/Meet_8/coding/assignment/bin num1.Main
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Insert Sentence : Politeknik Negeri Malang

Result :
gnalaM iregeN kinketiloP

┌──(zharsuke㉿asus-vivobook)-[~/…/Data_Structure_and_Algorithm_Practicum/Meet_8/coding/assignment]
└─$ 
```

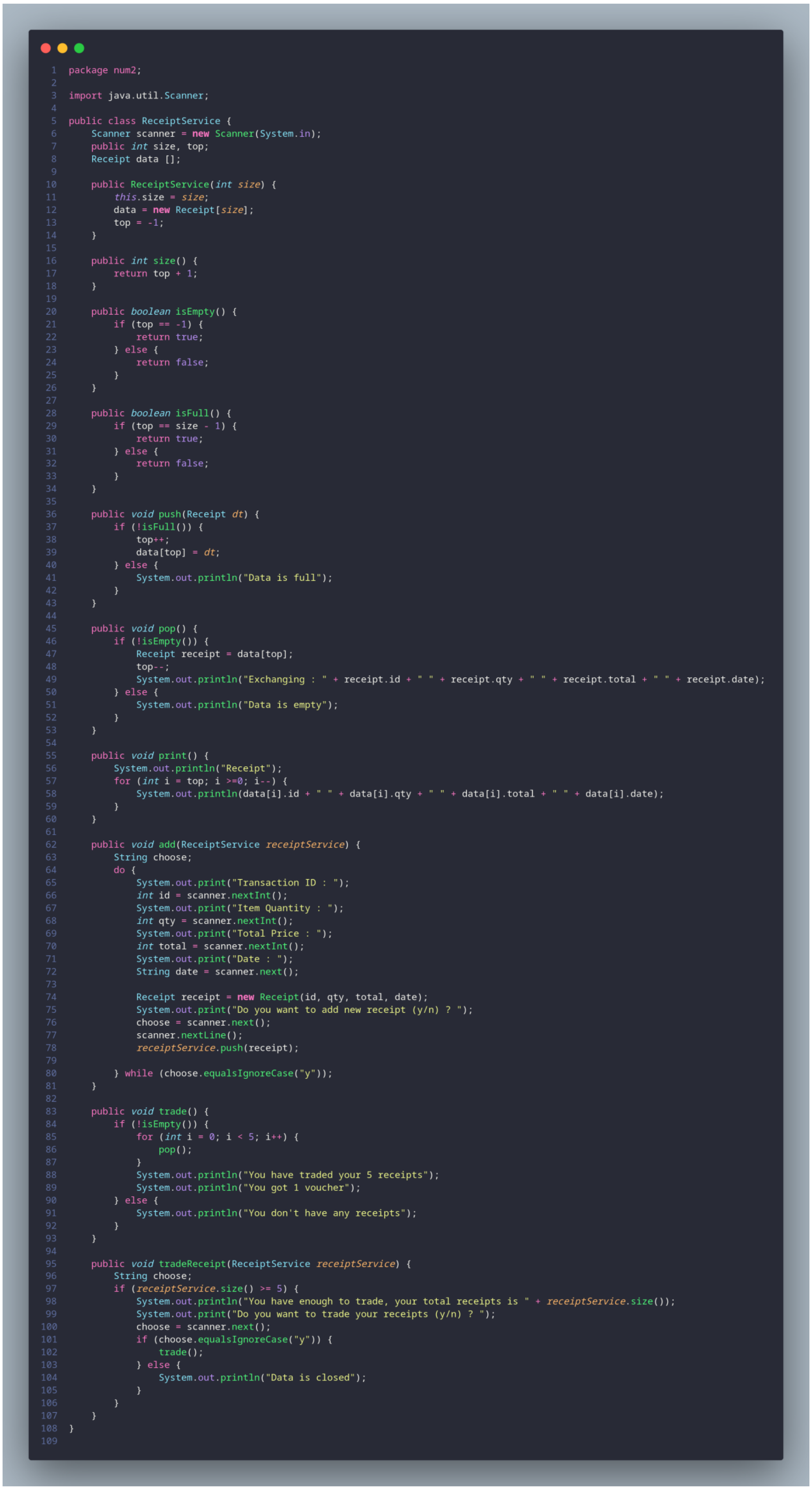
