1. Stack
   a) Stack menggunakan system LIFO(Last In First Out) dimana data yang terakhir di masukan adalah data yang hanya bisa dilihat, jika ingin melihat data disebelumnya, harus pop data yang paling atas tersebut (seperti menghapus data yang teratas).
   b) Sebuah contoh di dunia nyata bisa seperti di browser, dimana situs yang Anda akses adalah yang Anda hanya bisa lihat, dan jika ingin liat situs sebelumnya, harus backwards, seperti stack, harus satu-satu untuk lakukan backwards itu.
   c) Berikut adalah contoh programnya.

```
import java.util.Scanner;
import java.util.Stack;

public class Main {
    static void printMenu(){
        System.out.println("1. Input");
        System.out.println("2. Access Previous Data");
        System.out.println("3. Exit");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Stack<String> historyList = new Stack<>();
        int choice = -1 ;
        int i = 0;
        while(choice!=3){
            do{
                try{
                    printMenu();
                    System.out.print("Your Choice: ");
                    choice = scanner.nextInt();
                } catch (Exception e){
                    System.out.println("Invalid Choice!");
                    choice = -1;
                } finally {
                    scanner.nextLine();
                }
            } while ( choice < 1 || choice > 3 );

            if(choice==1){
                System.out.println("Input below:");
                String input = scanner.nextLine();
                historyList.push(input);
                System.out.println("Top             = " + historyList.peek());
                System.out.println("Stack size = " + historyList.size());
            } else if (choice==2) {
                if(historyList.isEmpty()){
                    System.out.println("Stack is empty!");
                } else {
                    historyList.pop();
                    if(historyList.isEmpty()){
                        System.out.println("Stack is empty!");
                    } else {
                        System.out.println("Top             = " + historyList.peek());
                        System.out.println("Stack size = " + historyList.size());
                    }
                }
            }
        }
    }
}
```

2. Queue

a) Queue menggunakan system FIFO(First In First Out) dimana data yang pertama di input adalah data yang akan keluar pertama. Dimana data-data yang lain tidak akan bisa di push jika ada yang perlu di selip di antara 2 data.

b) Sebuah contoh di dunia nyata bisa seperti antrian di kasir, yang pertama mengantri, akan pertama yang keluar dari antrian tersebut dan tidak ada siapa pun yang memotong antrian itu.

c) Berikut adalah contoh programnya.

```
import java.util.PriorityQueue;
import java.util.Queue;
import java.util.Scanner;

public class Main {
    static void printMenu(){
        System.out.println("1. Input Queue");
        System.out.println("2. Remove Head Queue");
        System.out.println("3. Exit");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Queue<Integer> queueList = new PriorityQueue<>();
        int choice = -1 ;
        int i = 0;
        while(choice!=3){
            do{
                try{
                    printMenu();
                    System.out.print("Your Choice: ");
                    choice = scanner.nextInt();
                } catch (Exception e){
                    System.out.println("Invalid Choice!");
                    choice = -1;
                } finally {
                    scanner.nextLine();
                }
            } while ( choice < 1 || choice > 3 );

            if(choice==1){
                ++i;
                queueList.offer(i);
                System.out.println("Current      = " + queueList.peek());
                System.out.println("Last Queue = " + i);
                System.out.println("Queue Size = " + queueList.size());
            } else if (choice==2) {
                if(queueList.isEmpty()){
                    System.out.println("There is no queue!");
                } else {
                    queueList.poll();
                    if(queueList.isEmpty()){
                        System.out.println("That is the last queue!");
                    } else {
                        System.out.println("Current      = " + queueList.peek());
                        System.out.println("Last Queue = " + i);
                        System.out.println("Queue size = " + queueList.size());
                    }
                }
            }
        }
    }
}
```

3. Priority Queue
   a) Priority Queue menggunakan system FIFO(First In First Out) seperti Queue dimana data yang pertama di input adalah data yang akan keluar pertama. Tetapi, jika ada data yang lebih penting, bisa di letak di depan queue.
   b) Sebuah contoh di dunia nyata bisa seperti antrian di rumah sakit, yang pertama mengantri, akan pertama yang keluar dari antrian tersebut, akan tetapi ada tingkat severitas, jika penyakitnya lebih kritis, maka akan diletak di depan semua antrian yang lain.
   c) Berikut adalah contoh programnya.

```
import java.util.PriorityQueue;
import java.util.Queue;
import java.util.Scanner;

public class Main {
    static void printMenu(){
        System.out.println("1. Input Queue");
        System.out.println("2. Remove Head Queue");
        System.out.println("3. Exit");
    }
    static void printPriorityMenu(){
        System.out.println("1. Priority");
        System.out.println("2. Standard");
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Queue<Integer> queueList = new PriorityQueue<>();
        int choice = -1 ;
        int which = -1;
        int i = 0;
        while(choice!=3){
            do{
                try{
                    printMenu();
                    System.out.print("Your Choice: ");
                    choice = scanner.nextInt();
                } catch (Exception e){
                    System.out.println("Invalid Choice!");
                    choice = -1;
                } finally {
                    scanner.nextLine();
                }
            } while ( choice < 1 || choice > 3 );

            if(choice==1){
                while(true) {
                    do {
                        try {
                            printPriorityMenu();
                            System.out.print("Your Choice: ");
                            which = scanner.nextInt();
                        } catch (Exception e) {
                            System.out.println("Invalid Choice!");
                            which = -1;
                        } finally {
                            scanner.nextLine();
                        }
                    } while (which < 1 || which > 2);
                    break;
                }
                if(which == 1){
```

```
                    int priorityNo = 1;
                    queueList.offer(priorityNo);
                    System.out.println("Current       = " + queueList.peek());
                    System.out.println("Queue          = " + queueList);
                } else {
                    ++i;
                    queueList.offer(i);
                    System.out.println("Current       = " + queueList.peek());
                    System.out.println("Queue          = " + queueList);
                }
            } else if (choice==2) {
                if(queueList.isEmpty()){
                    System.out.println("There is no queue!");
                } else {
                    queueList.poll();
                    if(queueList.isEmpty()){
                        System.out.println("That is the last queue!");
                    } else {
                        System.out.println("Current       = " + queueList.peek());
                        System.out.println("Queue          = " + queueList);
                    }
                }
            }
        }
    }
}
```