Problem 2:
Implement Stack and Queues using Arrays.

Solution :

Stack :-

```
class StackCode
{
static final int max = 1000;
int top;
int a [] = new int [max];
booleanisEmpty ()
{
return (top <0);
}
StackCode ()
{
top = -1;
}
booleanpush (int x)
{
if (top >= (max-1))
{
System.out.println ("Stack is overflow");
return false;
}
else
{
a[++top] = x;
System.out.println(x +"pushed into stack");
return true;
}
}
intpop()
{
if (top < 0)
{
System.out.println ("Stack underflow");
return 0;
}
else
{
int x = a[top--];
return x;
}
}
intpeek()
{
if (top < 0)
{
System.out.println ("Stack underflow");
return 0;
}
else
{
int x = a[top];
return x;
}
}
}
```

```
class Stack
{
public static void main (String args[])
{
StackCode c = new StackProgram();
c.push (11);
c.push (22);
c.push (33);
System.out.println (c.pop() + " Popped from stack");
}
}
```

Queue :-

```
class Queue
{
private static int front, rear, capacity;
private static int queue[];
Queue (int c)
{
front = rear = 0;
capacity = c;
queue = new int [capacity];
}
static void queueEnqueue (int data)
{
if (capacity == rear)
{
System.out.println("Queue is full");
return;
}
else
{
queue [rear] = data;
rear++;
}
return;
}
static void queueDequeue ()
{
if (front == rear)
{
System.out.println ("Queue is empty");
return;
}
else
{
for (int i=0; i<rear-1; i++)
{
queue[i] = queue [i+1];
}
if (rear < capacity)
queue [rear] = 0;
rear--;
}
return;
}
static void queueDisplay()
{
int i;
if (front == rear)
{
System.out.println("Queue is empty");
return;
}
for (i=front; i<rear; i++)
{
System.out.println(queue[i]);
}
return;
}
public static void main(String args[])
{
Queue q=new Queue(4);
System.out.println("After inserting 4 elements in the
                        queue");
q.queueEnqueue(5);
q.queueEnqueue(15);
q.queueEnqueue(25);
q.queueEnqueue(35);
q.queueDisplay();
System.out.println("Try to insert 5th element in the
                        queue");
q.QueueEnqueue(45);
System.out.println("Try to display queue after deleting all
elements");
q.queueDequeue();
q.queueDequeue();
q.queueDequeue();
q.queueDequeue();
q.queueDisplay();
}
}
```