2. Every Sunday, Dewi shops to a supermarket that is in her residential area. Everytime she finishes, she keeps the receipt of what she has bought in a wardrobe. After 2 months, She had 8 receipts. She plans to trade her 5 receipts in exchange for a voucher.
   Create a program using stack implementation 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

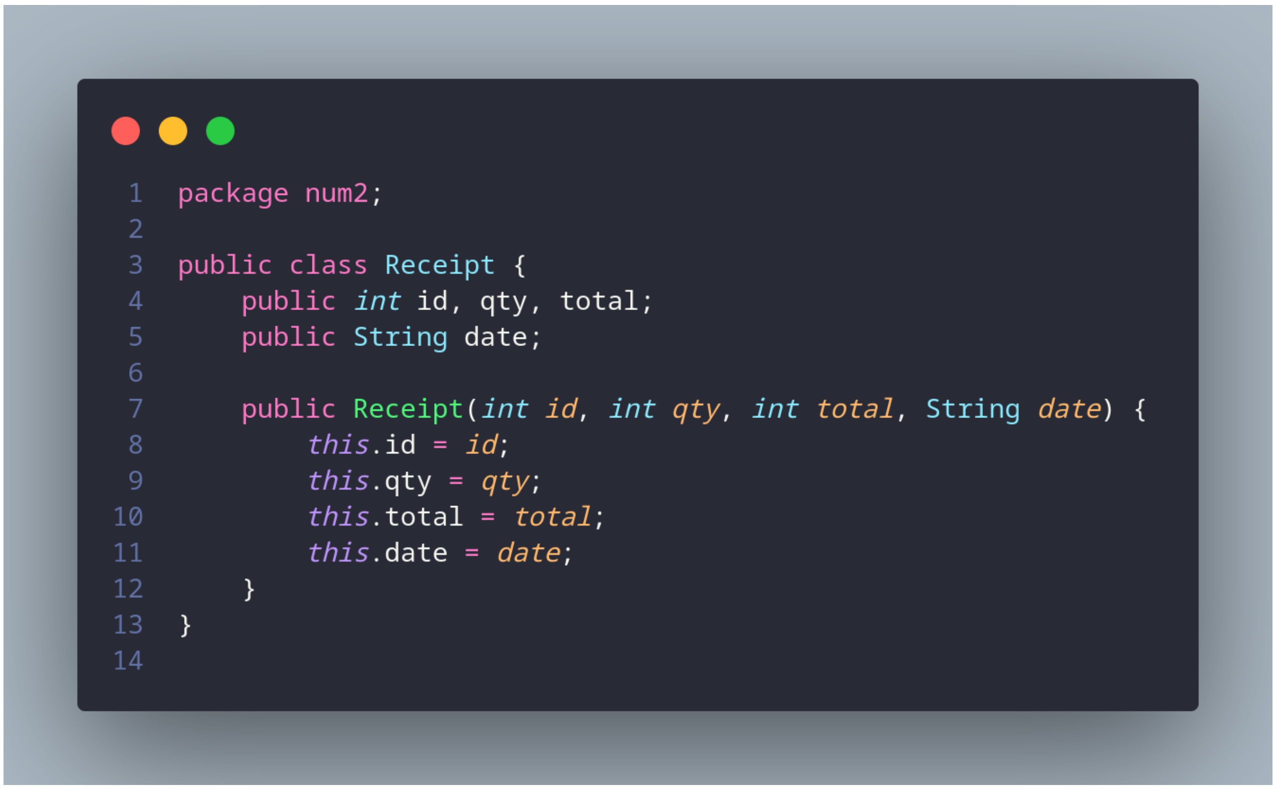
Code :



