JOBSHEET X
STACK

1. Learning Objectives

After completing this session, students will be able to::

1. Create a Stack data structure.
2. Implement Stack algorithms in a Java program.

2. Labs Activities

2.1 Experiment 1: Assignment Submission

Allocated time : 90 Minutes

A number of students submit their assignment files by stacking them on the lecturer's desk, applying the stack principle. The lecturer grades the assignments sequentially, starting from the top file. Refer to the following **Student** Class Diagram.

Student**<AttendanceNumber>**
nim: String name: String className: String grade: int
Student**<AttendanceNumber>**() Student**<AttendanceNumber>**(nim: String, name: String, className: String) grading(grade: int)

Next, to manage the submission of assignment files, a **StudentAssignmentStack** class is needed, serving as a Stack to store student assignment data. The attributes and methods within the **StudentAssignmentStack** class represent data processing using the Stack structure. Refer to the following **StudentAssignmentStack** Class Diagram.

StudentAssignmentStack <AttendanceNumber>
stack: Student[] size: int top: int
StudentAssignmentStack**< AttendanceNumber >**(size: int) isFull(): boolean isEmpty(): boolean push(mhs): void pop(): Student peek():Student print(): void

***Note:** The data type of the stack variable should be adjusted according to the type of data to be stored in the Stack. In this exercise, the data to be stored is an array of **Student** objects; therefore, the data type used is **Student**.*

2.1.1 Experiment Steps

A. Student Class

1. Create a new folder named, for example **Jobsheet10** inside your repository. Then, create a new file and name it **Student<AttendanceNumber>**.java.
2. Complete the **Student** class by adding the attributes shown in the **Student** class diagram, which include **name**, **nim**, **className**, and **grade**.
3. Add a parameterized constructor to the **Student** class as specified in the **Student** class diagram. Set a default value of **gade = -1** to indicate that the assignment has not yet been graded.

```
Student(String nim, String name, String className){
        this.nim = nim;
        this.name = name;
        this.className = className;
        this.grade = -1;
    }
```

4. Add a **grading()** method to the **Student** class, which is used to set the grade when a student's assignment is evaluated.

```
void grading(int grade){
        this.grade = grade;
    }
```

B. StudentAssignmentStack Class

5. After creating the **Student** class, you will then create the **StudentAssignmentStack** <**AttendanceNumber**>.java class to manage the stack of assignments. The **StudentAssignmentStack** class is an implementation of the Stack data structure.
6. Complete the **StudentAssignmentStack** class by adding the attributes shown in the **StudentAssignmentStack** class diagram, which include **stack**, **size**, and **top**.

```
Student[] stack;
int top, size;
```

7. Add a parameterized constructor to the **StudentAssignmentStack** class to initialize the maximum capacity of student assignment data that can be stored in the Stack, as well as to set the initial index of the **top** pointer.

```
StudentAssignmentStack(int size){
        this.size = size;
        top = -1;
        stack = new Student[size];
    }
```

8. Next, create a boolean **isFull** method to check whether the stack of student assignments has reached its maximum capacity.

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

9. Then, create another boolean **isEmpty** method to check whether the stack of assignments is empty.

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

10. To add assignment files to the Stack, create a **push** method. This method should accept a parameter **std**, which is an object of the **Student** class.

```
void push(Student std){
        if(!isFull()){
            top++;
            stack[top] = std;
        }else{
            System.out.println("Stack is already full!!");
        }
    }
```

11. The grading of student assignments by the lecturer is done using the **pop** method to remove the assignment to be graded. This method does not accept any parameters but returns an object of the **Student** class.

```
Student pop(){
        if(!isEmpty()){
            Student std = stack[top];
            top--;
            return std;
        }else{
            System.out.println("There is no data in Stack!!");
            return null;
        }
    }
```

***Note**: If information about the student data is required, the return type should be an object of the **Student** class. Conversely, a **void** return type can be used if the student data that is removed will not be processed or used again.*

12. Create a **peek** method to check the student assignment at the **top** of the stack.

```
Student peek(){
        if(!isEmpty()){
            return stack[top];
        }else{
            System.out.println("There is no data in Stack!!");
            return null;
        }
    }
```

13. Add a **print** method to display the entire list of student assignments in the Stack.

```
void print(){
        for(int i=0;i<=top;i++){
            System.out.println(stack[i].nim+"\t"+stack[i].name+"\t"
            +stack[i].className);
        }
        System.out.println("");
    }
```

C. Main Class

14. Create a new main class named **StudentDemo<AttendanceNumber>.java**
15. Add a new **main()** method
16. Within main() method, instantiate a new object from **StudentAssignmentStack** class. Give the object name as **stack** and pass **5** as the parameter.

