**Basics of Java Practice Exercise – Day 8**

Submitted by: Aniket Singh (Emp no. – 2605511)

**1. Write a Program to implement Stack using Array.**

**Output:**

**After Pushing 4 Elements: Hello world java Programming**

**After a Pop: Hello world java**

**Code –**

**package** practicsday8;

**public** **class** StackusingArray {

**int** maxSize;

String[] stackArray;

**int** top;

**public** StackusingArray(**int** size) {

maxSize=size;

stackArray=**new** String[maxSize];

top=-1;

}

**void** push(String value) {

**if**(top==maxSize-1) {

System.***out***.println("Stack Overflow");

}

**else** {

top++;

stackArray[top]=value;

}

}

**public** String pop() {

**if**(top==-1) {

System.***out***.println("Stack Underflow");

**return** **null**;

}

**else** {

System.***out***.println("\nPopped Element: "+stackArray[top--]+"\n");

**return** **null**;

}

}

**void** display() {

**if**(top==-1) {

System.***out***.println("Stack is empty");

}

**else** {

**for**(**int** i=0;i<top+1;i++) {

System.***out***.println(stackArray[i]);

}

}

}

**public** **static** **void** main(String[] args) {

StackusingArray stack=**new** StackusingArray(10);

stack.push("Hello");

stack.push("world");

stack.push("java");

stack.push("Programming");

System.***out***.println("Original Stack: ");

stack.display();

stack.pop();

stack.display();

}

}

**2. Write a Program to implement Stack using Linked List.**

**Input:**

**10.0 20.0 30.0 40.0**

**Output:**

**The elements of the stack are: 40.0 30.0 20.0 10.0 null**

**After popping twice: 20.0 10.0 null**

**Code –**

**package** practicsday8;

**class** node{

**double** data;

node next;

node (**double** data){

**this**.data=data;

**this**.next=**null**;

}

}

**class** Stack{

node top;

Stack(){

**this**.top=**null**;

}

**void** push(**double** data) {

node newNode=**new** node(data);

newNode.next=top;

top=newNode;

}

**public** **double** pop() {

**if**(top==**null**) {

System.***out***.println("Stack Underflow");

**return** -1;

}

**else** {

**double** value=top.data;

top=top.next;

**return** value;

}

}

**void** display() {

node current=top;

**while**(current!=**null**) {

System.***out***.println(current.data);

current=current.next;

}

}

}

**public** **class** StackusingLinkedList {

**public** **static** **void** main(String[] args) {

Stack stack=**new** Stack();

stack.push(10.0);

stack.push(20.0);

stack.push(30.0);

stack.push(40.0);

System.***out***.println("Original Stack: ");

stack.display();

System.***out***.println("\nPopped Element: "+stack.pop());

System.***out***.println();

stack.display();

System.***out***.println("\nPopped Element: "+stack.pop());

System.***out***.println();

stack.display();

}

}

**3. Write a Program to Reverse a String using Stack.**

**Input:**

**JavaQuiz**

**Output:**

**ziuQavaJ**

**Code –**

**package** practicsday8;

**import** java.util.Stack;

**import** java.util.\*;

**public** **class** ReverseStringStack {

**public** **static** **void** main(String[] args) {

Stack<String> revstr=**new** Stack<String>();

Scanner sc=**new** Scanner(System.***in***);

String str;

System.***out***.println("Enter String: ");

str=sc.next();

**char**[] charArray=str.toCharArray();

**for**(**int** i=0;i<str.length();i++) {

revstr.push(Character.*toString*(charArray[i]));

}

System.***out***.println("\nReversed String: ");

**for**(**int** i=0;i<str.length();i++) {

System.***out***.print(revstr.pop()) ;

}

}

}

**4. Write a Program to evaluate an Expression using Stacks.**

**Input:**

**10 + 2 \* 6**

**Output:**

**22**

**Code –**

**package** practicsday8;

**import** java.util.Scanner;

**import** java.util.Stack;

**public** **class** EvaluateExp {

**public** **static** **int** evaluate(String expression) {

**char**[] tokens=expression.toCharArray();

Stack<Integer> values=**new** Stack<>();

Stack<Character> operators=**new** Stack<>();

**for**(**int** i=0;i<tokens.length;i++) {

**if**(tokens[i]==' ') {

**continue**;

}

**if**(tokens[i]>='0' && tokens[i]<='9') {

StringBuilder sbuf=**new** StringBuilder();