```java
package num2;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        ReceiptService receiptService = new ReceiptService(15);
        Receipt receipt1 = new Receipt(1, 1, 250000, "04-10-2023");
        Receipt receipt2 = new Receipt(2, 2, 150000, "04-11-2023");
        Receipt receipt3 = new Receipt(3, 3, 350000, "04-12-2023");
        Receipt receipt4 = new Receipt(4, 4, 450000, "04-13-2023");
        Receipt receipt5 = new Receipt(5, 5, 650000, "04-14-2023");
        Receipt receipt6 = new Receipt(6, 6, 550000, "04-15-2023");
        Receipt receipt7 = new Receipt(7, 7, 750000, "04-16-2023");
        Receipt receipt8 = new Receipt(8, 8, 850000, "04-17-2023");

        receiptService.push(receipt1);
        receiptService.push(receipt2);
        receiptService.push(receipt3);
        receiptService.push(receipt4);
        receiptService.push(receipt5);
        receiptService.push(receipt6);
        receiptService.push(receipt7);
        receiptService.push(receipt8);

        int choose;
        do {
            receiptService.print();
            System.out.println("Wardrobe Options");
            System.out.println("1. Add receipts");
            System.out.println("2. Trade your receipts");
            System.out.println("3. Exit");
            System.out.print("Choose : ");
            choose = scanner.nextInt();

            switch (choose) {
                case 1:
                    receiptService.add(receiptService);
                    break;
                case 2:
                    receiptService.tradeReceipt(receiptService);
                    break;
                case 3:
                    System.out.println("Wardrobe is closed");
                    break;
                default:
                    System.out.println("Invalid choice");
                    break;
            }
        } while (choose != 3);

        scanner.close();
    }
}
```

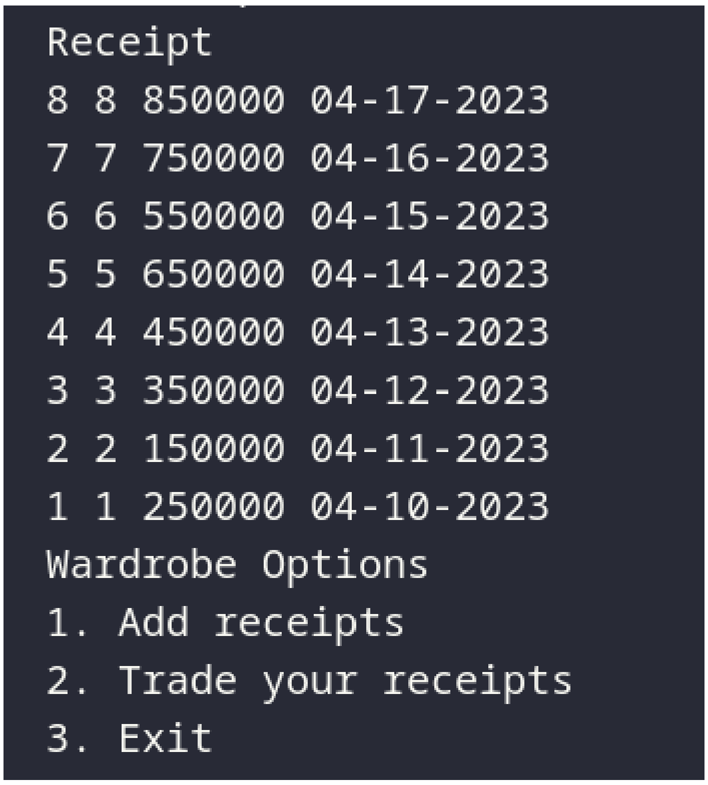


```java
package num2;

import java.util.Scanner;

public class ReceiptService {
    Scanner scanner = new Scanner(System.in);
    public int size, top;
    Receipt data [];

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

    public int size() {
        return 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(Receipt dt) {
        if (!isFull()) {
            top++;
            data[top] = dt;
        } else {
            System.out.println("Data is full");
        }
    }

    public void pop() {
        if (!isEmpty()) {
            Receipt receipt = data[top];
            top--;
            System.out.println("Exchanging : " + receipt.id + " " + receipt.qty + " " + receipt.total + " " + receipt.date);
        } else {
            System.out.println("Data is empty");
        }
    }

    public void print() {
        System.out.println("Receipt");
        for (int i = top; i >=0; i--) {
            System.out.println(data[i].id + " " + data[i].qty + " " + data[i].total + " " + data[i].date);
        }
    }

    public void add(ReceiptService receiptService) {
        String choose;
        do {
            System.out.print("Transaction ID : ");
            int id = scanner.nextInt();
            System.out.print("Item Quantity : ");
            int qty = scanner.nextInt();
            System.out.print("Total Price : ");
            int total = scanner.nextInt();
            System.out.print("Date : ");
            String date = scanner.next();

            Receipt receipt = new Receipt(id, qty, total, date);
            System.out.print("Do you want to add new receipt (y/n) ? ");
            choose = scanner.next();
            scanner.nextLine();
            receiptService.push(receipt);

