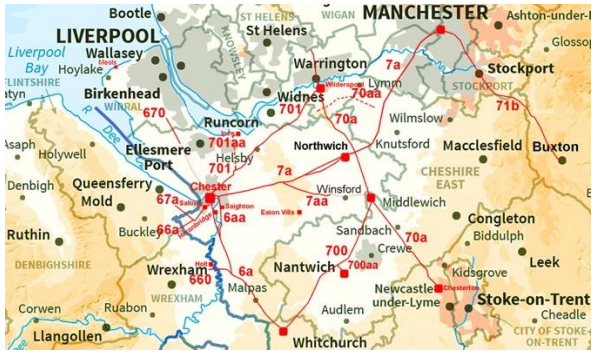
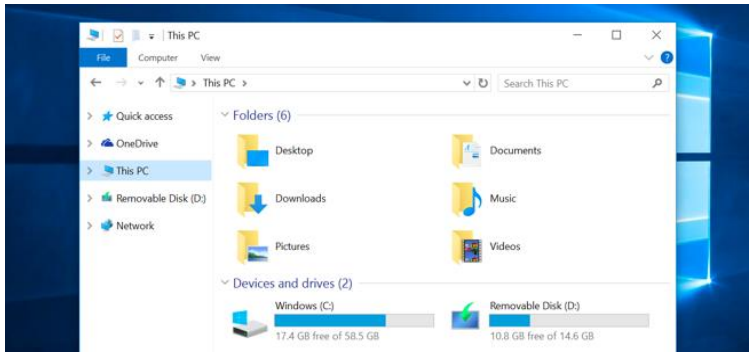
Ulster University

COM410 Programming in Practice

A5.1 Stacks

Recall Examples of Data Organisation





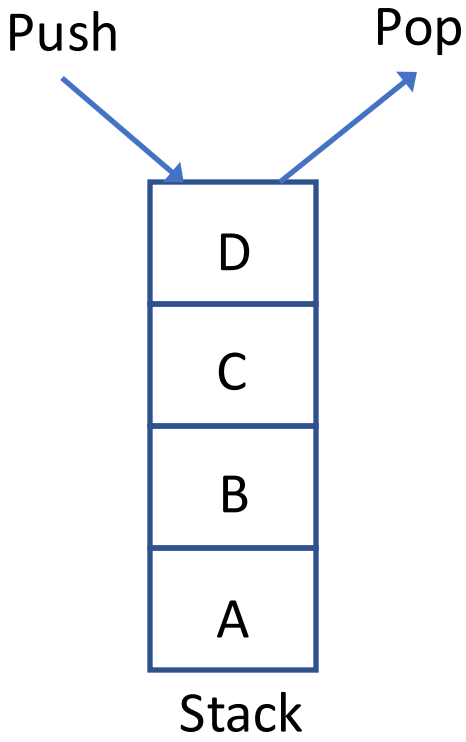
- Stack – Last in, first out – a very common data organisation technique in everyday life

Stack

- In everyday life a stack is a familiar thing (stack of books, stack of dishes, stack of presents . . .)
- When you remove an item, you take the one on the top of the stack
- Topmost item is the last one that was added to the stack

Stack Operations

- The behaviour of a stack is often described as **LIFO** (Last In, First Out)
- LIFO is exactly the behavior required by many important real-world applications
- Such algorithms make use of the Stack ADT



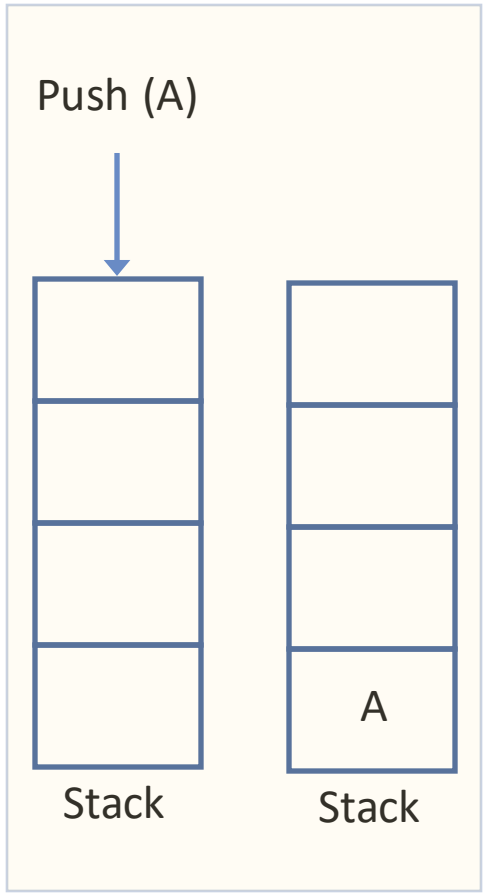
- In a stack all additions are to one end of the stack called the **top** (the entry at the top is the newest item in a stack)
- The operation that adds an entry to the stack is traditionally called **push**
- The operation that removes an entry from the stack is traditionally called **pop**

Stack Operations

- The stack restricts access to its entries (can only look at or remove top entry)
- In addition to push and pop, the operation to retrieve the top entry without removing it is called **peek**

- Typically you cannot search a stack for a specific entry
- The only way to look at an entry not at the top of the stack is to repeatedly remove items from the stack until the desired item reaches the top

Stack Operations



Java Interface for the Stack ADT

```
public interface StackInterface<T> {

    public void push (T newEntry);

    public T pop();

    public T peek();

    public boolean empty();

    public void clear();

}
```

- `push` → Add a new entry to the top
- `pop` → Remove entry from the top
- `peek` → Return entry from the top
- `empty` → Check for no entries in stack
- `clear` → Remove all entries

Java Stack Class

- Having defined our Stack Interface, we could implement all of the methods ourselves, just as we did earlier for the Bag.
- However, Java provides a built-in Stack class in the library `java.util.Stack` which we need to import at the top of the class in which we want to use it

```
1   import java.util.Stack;
2
3   public class UsingStacks {
4
5       public static void main(String[] args) {
6
7           Stack<String> myStack = new Stack<String>();
```

- Stacks of any object type can be created.