**while**(i<tokens.length && tokens[i]>='0' && tokens[i]<='9') {

sbuf.append(tokens[i++]);

}

values.push(Integer.*parseInt*(sbuf.toString()));

i--;

}

**else** **if**(tokens[i]=='+' || tokens[i]=='-' || tokens[i]=='\*' || tokens[i]=='/'){

**while**(!operators.isEmpty() && *hasPrecedence*(tokens[i],operators.peek())) {

values.push(*applyOp*(operators.pop(),values.pop(),values.pop()));

}

operators.push(tokens[i]);

}

}

**while**(!operators.isEmpty()) {

values.push(*applyOp*(operators.pop(),values.pop(),values.pop()));

}

**return** values.pop();

}

**public** **static** **boolean** hasPrecedence(**char** op1,**char** op2) {

**if**(op2=='(' || op2 ==')') {

**return** **false**;

}

**if**((op1=='\*' || op1=='/')&& (op2=='+' || op2=='-')) {

**return** **false**;

}

**else** {

**return** **true**;

}

}

**public** **static** **int** applyOp(**char** op,**int** b,**int** a) {

**switch**(op) {

**case** '+':

**return** a+b;

**case** '-':

**return** a-b;

**case** '\*':

**return** a\*b;

**case** '/':

**if**(b==0) {

**throw** **new** UnsupportedOperationException("cannot divide by zero");

}

**return** a/b;

}

**return** 0;

}

**public** **static** **void** main(String[] args) {

System.***out***.println("Enter Expression: ");

Scanner sc=**new** Scanner(System.***in***);

String exp=sc.nextLine();

System.***out***.println(EvaluateExp.*evaluate*(exp));

}

}

**5. Write a program to reverse a stack using recursion, without using any loop.**

**Input:**

**1 2 3 4**

**Output:**

**4 3 2 1**

**Code –**

**package** practicsday8;

**import** java.util.Stack;

**public** **class** ReverseStack {

**public** **static** **void** reverseStack(Stack<Integer> stack) {

**if**(!stack.isEmpty()) {

**int** bottom=*popBottom*(stack);

*reverseStack*(stack);

stack.push(bottom);

}

}

**public** **static** **int** popBottom(Stack<Integer> stack) {

**int** top=stack.pop();

**if**(stack.isEmpty()) {

**return** top;

}

**else** {

**int** bottom=*popBottom*(stack);

stack.push(top);

**return** bottom;

}

}

**public** **static** **void** main(String[] args) {

Stack<Integer> stack=**new** Stack<>();

stack.push(1);

stack.push(2);

stack.push(3);

stack.push(4);

stack.push(5);

System.***out***.println("Original stack: "+stack);

System.***out***.println();

*reverseStack*(stack);

System.***out***.println("Reversed Stack; "+stack);

}

}

**6. Write a program to find the minimum element in a stack.**

**Input:**

**16 15 29 19 18**

**Output:**

**15**

**Code –**

**package** practicsday8;

**import** java.util.Scanner;

**import** java.util.Stack;

**public** **class** MinStack {

**public** **static** **void** main(String[] args) {

Stack<Integer> stack=**new** Stack<>();

System.***out***.println("Enter Number of elements is stack: ");

Scanner sc=**new** Scanner(System.***in***);

**int** num=sc.nextInt();

**for**(**int** i=0;i<num;i++) {

System.***out***.println("Enter Element value:");

**int** val=sc.nextInt();

stack.push(val);

}

**int** temp=stack.peek();

**for**(**int** i=0;i<num;i++) {

**if**(temp>stack.peek()) {

temp=stack.peek();

}

stack.pop();

}

System.***out***.println("Min: "+temp);

}

}

**7. Given a stack of integers, find whether the top element of the stack is an even number or not. Return true if**

**the top element is an even number, else return false.**

**Input:**

**40 30 25 15**

**Output:**

**True**

**Code –**

**package** practicsday8;

**import** java.util.Scanner;

**import** java.util.Stack;

**public** **class** EvenStack {

**public** **static** **void** main(String[] args) {

Stack<Integer> stack=**new** Stack<>();

System.***out***.println("Enter Number of elements is stack: ");

Scanner sc=**new** Scanner(System.***in***);

**int** num=sc.nextInt();

**for**(**int** i=0;i<num;i++) {

System.***out***.println("Enter Element value:");

