Assignment 3:

1) **Implement Stack Data Structure.**

Source Code:

```java
import java.util.Scanner;

public class Stack {
    int arr[];
    int top;
    int capacity;

    Stack(int cap) {
        arr = new int[cap];
        capacity = cap;
        top = -1;
    }

    public void push(int x) {
        if (top == capacity - 1)
            System.out.println("Overflow\nProgram Terminated\n");
        else {
            top += 1;
            arr[top] = x;
        }
    }

    public int pop() {
        int x = -1;
        if (top == -1)
            System.out.println("Underflow\nProgram Terminated\n");
        else {
            x = arr[top];
            top--;
        }
        return x;
    }

    public int peek() {
        if (top != -1) {
            return arr[top];
        }
        return -1;
    }

    public int size() {
        return top + 1;
    }

    public static void main(String[] args) {
        Stack stack = new Stack(4);

        System.out.println("inserting 1 in the stack");
        stack.push(1);
        System.out.println("inserting 2 in the stack");
        stack.push(2);

        System.out.println("removing the top element (2)");
        stack.pop();
        System.out.println("removing the top element (1)");
        stack.pop();

        System.out.println("inserting 3 in the stack");
        stack.push(3);

        System.out.println("The top element is " + stack.peek());
        System.out.println("The stack size is " + stack.size());

        System.out.println("removing the top element (3)");
        stack.pop();
    }
}
```

Output:

```
inserting 1 in the stack
inserting 2 in the stack
removing the top element (2)
removing the top element (1)
inserting 3 in the stack
The top element is 3
The stack size is 1
removing the top element (3)
```

2) Implement Queue Data Structure.

Source Code:

```java
public class Queue {
    private int arr[];
    private int capacity;
    private int front;
    private int rear;

    Queue(int capacity) {
        this.capacity = capacity;
        arr = new int[capacity];
        front = rear = -1;
    }

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

```java
public boolean isFull() {
    return (rear == capacity - 1);
}

public void push_elements(int data) {
    if (isFull()) {
        System.out.println("Overflow");
        return;
    }
    arr[++rear] = data;
}

public int pop_elements() {
    if (isEmpty()) {
        System.out.println("Queue is Empty");
        return -1;
    }
    int front = arr[0];
    for (int i = 0; i < rear; i++)
        arr[i] = arr[i + 1];

    rear--;
    return front;
}

public int peek() {
    if (isEmpty()) {
        System.out.println("Queue is Empty");
        return -1;
    }
    return arr[0];
}

public static void main(String[] args) {
    Queue q = new Queue(5);
    System.out.println("inserting 1 in the queue");
    q.push_elements(1);
    System.out.println("inserting 2 in the queue");
    q.push_elements(2);

    System.out.println("removing the element (1) from front");
    q.pop_elements();
    System.out.println("removing the element (2) from front");
    q.pop_elements();

    System.out.println("inserting 3 in the queue");
    q.push_elements(3);

    System.out.println("inserting 4 in the queue");
    q.push_elements(4);

    System.out.println("The element in front is " + q.peek());

    System.out.println("removing the element (3) from front");
```

```
        q.pop_elements();
    }

}
```

Output:

```
inserting 1 in the queue
inserting 2 in the queue
removing the element (1) from front
removing the element (2) from front
inserting 3 in the queue
inserting 4 in the queue
The element in front is 3
removing the element (3) from front
```