**Algorithms and Data Structures**

**Lab 3 : Week beginning 30th January, 2023**

**Array Algorithms and Using exceptions to make code more robust**

**Q1.** In a class called MyArrays.java write a method to insert an item, newElement, into an array, values, at position given by pos. The number of elements in the array is given by currentSize. If currentSize equals values.length, then it will not be possible to insert into the array.

You could make the code more robust by checking the value of pos. What values of pos should be allowed?

Test the code in main() method of another class called TestMyArrays.

The method header is

public static int insert(int [] values, int currentSize, int pos, int newElement)

The pseudocode is:

IF currentSize < values.length

FOR i from currentSize-1 downto pos

move values[i] down one slot

ENDFOR

values[pos] = newElement

INCREMENT currentSize

ENDIF

return currentSize

**Making the code more robust**

First, take a look at the slides “**Exception Handling**” in Teams->Files->Algorithms and Data Structures

To make the code for Q1 above more robust, add a new line to the beginning of the pseudocode:

The pseudocode then becomes:

validate pos // new line added

IF currentSize < values.length

FOR i from currentSize-1 downto pos

move values[i] down one slot

ENDFOR

values[pos] = newElement

INCREMENT currentSize

ENDIF

return currentSize

The best way to do the validation would be to throw an exception – what type of exception?

Look at Slide 2 for a suggestion.

In TestMyArrays, when testing the insert()method, be sure to see that the exception is thrown, if you do pass an illegal value for pos.

**Q2.** The code for a class to represent a playing card is:

**public class** Card{  
   
 **private int face**; *// 1 to 13 representing Ace to King* **private** String **suit**; *//"Hearts", "Diamonds", "Clubs", or "Spades"* **public** Card(**int** f, String s) {  
 **face** = f;  
 **suit** = s;  
 }  
   
 **public int** getFace(){  
 **return face**;  
 }  
  
 **public** String getSuit() {  
 **return suit**;  
 }  
   
 **public** String toString(){  
 **return " "** + **face** + **" of "** + **suit**;   
 }  
}

Write the code for a class Deck to represent a deck of cards.

It should have

Attribute – array of Card objects

Methods:

1. A constructor to construct a deck - the array of Card objects
2. A shuffle() method to shuffle the deck
3. A toString() method

Test the methods in a driver class.