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

17. Declare a Scanner object with the variable name **scan** and an int variable named **choice**.

18. Add a menu to allow users to choose Stack operations for managing student assignment data, using a **do-while** loop structure.

```java
do {
    System.out.println("\nMenu:");
    System.out.println("1. Mengumpulkan Tugas");
    System.out.println("2. Menilai Tugas");
    System.out.println("3. Melihat Tugas Teratas");
    System.out.println("4. Melihat Daftar Tugas");
    System.out.print("Pilih: ");
    pilih = scan.nextInt();
    scan.nextLine();
    switch (pilih) {
        case 1:
            System.out.print("Nama: ");
            String nama = scan.nextLine();
            System.out.print("NIM: ");
            String nim = scan.nextLine();
            System.out.print("Kelas: ");
            String kelas = scan.nextLine();
            Mahasiswa mhs = new Mahasiswa(nama, nim, kelas);
            stack.push(mhs);
            System.out.printf("Tugas %s berhasil dikumpulkan\n", mhs.nama);
            break;
        case 2:
            Mahasiswa dinilai = stack.pop();
            if (dinilai != null) {
                System.out.println("Menilai tugas dari " + dinilai.nama);
                System.out.print(s:"Masukkan nilai (0-100): ");
                int nilai = scan.nextInt();
                dinilai.tugasDinilai(nilai);
                System.out.printf(format:"Nilai Tugas %s adalah %d\n", dinilai.nama, nilai);
            }
            break;
        case 3:
            Mahasiswa lihat = stack.peek();
            if (lihat != null) {
                System.out.println("Tugas terakhir dikumpulkan oleh " + lihat.nama);
            }
            break;
        case 4:
            System.out.println(x:"Daftar semua tugas");
            System.out.println(x:"Nama\tNIM\tKelas");
            stack.print();
            break;
        default:
            System.out.println(x:"Pilihan tidak valid.");
    }
} while (pilih >= 1 && pilih <= 4);
```

19. **Commit and push your code to your Github repository**
20. Compile and run your program.

2.1.2 Verification

Compare the output of your compiled program with the following output.

```
Menu:
```

```
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 1
Name: Bima
NIM: 1001
Class Name: 1I
Bima's assignment has been successfully submitted!!

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 1
Name: Fidela
NIM: 1002
Class Name: 1I
Fidela's assignment has been successfully submitted!!

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 1
Name: Resty
NIM: 1003
Class Name: 1I
Resty's assignment has been successfully submitted!!

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 3
The last assignmentcomes from Resty

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 1
Name: Joseph
NIM: 1004
Class Name: 1I
Joseph's assignment has been successfully submitted!!

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 4
Assignment list:
Name    NIM     Class Name
1001    Bima    1I
1002    Fidela  1I
1003    Resty   1I
1004    Joseph  1I

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
```

```
4. View All Assignments
Choose a menu: 2
Grading assignment from Joseph
Input grade (0-100): 85
Assignment grade of Joseph is 85

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 4
Assignment list:
Name    NIM     Class Name
1001    Bima    1I
1002    Fidela  1I
1003    Resty   1I

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu:
```

2.1.3 Questions

1. Explain the role of the stack data structure in the student assignment management system. Why was a stack used instead of another data structure (e.g., queue or list)?
2. What is the difference between the **push()** and **pop()** methods in a stack, and how are they used in this program?
3. Why is it important to check the condition **!isFull()** before calling the **push()** method? What could go wrong if this check is removed?
4. How many student assignments can be stored in the current implementation of the stack? Provide the specific line of code or variable that determines this.
5. Modify the existing stack implementation so that the system can also show **the first student who submitted their assignment**. Describe the changes you made in both **StudentDemo** and **StudentAssignmentStack** classes.
6. Implement a method to count and return the number of assignments currently stored in the stack. Describe how your method works.
7. What did you learn from this experiment about stack-based systems? Reflect on a real-world application where this kind of system might be useful.
8. Don't forget to synchronize the changes you made from this questions into your repository!

