Computer Science 3A
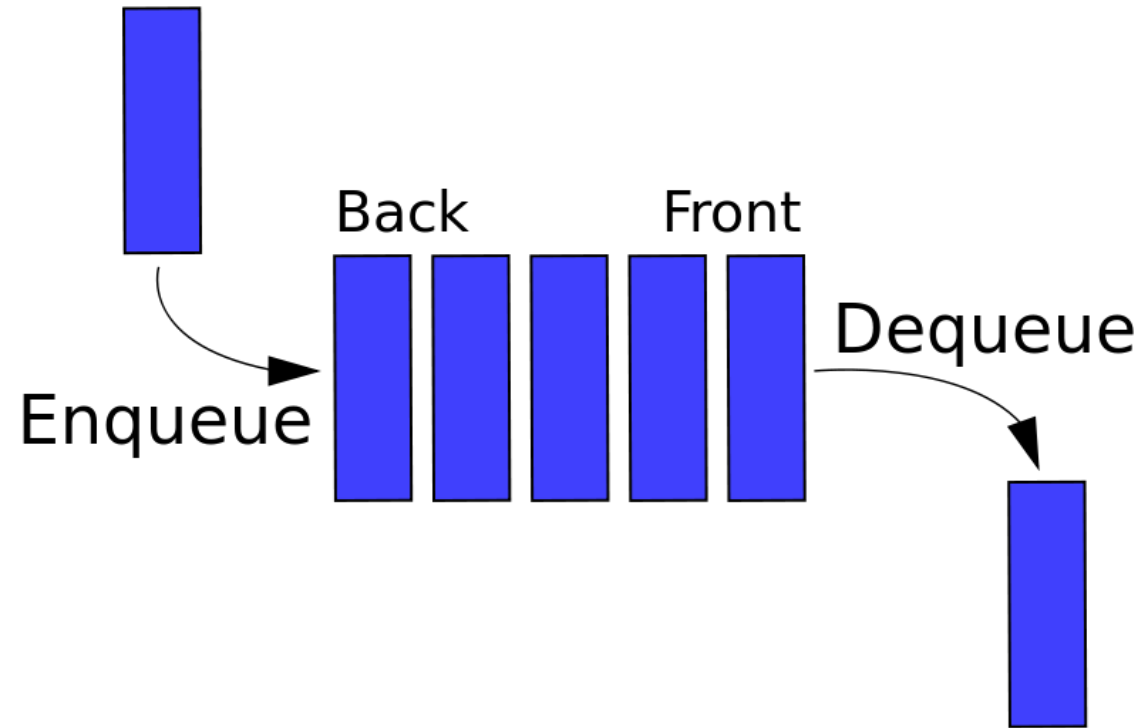
Practical 4

2023

The Queue ADT

"First In First Out"

Building on our Singly Linked Lists:

The basic Singly Linked List:

A node

head

tail

1

2

3

| element | next |

| element | next |

| element | next |
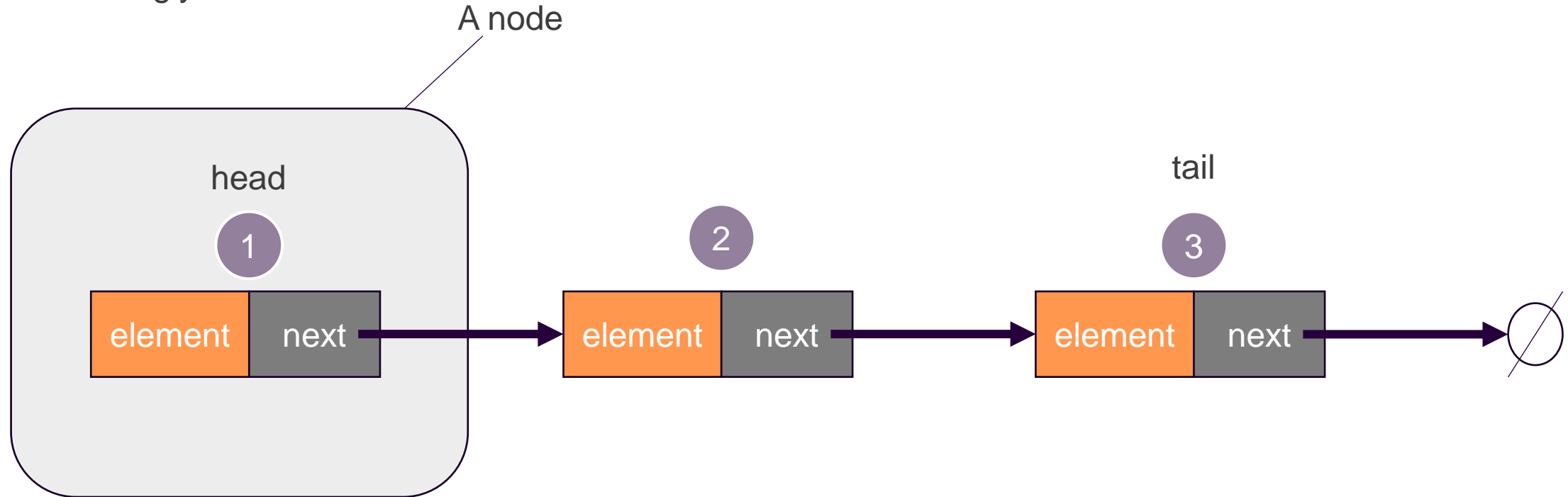
Size = 3

Practical 4 - Overview:

1. **Implement a Queue Class using Adaptor pattern**

2. **Complete the Main Class**

Adapter Pattern

• **Adapter pattern** allows you to convert the interface of a class into another interface which a client expects.
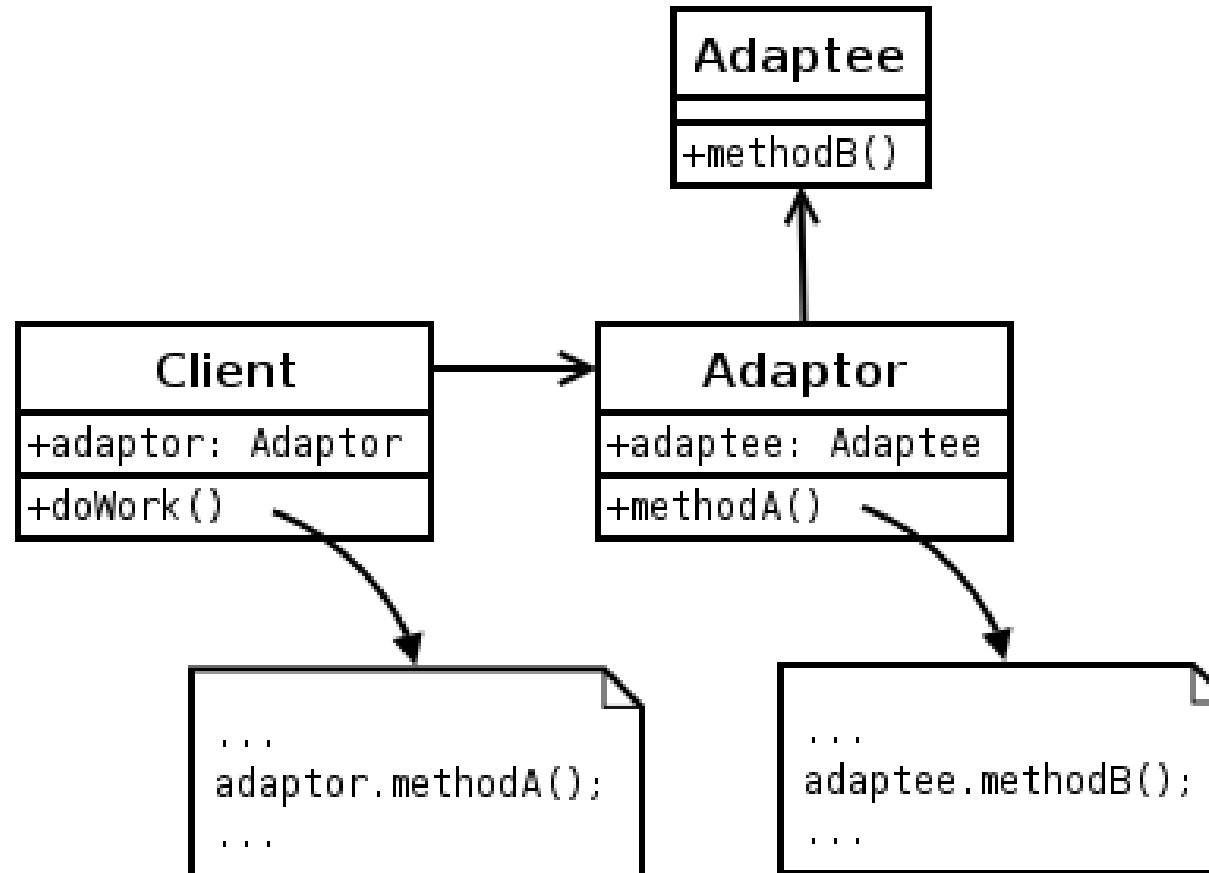
For Practical 4:

"Calculating capital gain or loss"

Calculating capital gain or loss

- Given a sequence of transactions:
  - Transaction: BUY 100 20
  - Transaction: BUY 20 24
  - Transaction: BUY 200 36
  - Transaction: SELL 150 30

  - Calculate the capital gain (or loss):R940

For Practical 4:

BUY 100 20
BUY 20 24
BUY 200 36
SELL 150 30

Buy Queue:

BUY 100 20
BUY 20 24
BUY 200 36

==Sell shares that have been held the longest (FIFO):==

100 of the 150 required were bought at 20:
Profit = (30 – 20) * 100 = 1000

20 of the 150 required were bought at 25:
Profit = (30 – 24) * 20 = 120

30 of the 150 required were bought at 36:
Profit = (30 – 36) * 30 = -180

Total capital gain (loss):
1000+120-180 = R940

Sell Queue:

SELL 150 30

Hint: 170 shares bought at 36 should remain in our queue

Using JAR Files

Using Eclipse

- Extract studentnumber_p4.zip
- Rename extracted folder to *yourstudentnumber*_p4
- Move folder to eclipse workspace
- Create new project with the same name, i.e. *yourstudentnumber*_p4

If you get errors:

- Create new project with the same name, i.e. *yourstudentnumber*_p4
- Properties > Java Build Path > Add External JARs
- Navigate to given JAR file
- Select JAR file and click open
- Click Apply and close

Classes Contained in the JAR file

<<Java Interface>>
IQueue<T>
(default package)

- enqueue(T):void
- first()
- dequeue()
- size():int
- isEmpty():boolean

<<Java Class>>
LinkedQueue<T>
(default package)

- LinkedQueue()
- size():int
- isEmpty():boolean
- enqueue(T):void
- first()
- dequeue()
- clone():LinkedQueue<T>
- toString():String

<<Java Class>>
SList<T>
(default package)

- head: Node<T>
- tail: Node<T>
- size: Integer

- SList()
- clone():SList<T>
- addAfter(Node<T>,T):void
- addFirst(T):void
- addLast(T):void
- removeFirst()
- search(T):Node<T>
- first()
- isEmpty():boolean
- size():Integer
- toString():String

-list
0..1

<<Java Class>>
Transaction
(default package)

- quantity: int
- unitPrice: int

- Transaction(int,int)
- getQuantity():int
- setQuantity(int):void
- getUnitPrice():int
- setUnitPrice(int):void

<<Java Interface>>
① IQueue<T>
(default package)

- ● enqueue(T):void
- ● first()
- ● dequeue()
- ● size():int
- ● isEmpty():boolean

<<Java Class>>
ⓒ LinkedQueue<T>
(default package)

- ●c LinkedQueue()
- ● size():int
- ● isEmpty():boolean
- ● enqueue(T):void
- ● first()
- ● dequeue()
- ● clone():LinkedQueue<T>
- ● toString():String

<<Java Class>>
ⓒ SList<T>
(default package)

- ◻ size: Integer

- ●c SList()
- ● clone():SList<T>
- ● addAfter(Node<T>,T):void
- ● addFirst(T):void
- ● addLast(T):void
- ● removeFirst()
- ● search(T):Node<T>
- ● first()
- ● isEmpty():boolean
- ● size():Integer
- ● toString():String

-list
0..1

<<Java Class>>
ⓒ Transaction
(default package)

- △ quantity: int
- △ unitPrice: int

- △c Transaction(int,int)
- ● getQuantity():int
- ● setQuantity(int):void
- ● getUnitPrice():int
- ● setUnitPrice(int):void

~sellQueue
~buyQueue

0..1 0..1

<<Java Class>>
ⓒ Main
(default package)

- △S totalGain: int

- ●c Main()
- ●S processTransactions(LinkedQueue<String>):void
- ●S calculateCapitalGainLoss():Integer
- ●S main(String[]):void

-head 0..1
-tail 0..1

<<Java Class>>
ⓒ Node<T>
(default package)

- ◻ element: T

- ●c Node(Node<T>,T)
- ● setElement(T):void
- ● getElement()
- ● getNext():Node<T>
- ● setNext(Node<T>):void
- ● toString():String

-next
0..1