Queue

Ja-Hee Kim

Agenda

01 ADT

Queue & Deque (enqueue, dequeue, getFront, isEmpty, clear)

02 Examples

Simulating a waiting line, Computing the Capital Gain in a Sale of Stock,

03 Linked Queue

Enqueue & dequeue

04 Array Queue

Circular queue

ADT of Queue

Some everyday queues

ADT of a Queue

Queue
+enqueue(newEntry: integer): void +dequeue(): T +getFront(): T +isEmpty(): Boolean +clear(): void

- enqueue:
  - =put, insert
    - Task: add a given element to the back of the queue.
    - Input: newEntry is a new entry.
- dequeue:
  - =get, remove
    - Task: if the queue is not empty, delete and return the entry a front of the queue
    - Output: Returns either the queue's front entry or, if the queue is empty before the operation, null.
- getFront:
  - = peek
    - Task: Retrieves the queue's front entry without changing the queue in any way.
    - Output: Returns either the queue's front entry or, if the queue is empty, null.
- isEmpty:
  - Task: detects whether the queue is empty.
  - Output: returns true if the queue is empty.
- clear:
  - Task: removes all entries from the queue

An interface of the ADT Queue

Queue
+enqueue(newEntry: integer): void +deque(): T +getFront(): T +isEmpty(): Boolean +clear(): void

```
public interface QueueInterface<T>{
    /** Adds a new entry to the back of the queue.
     * @param newEntry an object to be added */
    public void enqueue(T newEntry);
    /** Removes and returns the entry at the front of this queue
     * @return either the object at the front of the queue or, if the
     * queue is empty before the operation, null */
    public T dequeue();
    /** Retrieves the entry at the front of this queue.
     * @return either the object at the front of the queue or, if the
     * queue is empty, null */
    public T getFront();
    /** Detects whether this queue is empty.
     * @return true if the queue is empty, or false otherwise */
    public boolean isEmpty();
    /** Removes all entries from this queue. */
    public void clear();
} // end QueueInterface
```

Build-in library, Queue

java.util

Interface Queue<E>

Type Parameters:

`E - the type of elements held in this collection`

All Superinterfaces:

`Collection<E>, Iterable<E>`

All Known Subinterfaces:

`BlockingDeque<E>, BlockingQueue<E>, Deque<E>, TransferQueue<E>`

All Known Implementing Classes:

`AbstractQueue, ArrayBlockingQueue, ArrayDeque, ConcurrentLinkedDeque, ConcurrentLinkedQueue, DelayQueue, LinkedBlockingDeque, LinkedBlockingQueue, LinkedList, LinkedTransferQueue, PriorityBlockingQueue, PriorityQueue, SynchronousQueue`

```
public interface Queue<E>
extends Collection<E>
```

A collection designed for holding elements prior to processing. Besides basic `Collection` operations, queues provide additional insertion, extraction, and inspection operations. Each of these methods exists in two forms: one throws an exception if the operation fails, the other returns a special value (either `null` or `false`, depending on the operation). The latter form of the insert operation is designed specifically for use with capacity-restricted `Queue` implementations; in most implementations, insert operations cannot fail.

Summary of Queue methods

	Throws exception	*Returns special value*
Insert	`add(e)`	`offer(e)`
Remove	`remove()`	`poll()`
Examine	`element()`	`peek()`

Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner. Among the exceptions are priority queues, which order elements according to a supplied comparator, or the elements' natural ordering, and LIFO queues (or stacks) which order the elements LIFO (last-in-first-out). Whatever the ordering used, the *head* of the queue is that element which would be removed by a call to `remove()` or `poll()`. In a FIFO queue, all new elements are inserted at the *tail* of the queue. Other kinds of queues may use different placement rules. Every `Queue` implementation must specify its ordering properties.

