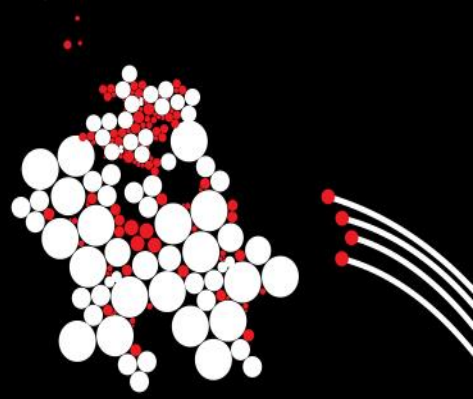
UNIVERSITY OF TWENTE.



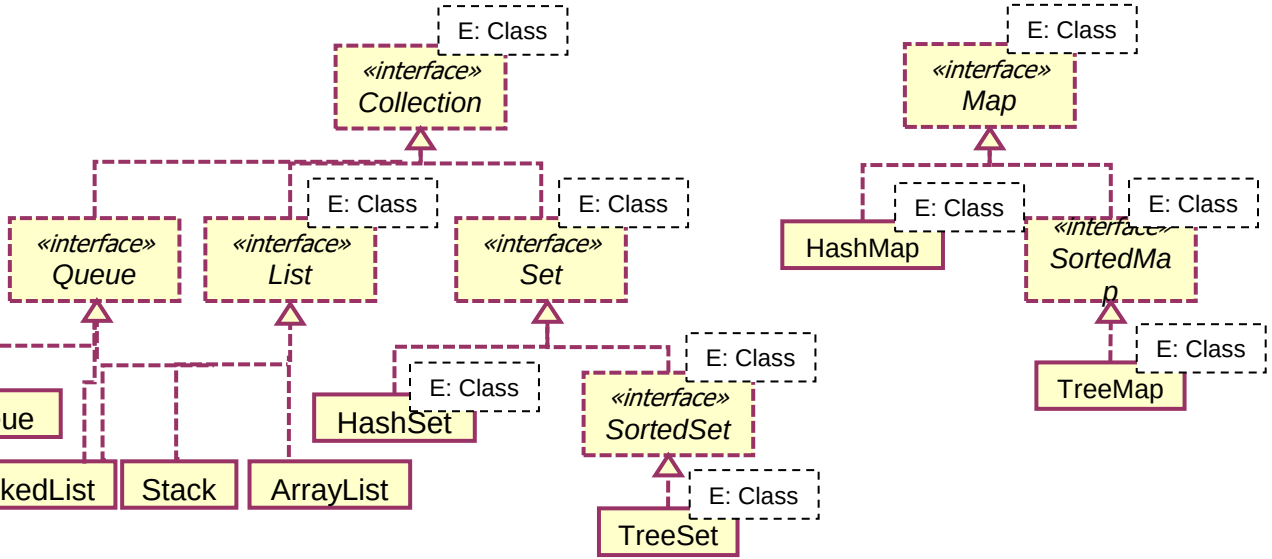
Queue and Stack

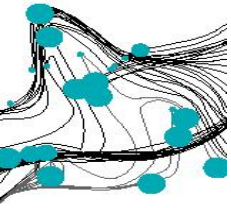
Topic of Software Systems (TCS module 2)

Lecturer: Faizan Ahmed

JAVA COLLECTION HIERARCHY

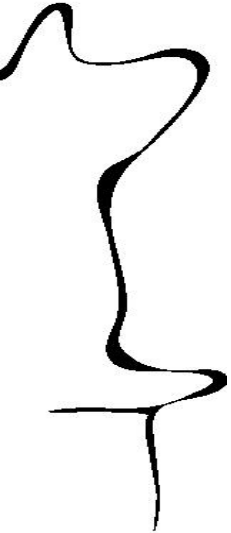
- Besides `List`, there are other fundamental data structures
  - `Set` implements the mathematical concept of a set (surprise…)
  - `Map` implements the mathematical concept of a function
  - Again, both have (many) different implementations





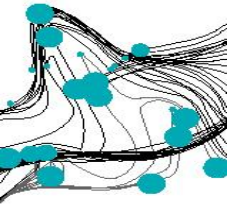
STACK

- Ordered collection of elements
- Elements are “stacked”
- Last In First Out (LIFO)
- Only the top is accessible
- New elements come on top



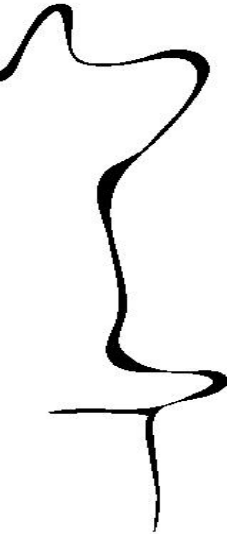
STACK

- class Stack<E> extends Vector<E>
- Most important methods:
  - E push(E e)
    - puts e on top of the stack
  - E pop()
    - removes the top element from the stack and returns it
  - E peek()
    - returns the top element of the stack (leaves it in place)
  - boolean empty()
    - returns true if the stack is empty