**int** val=sc.nextInt();

stack.push(val);

}

**if**(stack.peek()%2==0) {

System.***out***.println("true");

}

**else** {

System.***out***.println("false");

}

}

}

**8. Write a Program to implement Queue using Array.**

**Output:**

**Elements in queue: 10 20 30 40**

**After removing first element: 20 30 40**

**Code –**

**package** practicsday8;

**public** **class** QueueusingArray {

**int** front;

**int** rear;

**int** capacity;

**int**[] queue;

**public** QueueusingArray(**int** size) {

capacity=size;

queue=**new** **int**[capacity];

front=-1;

rear=-1;

}

**public** **boolean** isEmpty() {

**return** front==-1;

}

**public** **boolean** isFull() {

**return** (rear+1)%capacity==front;

}

**void** enqueue(**int** item) {

**if**(isFull()) {

System.***out***.println("Queue is Full");

}

**if**(isEmpty()) {

front=0;

}

rear=(rear+1)%capacity;

queue[rear]=item;

}

**public** **int** dequeue() {

**if**(isEmpty()) {

System.***out***.println("Queue is Empty");

**return** -1;

}

**int** item=queue[front];

**if**(front==rear) {

front=rear=-1;

}

**else** {

front=(front+1)%capacity;

}

**return** item;

}

**void** display() {

**if**(isEmpty()) {

System.***out***.println("Queue is Empty");

**return**;

}

System.***out***.println("Queue: ");

**int** i=front;

**while**(**true**) {

System.***out***.println(queue[i]+" ");

**if**(i==rear) {

**break**;

}

i=(i+1)%capacity;

}

System.***out***.println();

}

**public** **static** **void** main(String[] args) {

QueueusingArray queueArray=**new** QueueusingArray(4);

queueArray.enqueue(10);

queueArray.enqueue(20);

queueArray.enqueue(30);

queueArray.enqueue(40);

System.***out***.println("Original Queue: ");

queueArray.display();

System.***out***.println();

queueArray.dequeue();

System.***out***.println("Queue after removing first element: ");

queueArray.display();

}

}

**9. Write a Program to implement Queue using Linked List.**

**Output:**

**Elements in queue: 89 99 109 209 309**

**After removing two elements: 109 209 309**

**Code –**

**package** practicsday8;

**class** Node{

**int** data;

Node next;

**public** Node(**int** data) {

**this**.data=data;

**this**.next=**null**;

}

}

**class** Queue{

Node front,rear;

**int** size;

**public** Queue() {

**this**.front=**this**.rear=**null**;

**this**.size=0;

}

**void** enqueue(**int** item) {

Node newNode=**new** Node(item);

**if**(**this**.rear==**null**) {

**this**.front=**this**.rear=newNode;

**return**;

}

**this**.rear.next=newNode;

**this**.rear=newNode;

size++;

}

**int** dequeue() {

**if**(**this**.front==**null**) {

**return** Integer.***MIN\_VALUE***;

}

Node temp=**this**.front;

**this**.front=**this**.front.next;

**if**(**this**.front==**null**) {

**this**.rear=**null**;

}

size--;

**return** temp.data;

}

**void** display() {

**if**(**this**.front==**null**) {

System.***out***.println("Queue is Empty");

**return**;

}

Node temp=**this**.front;

**while**(temp!=**null**) {

System.***out***.print(temp.data+" ");

temp=temp.next;

}

System.***out***.println();

}

**public** **int** getSize() {

**return** **this**.size;

}

}

**public** **class** QueueusingLinkedList {

**public** **static** **void** main(String[] args) {

Queue queue=**new** Queue();

queue.enqueue(56);

queue.enqueue(2);

queue.enqueue(99);

queue.enqueue(45);

queue.enqueue(12);

queue.enqueue(46);

System.***out***.println("Original Queue: ");

queue.display();

System.***out***.println();

System.***out***.println("Popped Element: "+queue.dequeue());

System.***out***.println();

System.***out***.println("Queue now: ");

queue.display();