Usage of a Queue

- Demonstration the queue methods: a pipe of fruits





- Demo
  1. enqueue
     offer

  7. enqueue
     offer

  3. enqueue
     offer

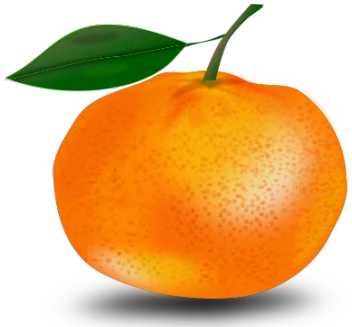
  4. dequeue(poll)
  5. getFront (peek)
  6. size
  7. enqueue
     offer

  8. enqueue
     offer

enqueue

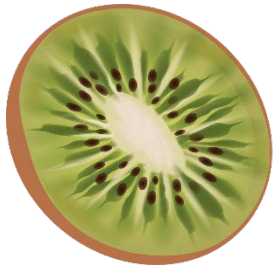
```
crate.enqueue("Apple");
```

enqueue



```
crate.enqueue("Apple");
crate.enqueue("Orange");
```

enqueue



```
crate.enqueue("Apple");
crate.enqueue("Orange");
crate.enqueue("Kiwi");
```

dequeue

```java
public class QueueTest {
    public static void main(String[] args) {
        QueueInterface<String> crate = new LinkedQueue<>();
        crate.enqueue("Apple");
        crate.enqueue("Orange");
        crate.enqueue("Kiwi");
        System.out.println("dequeue: "+ crate.dequeue());
    }
}
```

Problems Javadoc Declaration Console

<terminated> QueueTest [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2020. 10. 6. 오후 3:08:29 –

```
dequeue: Apple
```

getFront & size

```java
public class QueueTest {
    public static void main(String[] args) {
        QueueInterface<String> crate = new LinkedQueue<>();
        crate.enqueue("Apple");
        crate.enqueue("Orange");
        crate.enqueue("Kiwi");
        System.out.println("dequeue: "+ crate.dequeue());
        System.out.println("size: "+ crate.size()+", getFront: "+ crate.getFront());
    }
}
```

Problems Javadoc Declaration Console

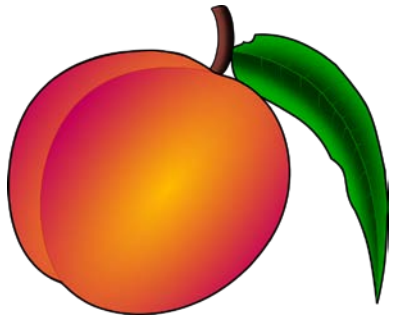
<terminated> QueueTest [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2020. 10. 6. 오후 3:09:20 – 오후 3:09:20)

```
dequeue: Apple
size: 2, getFront: Orange
```

enqueue

```
QueueInterface<String> crate = new LinkedQueue<>();
crate.enqueue("Apple");
crate.enqueue("Orange");
crate.enqueue("Kiwi");
System.out.println("dequeue: "+ crate.dequeue());
System.out.println("size: "+ crate.size()+", getFront: "+ crate.getFront());
crate.enqueue("Paprika");
```

enqueue



```java
public class QueueTest {
    public static void main(String[] args) {
        QueueInterface<String> crate = new LinkedQueue<>();
        crate.enqueue("Apple");
        crate.enqueue("Orange");
        crate.enqueue("Kiwi");
        System.out.println("dequeue: "+ crate.dequeue());
        System.out.println("size: "+ crate.size()+", getFront: "+ crate.getFront());
        crate.enqueue("Paprika");
        crate.enqueue("Peach");
        System.out.println("Queue["+ crate.size() +"]: "+ crate.toString());
    }
}
```

