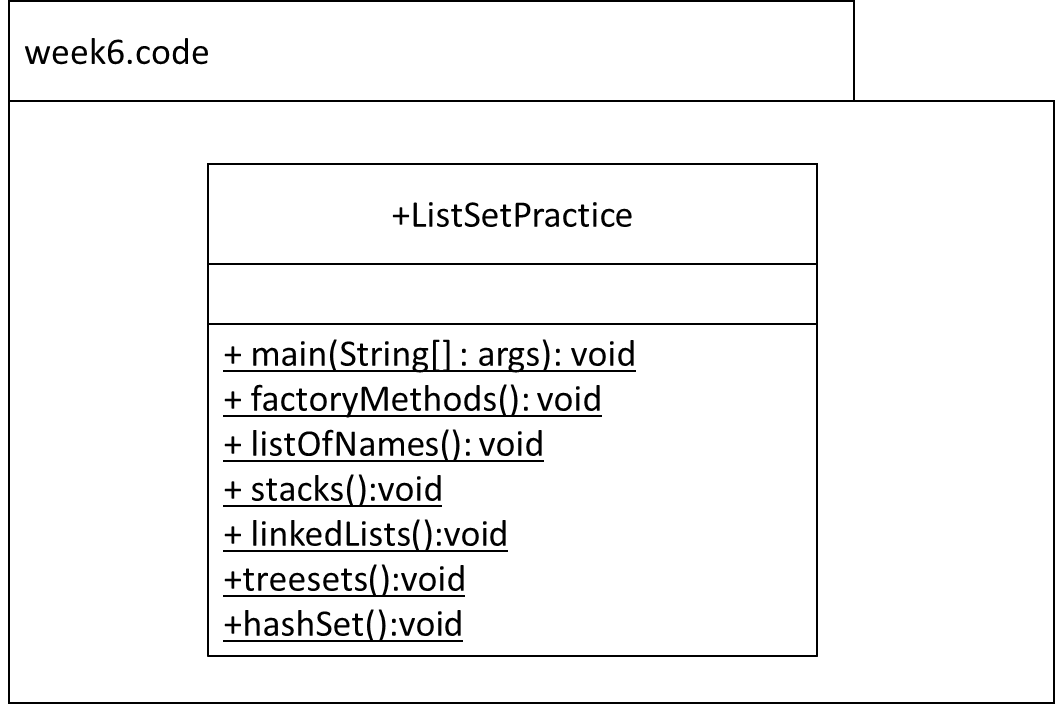
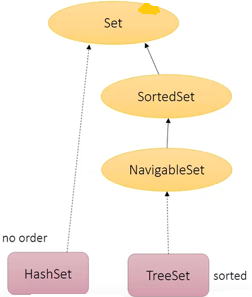
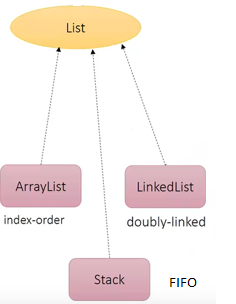
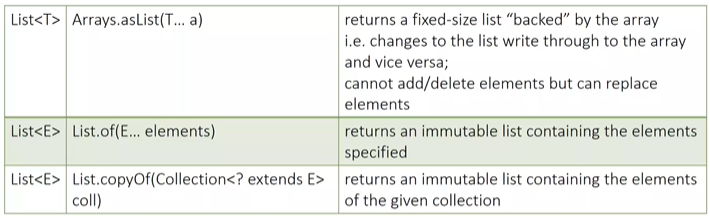
**Q1. Code the following**

**Q2. Create a simple List from simple String array and use factory methods**



asList(), of, copyOf()

* Declare a reference of type String[]. Name the reference myArray. Populate myArray with the values “A”, “B” and “C”.
* Declare a reference of type List that is generically typed for String i.e. it will only hold String references. Name the reference asList. Assign asList to the array created in step above. Hint: Use the asList() method.
* Output asList.
* Try and add an entry “D” to the list.
* What exceptions did you get?
* Output List.of()
* Output List.copyOf()

**Q3. List – ArrayList** – continue from **last week**

Methods: .add(), indexOf(), size(), contains(), remove(), isEmpty(), set(), **replaceAll(), get()**

Code the listOfNames method as follows:

* Declare a reference of type List that is generically typed for String i.e. it will only hold String references. Name the reference ‘myList’. The implementation type is ArrayList. Use type inference (the diamond operator).
* Insert the following in order: “Amy”, “Betty”, “Doris”, “Enda”,”Feena”
* Using the API i.e. do not hardcode the index, insert “Conor” in its correct position alphabetically. In other words, when you output the list, you should get [“Amy”, “Betty”, “Conor”, “Doris”, “Enda”, ”Feena”].
* Using the API, output the size of the list. This should be 6.
* Using the API, output whether the list contains “Mary”. This should output false.
* Using the API, delete “Betty” from the list.
* Output the list. You should get [“Amy”, “Conor”, “Doris”, “Enda”, ”Feena”].
* Using the API, output if the list is empty or not. This should output false.
* Using the API, change the “Feena” to a “Doris”.
* Output the list. You should get [“Amy”, “Conor”, “Doris”, “Enda”, ”Doris”].

**New week6**

* Using the API, add a surname “Bloggs” to each Name
* Output the list.
* Using the API, output the name in position 3.
* Using the API, replace the name in the last position (hardcode) with Andrea
* Output the list
* Using Collections to sort the list
* Output the list
* What do you notice in terms of the order in the list? Check the sort() on the use of the comparable interface Interface and then check the String documentation to see that String actually implements Comparable

**Q4. List – Stack**

Methods: push(), peek(), pop()

* Declare a reference of type Stack that is generically typed for String i.e. it will only hold String references. Name the reference ‘myStack’. The implementation type is Stack. Use type inference (the diamond operator).
* Using the API, add “Andrea”, “Barbara”, “Caroline” to the stack.
* Output the stack
* Output the stack peek()
* Output the stack pop()
* Output the stack
* What do you notice compared to List?
* Using the API, add 14 to the stack.
* What type of error do you get?

**Q5. List – LinkedList**

Methods: add(), addFirst(), addLast(),remove(), removeFirst(), removeLast(),peek()

* Declare a reference of type LinkedList that is generically typed for String i.e. it will only hold String references. Name the reference ‘myLinkedList’. The implementation type is LinkedList. Use type inference (the diamond operator).
* Using the API, add “Barbara” to the LinkedList.
* Output myLinkedList.
* Using the API, add “Andrea” to the first position to the LinkedList.
* Output myLinkedList.
* Using the API, add “Caroline” to the last position to the LinkedList.
* Output myLinkedList.
* Using the API, remove the first entry to the LinkedList.
* Output myLinkedList.
* Output the myLinkedList peek()
* Output the myLinkedList pop()
* Output myLinkedList
* **How does peek() and pop() work compared to Q4?**

**Q6. Set – TreeSet**

Methods: add()

* Declare a reference of type Set that is generically typed for String i.e. it will only hold String references. Name the reference ‘myTreeSet’. The implementation type is TreeSet. Use type inference (the diamond operator).
* Using the API, add “Barbara”, “Caroline”, “Andrea”, to the Treeset in that order.
* Output myTreeSet.
* **What do you notice?**