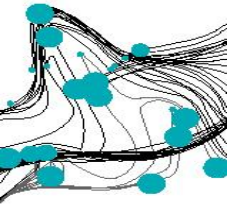


IMPLEMENTATION

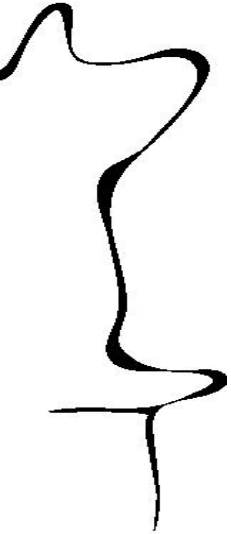
- Array



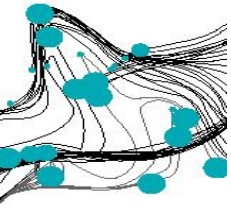
- Linked List



QUEUE

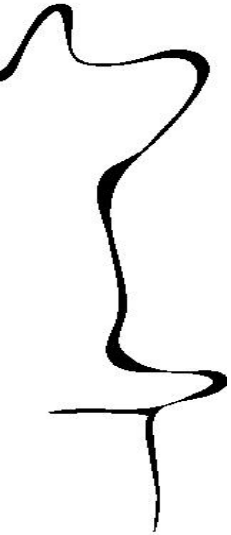


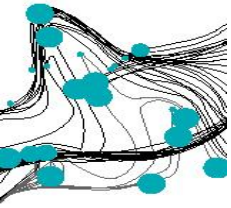
- Ordered collection of elements
- Elements stand one behind the other
- First In First Out (FIFO)
- Only the first is accessible
- New elements come at the back



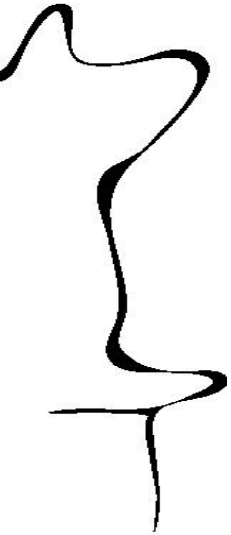
QUEUE

- interface Queue<E> extends Collection<E>
- Most important methods:
  - boolean add(E e)
    - adds e at the back of the queue
      (or an exception if the queue is full)
  - E remove()
    - removes the first from the queue and returns it
      (or an exception if the queue is empty)
  - E peek()
    - returns the first element of the queue (leaves it in place)
      (or null if the queue is empty)

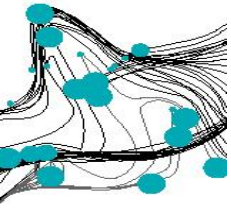




PRIORITY QUEUE

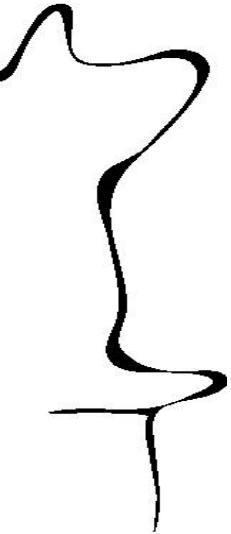


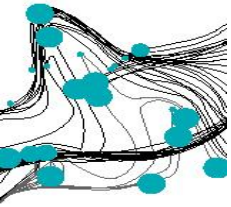
- Ordered collection of elements
- Elements stand one behind the other
- Ordered by priority, higher is first
- First is accessible
- New elements come in between according to their priority



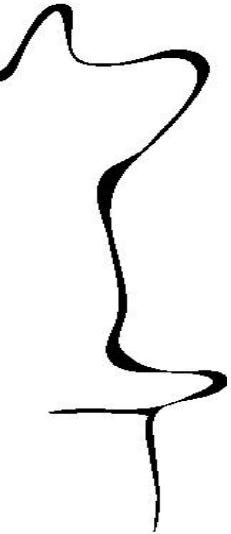
PRIORITY QUEUE

- class PriorityQueue<E> extends AbstractQueue<E>
- Most important methods:
  - PriorityQueue()
    - Constructor for a priority queue with natural order
  - PriorityQueue(int s, ComparatorE> comp)
    - Constructor for a priority queue with order based upon the Comparator
  - boolean add(E e)
    - adds e at the right position in the priority queue
  - E poll()
    - removes the first element from the priority queue and returns it
  - E peek()
    - returns the first element of the priority queue (leaves it in place)





JAVA COLLECTION SUMMARY



- `Collection`: general methods (`add`, `remove`, `contains`, `iterator`, …)
- `List`: see above (implementations: `ArrayList`, `LinkedList`)
- `Set`: no duplicates, no indexing (`get`, `set`), no predetermined ordering
  - `HashSet`: fast implementation based on hash codes
  - Requires element type to have overwritten `equals` and `hashcode`
- `SortedSet`: set with predetermined ordering (still no indexing)
  - Requires element type to be subtype of (interface) `Comparable`
  - `TreeSet`: `SortedSet` implementation based on binary trees
  - Slightly less efficient than `HashSet`
- `Map`: implements the mathematical concept of a function
  - `HashMap`: fast implementation based on hash codes
- `SortedMap`: map with fixed ordering, key type should be `Comparable`
  - `TreeMap`: `SortedMap` implementation based on binary trees