Java Stack Class

- Using the `empty()` method to check for an empty `Stack` and the `size()` method to report the number of elements contained
- Initially a newly-created `Stack` is empty.

```
9           System.out.println("\nShowing the initial stack");
10          System.out.println(myStack);
11          if (myStack.empty()) System.out.println("Stack is empty");
12          else System.out.printf("Stack contains %d entries \n", myStack.size());
```

```
Showing the initial stack
[]
Stack is empty
```

Java Stack Class

- Using the `push()` method to add an element to the stack
- New elements are added to the top of the stack

```
14          System.out.println("\nAdding Adrian, Belle and Charles");
15          myStack.push( item: "Adrian");
16          myStack.push( item: "Belle");
17          myStack.push( item: "Charles");
18          System.out.println(myStack);
19          if (myStack.empty()) System.out.println("Stack is empty");
20          else System.out.printf("Stack contains %d entries \n", myStack.size());
```

```
Adding Adrian, Belle and Charles
[Adrian, Belle, Charles]
Stack contains 3 entries
```

Java Stack Class

- Using the `pop()` method to remove an element from the stack
- Elements are removed from the top of the stack. The most recently added (`push`ed) element is the first to be removed (`pop`ped)

```
22          System.out.println("\nRemoving from the top of the stack");
23          System.out.println(myStack.pop());
24          System.out.println(myStack);
25          if (myStack.empty()) System.out.println("Stack is empty");
26          else System.out.printf("Stack contains %d entries \n", myStack.size());
```

```
Removing from the top of the stack
Charles
[Adrian, Belle]
Stack contains 2 entries
```

Java Stack Class

- Using the `peek()` method to return an element from the stack
- Elements are returned from the top of the stack but are NOT removed. The `peek()` operation does not change the stack contents

```
28          System.out.println("\nReading the element at the the top of the stack");
29          System.out.println(myStack.peek());
30          System.out.println(myStack);
31          if (myStack.empty()) System.out.println("Stack is empty");
32          else System.out.printf("Stack contains %d entries \n", myStack.size());
```

```
Reading the element at the the top of the stack
Belle
[Adrian, Belle]
Stack contains 2 entries
```

Java Stack Class

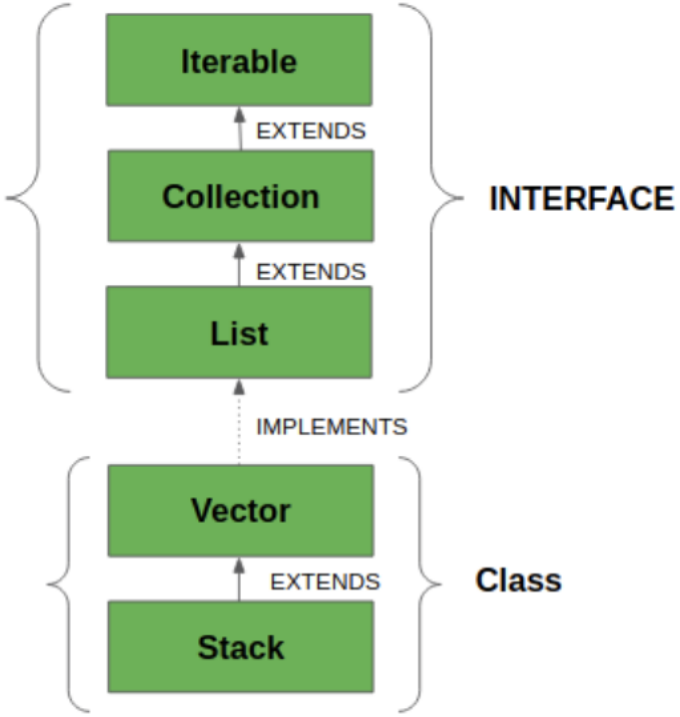
- Using the `clear()` method to empty the stack
- No elements are returned, but all are removed. The `clear()` operation returns the stack to its initial empty state

```
34          System.out.println("\nClearing the stack");
35          myStack.clear();
36          System.out.println(myStack);
37          if (myStack.empty()) System.out.println("Stack is empty");
38          else System.out.printf("Stack contains %d entries \n", myStack.size());
```

```
Clearing the stack
[]
Stack is empty
```

Note on the Java Class Libraries

- In the Java Class Library, `Stack` is defined as an extension of class `Vector`, which itself implements class `List`
- Even though the traditional stack methods `push()`, `pop()` and `peek()` are the only means of adding, accessing, and removing data, the Java `Stack` class inherits a very wide range of methods from its parent classes – which include non-traditional possibilities such as searching, retrieving from the middle, etc.
- We will not use these inherited methods here.



Scenario

- We have seen how pushing a series of elements onto a stack and then popping them off retrieves the elements in the reverse order from which they were originally presented.
- Create a new Java project `DataStructures` and a class within this called `Stacks`. In the `main()` method of your new class, implement a string reverse and palindrome check application as follows
    - The user should be prompted to enter a string from the keyboard
    - The individual characters should be retrieved one at a time, beginning from the 1st character in the string and pushed onto a stack
    - Once all characters are on the stack, they should be popped off one at a time with the individual characters joined to make a single string
    - If the reversed string is the same as the original, the message “<string> is a palindrome” should be displayed, where <string> is the original string provided by the user