Problems Javadoc Declaration Console

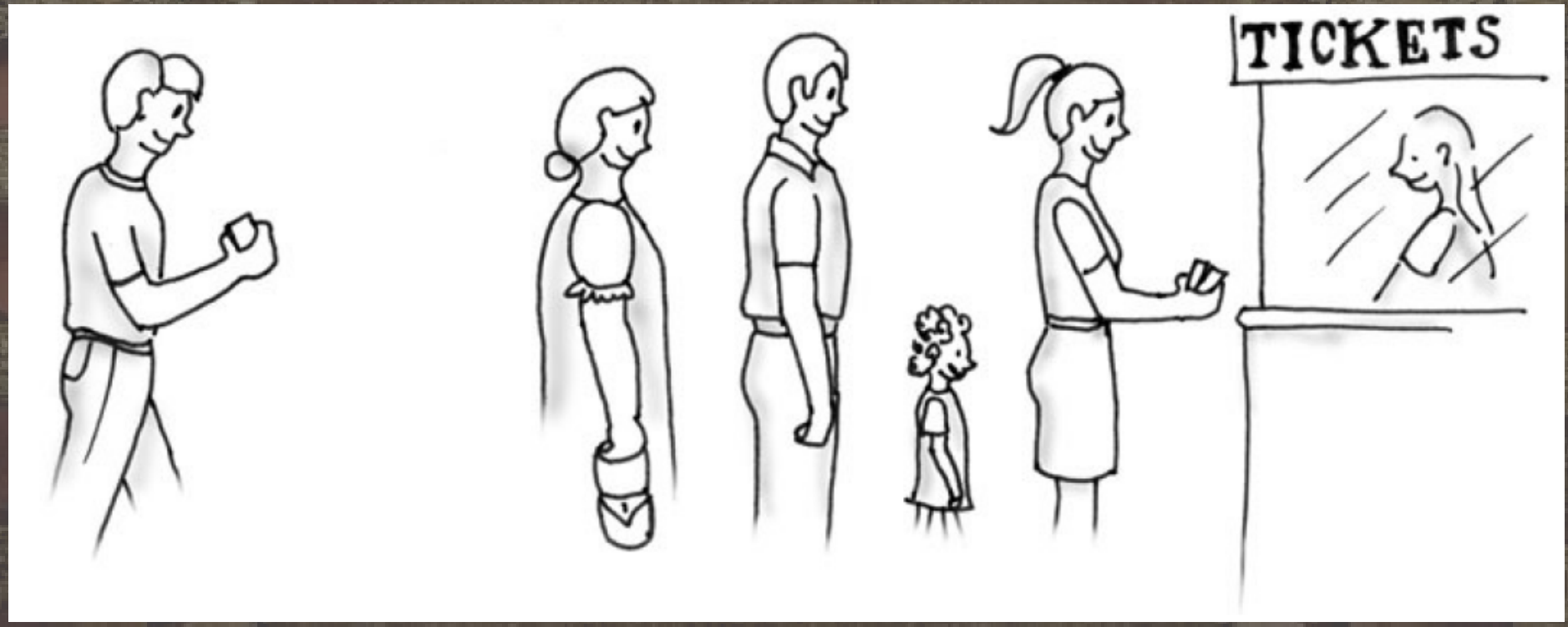
<terminated> QueueTest [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2020. 10. 6. 오후 3:10:58 – 오후 3:11:02)

```
dequeue: Apple
size: 2, getFront: Orange
Queue[4]: [Orange, Kiwi, Paprika, Peach]
```