2.2 Experiment 2: Convert Assignment Grade to Binary

Time Allocated: 60 Minutes

In Experiment 2, a new method is added that functions to convert the task value of type **int** into **binary** form after the task has been assigned a value and removed from the Stack.

2.2.1 Experiment Steps

1. Re-open **StudentAsignmentStack<AttendanceNumber>.java**
2. Create a new method named **convertToBinary()** with an **int** parameter **grade**

```
String convertToBinary(int grade){
        ConversionStack stack = new ConversionStack();
        while (grade > 0) {
            int mod = grade % 2;
            stack.push(mod);
            grade = grade / 2;
        }
        String binary = "";
        while (!stack.isEmpty()) {
            binary += stack.pop();
        }
        return binary;
    }
```

In this method, the use of **ConversionStack** is implemented, which is similar to the **StudentAssignmentStack** class. The purpose of this is to differentiate the Stack used for students from the Stack used for binary data, as the data types involved are different. Therefore, create a new file named **ConversionStack<AttendanceNumber>.java**.

Note: *It is important to remember that all Stack classes essentially have the same operations (methods). What distinguishes them is the specific activities that need to be performed, such as after adding or removing data.*

```
public class ConversionStack {
    int[] binaryStack;
    int size;
    int top;

    public ConversionStack() {
        this.size = 32; //asumsi 32 bit
        binaryStack = new int[size];
        top = -1;
    }
    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == size - 1;
    }
    public void push(int data) {
        if (isFull()) {
            System.out.println("Stack is already full-filled!!");
        } else {
            top++;
            binaryStack[top] = data;
        }
    }

    public int pop() {
        if (isEmpty()) {
            System.out.println("Stack is still empty!!");
            return -1;
        } else {
            int data = binaryStack[top];
            top--;
            return data;
        }
    }
}
```

3. To convert the student's assignment value into **binary** form after grading, add a line of code in StudentDemo class, specifically within case 2.

```
case 2:
      Student graded = stack.pop();
     if (graded != null) {
       System.out.println("Grading assignment from " + graded.name);
       System.out.print("Input grade (0-100): ");
       int grade = scan.nextInt();
       graded.grading(grade);
       System.out.printf("Assignment grade of %s is %d\n", graded.name,
grade);
       String binary = stack.convertToBinary(grade);
       System.out.printf("Assignment grade in binary is %s\n", binary);
     }
     break;
```

4. Compile and run the program.
5. **Commit and push the code to your Github repository.**

2.2.2 Verification

Compare the output of your compiled program with the following output.

```
Menu:
1. Submit Assignment
2. Grade Assignment
```

```
3. View Top Assignment
4. View All Assignments
Choose a menu: 1
Name: Excel
NIM: 1001
Class Name: 1I
Excel's assignment has been successfully submitted!!

Menu:
1. Submit Assignment
2. Grade Assignment
3. View Top Assignment
4. View All Assignments
Choose a menu: 2
Grading assignment from Excel
Input grade (0-100): 87
Assignment grade of Excel is 87
Assignment grade in binary is 1010111
```

2.2.3 Questions

1. Explain the workflow of the **convertToBinary()** method.
2. In the **convertToBinary()** method, change the loop condition to **while (grade != 0)**. What is the result? Explain the reason!

2.4 Assignment

Allocated Time : 90 Minutes

Students submit an excuse letter (due to illness or other reasons) each time they are absent from a lecture. The most recent letter submitted will be processed or validated first by the department administrator. Refer to the class diagram below.

ExcuseLetter<AttendanceNumber>
id: String name: String className: String typeOfExcuse: char duration: int
ExcuseLetter<AttendanceNumber>() ExcuseLetter<AttendanceNumber>(id: String, name: String, className: String, type: char, duration: int)

The **typeOfExcuse** attribute is used to store the type of student leave (S: Sick, or I: Other personal reasons), while the **duration** attribute stores the length of the leave period.

Based on the class diagram, implement a class named **ExcuseLetter**, and add a **ExcuseLetterStack** class to manage the collection of **ExcuseLetter** objects. In the class containing the **main** method, provide the following menu options:

1. **Submit Excuse Letter** – to input new excuse letter data
2. **Process Excuse Letter** – to process or verify the latest excuse letter

3. **View Latest Excuse Letter** – to display the top letter in the stack
4. **Search for Letter** – to search for the presence of a letter by the **student's name**