        } while (choose.equalsIgnoreCase("y"));
    }

    public void trade() {
        if (!isEmpty()) {
            for (int i = 0; i < 5; i++) {
                pop();
            }
            System.out.println("You have traded your 5 receipts");
            System.out.println("You got 1 voucher");
        } else {
            System.out.println("You don't have any receipts");
        }
    }

    public void tradeReceipt(ReceiptService receiptService) {
        String choose;
        if (receiptService.size() >= 5) {
            System.out.println("You have enough to trade, your total receipts is " + receiptService.size());
            System.out.print("Do you want to trade your receipts (y/n) ? ");
            choose = scanner.next();
            if (choose.equalsIgnoreCase("y")) {
                trade();
            } else {
                System.out.println("Data is closed");
            }
        }
    }
}
```

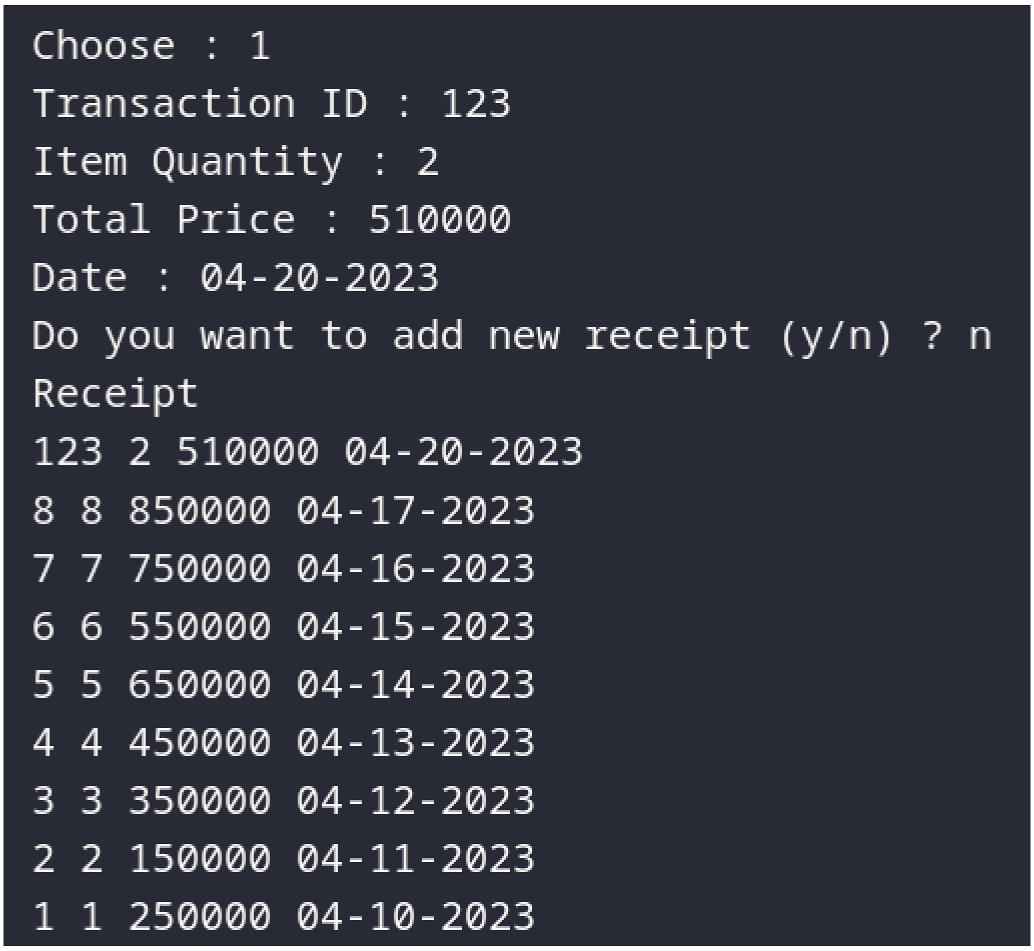


```java
package num2;

public class Receipt {
    public int id, qty, total;
    public String date;

