Faculty Civil Engineering
Chair of Intelligent Technical Design
Prof. Dr.-Ing. Christian Koch

Bauhaus-Universität Weimar

Object-oriented Modeling and Programming in Engineering (OOMPE)

Winter semester 2018-19

06 – Data structures

Structuring data

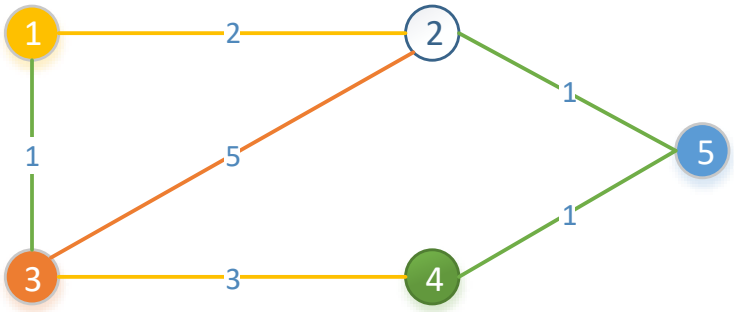
- Think about what is your data
  - Composition of primitive data types, e.g. numbers, characters, boolean, strings,… (most times easy)
  - Think about how is your data related? (get to know some patterns)
- Array: multiple objects of the same type, fixed size (e.g. vectors, matrices, images)
- List: multiple objects of the same type, variable size (e.g. users in a system, stock in a ware house)
- Map: key-value pairs (e.g. user and password, id and username, …)
- Trees:
  - hierarchical data (e.g. directories, models, structures)
  - object groups (e.g. search within data, routing, travelling salesman)
- Stack: First in Last out (e.g. Backward function in Browser)
- Queue: First in First out (e.g. a queue)
- …

double linked list

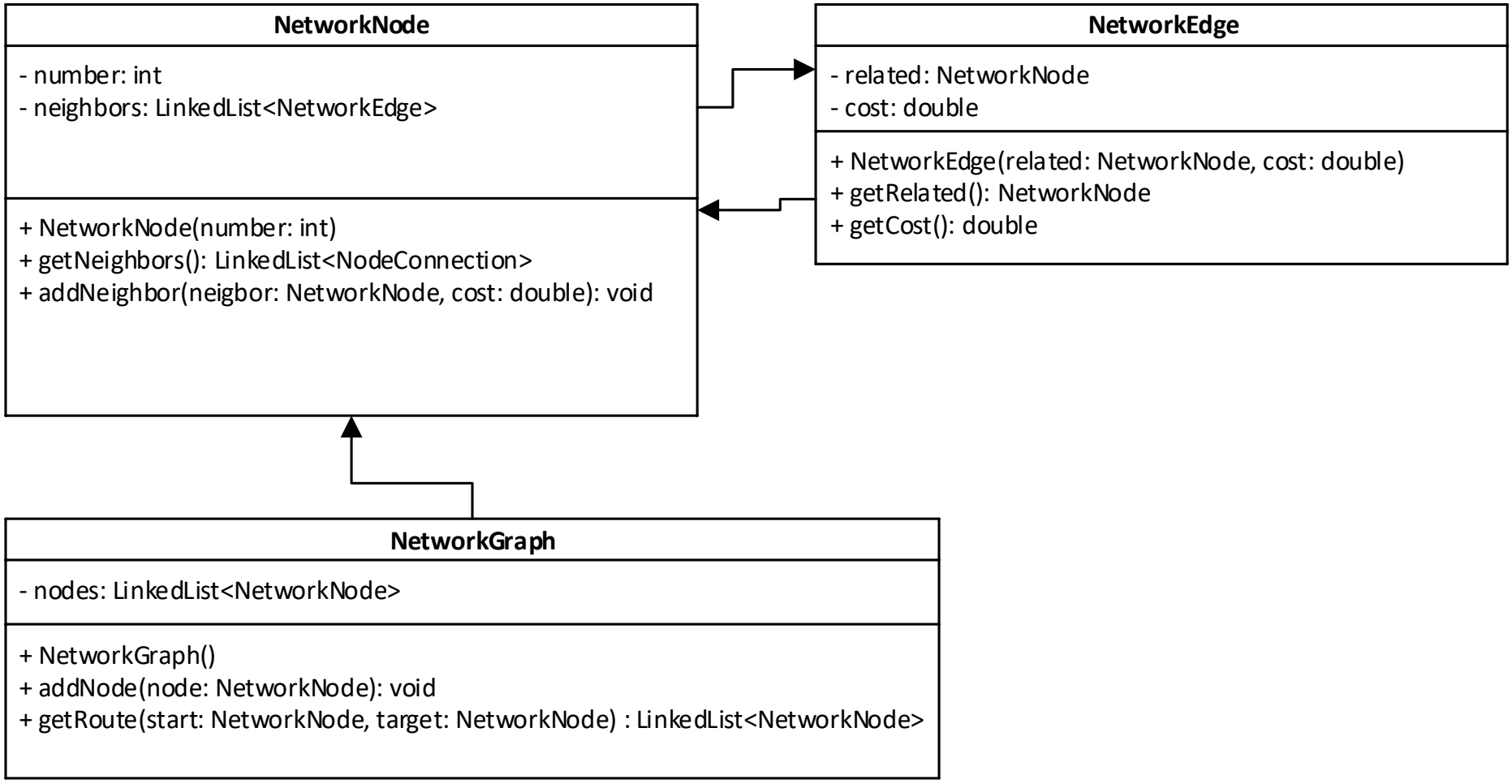
- Implement the classes needed for a ascending sorted double list
  - Design two classes: one for a node, one for the list
  - Each node has to relating elements: previous and next
  - previous of the list head points to the end of the list
  - The list shall offer the following options:
    - add a new double to it
    - remove an element at a specified position
    - Sort the list ascending

Network diagram

- Think about a network with different routers
- Find a route from [start] to [end]
- Use 4 classes for this
  - A class for a single node
  - A class for a connection of two nodes
  - A class for a network
  - A class for testing (main method)



UML Diagram


NetworkNode
- number: int
- neighbors: LinkedList<NetworkEdge>
+ NetworkNode(number: int)
+ getNeighbors(): LinkedList<NodeConnection>
+ addNeighbor(neigbor: NetworkNode, cost: double): void
NetworkEdge
- related: NetworkNode
- cost: double
+ NetworkEdge(related: NetworkNode, cost: double)
+ getRelated(): NetworkNode
+ getCost(): double
NetworkGraph
- nodes: LinkedList<NetworkNode>
+ NetworkGraph()
+ addNode(node: NetworkNode): void
+ getRoute(start: NetworkNode, target: NetworkNode) : LinkedList<NetworkNode>