}

}

**10. Write a Program to Implement Circular Queue using Array.**

**Output:**

**Elements in circular queue: 14 13 22 -8**

**After removing first element: 13 22 -8**

**Code –**

**package** practiceday8;

**public** **class** CircularQueue{

**int**[] queue;

**int** front,rear,size,capacity;

**public** CircularQueue(**int** capacity) {

**this**.capacity=capacity;

**this**.queue=**new** **int**[capacity];

**this**.front=**this**.size=0;

**this**.rear=capacity-1;

}

**private** **boolean** isFull() {

**return** size==capacity;

}

**private** **boolean** isEmpty() {

**return** size==0;

}

**public** **void** enqueue(**int** item) {

**if**(isFull()) {

System.***out***.println("Queue is full");

**return**;

}

rear=(rear+1)%capacity;

queue[rear]=item;

size++;

}

**public** **int** dequeue() {

**if**(isEmpty()) {

System.***out***.println("Queue is empty");

**return** Integer.***MIN\_VALUE***;

}

**int** item=queue[front];

front=(front+1)%capacity;

size--;

**return** item;

}

**void** display() {

**if**(isEmpty()) {

System.***out***.println("Queue is Empty");

**return**;

}

**int** i=front;

**while**(i!=rear) {

System.***out***.println(queue[i]+" ");

i=(i+1)%capacity;

}

System.***out***.println(queue[i]);

}

**public** **static** **void** main(String[] args) {

CircularQueue queue=**new** CircularQueue(5);

queue.enqueue(14);

queue.enqueue(46);

queue.enqueue(89);

queue.enqueue(7);

queue.enqueue(-8);

System.***out***.println("Original Queue: ");

queue.display();

System.***out***.println();

System.***out***.println("Popped Element: "+queue.dequeue());

System.***out***.println();

System.***out***.println("Queue Now: ");

queue.display();

}

}

**11. Write a program to check whether a queue is empty or not.**

**Input 1:**

**Yellow Green Pink Black Blue White**

**Output 1:**

**Not Empty**

**Input 2:**

**[]**

**Output 2:**

**Empty**

**Code –**

**package** practiceday8;

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** EmptyQueue {

**public** **static** **void** main(String[] args) {

Queue<String> queue1=**new** LinkedList<>();

queue1.add("Yellow");

queue1.add("Green");

queue1.add("Pink");

queue1.add("Black");

queue1.add("Blue");

queue1.add("White");

Queue<String> queue2=**new** LinkedList<>();

*checkQueueEmpty*(queue1);

*checkQueueEmpty*(queue2);

}

**public** **static** **void** checkQueueEmpty(Queue<String> queue) {

**if**(queue.isEmpty()) {

System.***out***.println("Empty");

}

**else** {

System.***out***.println("Not Empty");

}

}

}

**12. Given a queue, split the queue into two queues, one containing odd numbers and the other even numbers. The**

**relative order of elements must be maintained in both the queues. Return an array containing the two queues,**

**the 0th index should contain the queue of odd numbers and the 1st index should contain the queue of even**

**numbers.**

**Input:**

**2 7 9 4 6 5 10**

**Output:**

**Odd Queue: 7 9 5**

**Even Queue: 2 4 6 10**

**Code –**

**package** practiceday8;

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** SplitQueue {

**public** **static** Queue<Integer>[] splitQueue(Queue<Integer> originalQueue){

Queue<Integer> oddQueue=**new** LinkedList<>();

Queue<Integer> evenQueue=**new** LinkedList<>();

**while**(!originalQueue.isEmpty()) {

**int** element=originalQueue.poll();

**if**(element%2==0) {

evenQueue.add(element);

}

**else** {

oddQueue.add(element);

}

}

Queue<Integer>[] result=**new** Queue[2];

result[0]=oddQueue;

result[1]=evenQueue;

**return** result;

}

**public** **static** **void** main(String[] args) {

Queue<Integer> originalQueue=**new** LinkedList<>();

originalQueue.add(2);

originalQueue.add(1);

originalQueue.add(3);

originalQueue.add(4);

originalQueue.add(5);

originalQueue.add(6);

originalQueue.add(7);

originalQueue.add(8);

System.***out***.println("Orginal Queue: ");

**for**(**int** num:originalQueue) {

System.***out***.println(num+" ");

}

System.***out***.println();

Queue<Integer>[] splitQueues=*splitQueue*(originalQueue);

System.***out***.println("Odd Queue: ");

**for**(**int** num:splitQueues[0]) {

System.***out***.println(num+" ");

}

System.***out***.println();

System.***out***.println("Even Queue: ");

**for**(**int** num:splitQueues[1]) {

System.***out***.println(num+" ");

}

}

}