    public Receipt(int id, int qty, int total, String date) {
        this.id = id;
        this.qty = qty;
        this.total = total;
        this.date = date;
    }
}
```

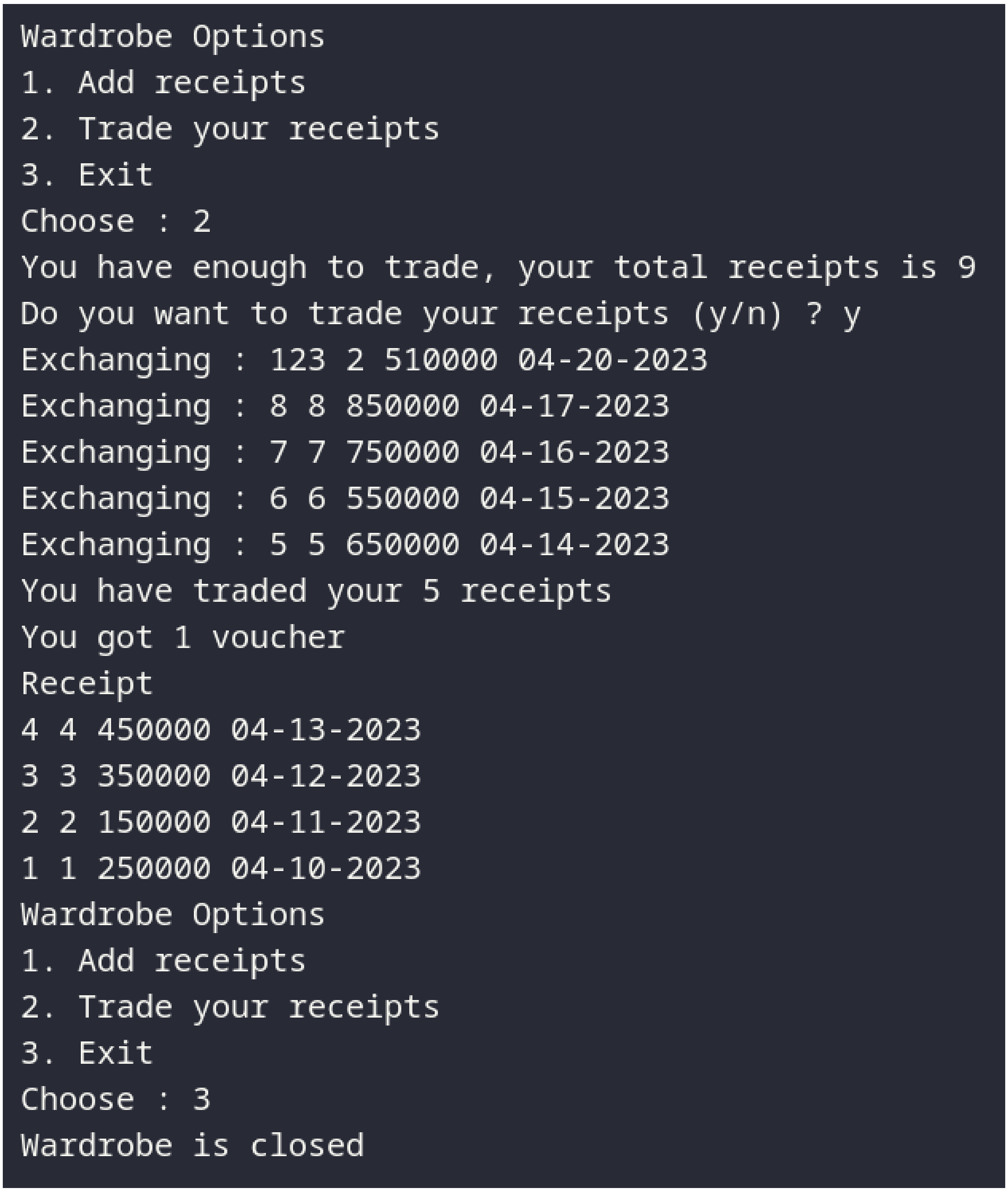
Result :



```
Receipt
8 8 850000 04-17-2023
7 7 750000 04-16-2023
6 6 550000 04-15-2023
5 5 650000 04-14-2023
4 4 450000 04-13-2023
3 3 350000 04-12-2023
2 2 150000 04-11-2023
1 1 250000 04-10-2023
Wardrobe Options
1. Add receipts
2. Trade your receipts
3. Exit
```



```
Choose : 1
Transaction ID : 123
Item Quantity : 2
Total Price : 510000
Date : 04-20-2023
Do you want to add new receipt (y/n) ? n
Receipt
123 2 510000 04-20-2023
8 8 850000 04-17-2023
7 7 750000 04-16-2023
6 6 550000 04-15-2023
5 5 650000 04-14-2023
4 4 450000 04-13-2023
3 3 350000 04-12-2023
2 2 150000 04-11-2023
1 1 250000 04-10-2023
```



```
Wardrobe Options
1. Add receipts
2. Trade your receipts
3. Exit
Choose : 2
You have enough to trade, your total receipts is 9
Do you want to trade your receipts (y/n) ? y
Exchanging : 123 2 510000 04-20-2023
Exchanging : 8 8 850000 04-17-2023
Exchanging : 7 7 750000 04-16-2023
Exchanging : 6 6 550000 04-15-2023
Exchanging : 5 5 650000 04-14-2023
You have traded your 5 receipts
You got 1 voucher
Receipt
4 4 450000 04-13-2023
3 3 350000 04-12-2023
2 2 150000 04-11-2023
1 1 250000 04-10-2023
Wardrobe Options
1. Add receipts
2. Trade your receipts
3. Exit
Choose : 3
Wardrobe is closed
```