Examples of Queue

Examples1 - Simulating a Waiting Line

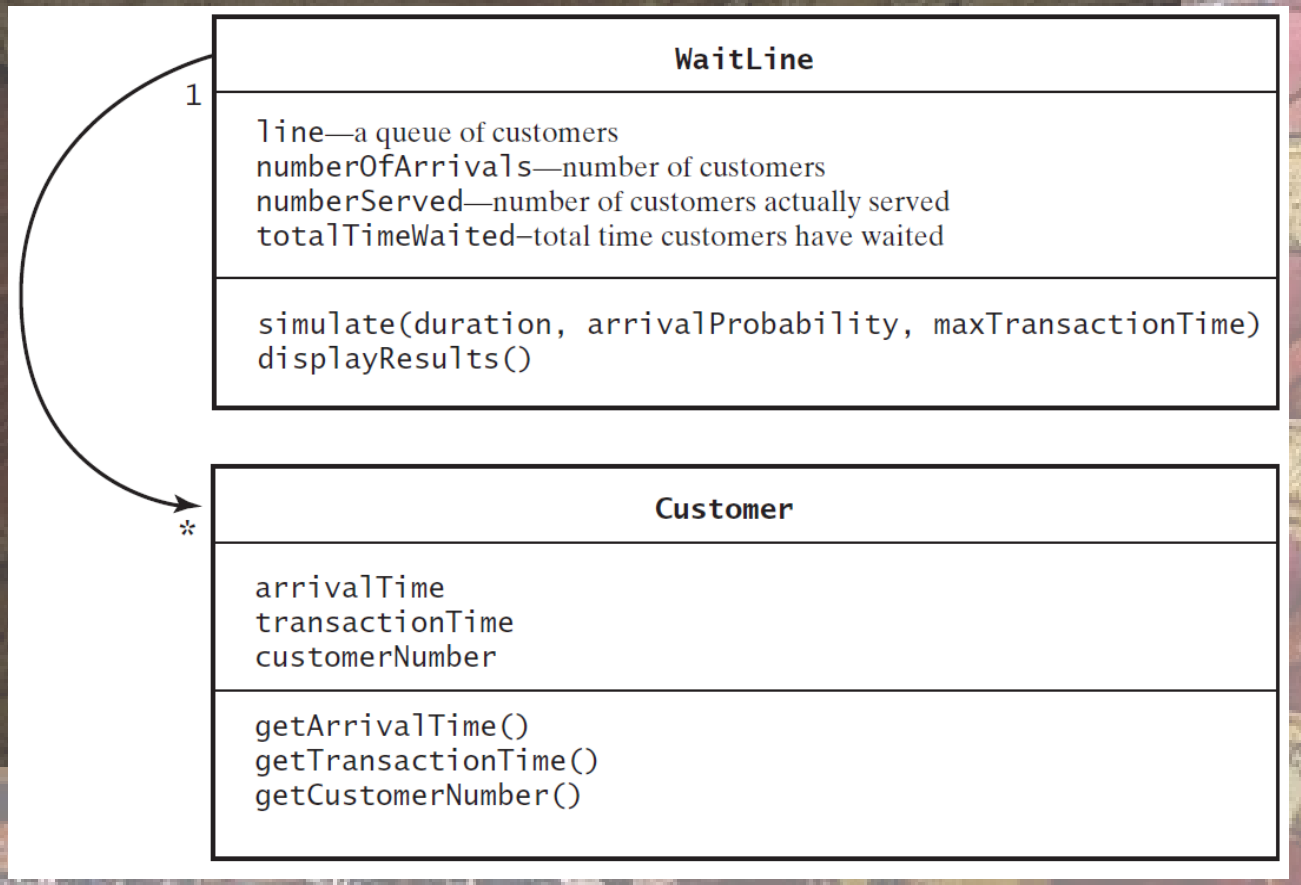
- Problem: a company wants to know the smallest number of agents not to increase the waiting time of the customers.
- Solution: computer simulation



Design of a Waiting Line

- Responsibility of a waiting line
  - Simulate customers entering and leaving a waiting line
  - Display number served, total wait time, average wait time, and number left in line

- Responsibility of a customer
  - Arrive the ticket box
  - Get the service
  - Get the customer number



A simulated waiting line

- Responsibility of a waiting line
  - Simulate customers entering and leaving a waiting line
  - Display number served, total wait time, average wait time, and number left in line

- Responsibility of a customer
  - Arrival time the ticket box
  - Get the service time
  - Get the customer number

Method simulate

```
public void simulate(int duration, double arrivalProbability,
        int maxTransactionTime) {
    int transactionTimeLeft = 0;
    for (int clock = 0; clock < duration; clock++)      {
        if (Math.random() < arrivalProbability)         {
            numberArrivals++;
            int transactionTime = (int)(Math.random()
                    * maxTransactionTime + 1);
            line.offer(new Customer(clock, transactionTime));
            System.out.println("Customer " + numberArrivals
                    + " enters line at time " + clock
                    + ". Transaction time is "
                    + transactionTime);
        } // end if
        if (transactionTimeLeft > 0)
            transactionTimeLeft--;
        else if (!line.isEmpty()){
            Customer nextCustomer = line.poll();
            if(nextCustomer != null) {
                transactionTimeLeft = nextCustomer.getTransactionTime() - 1;
                int timeWaited = clock - nextCustomer.getArrivalTime();
                totalTimeWaited = totalTimeWaited + timeWaited;
                numberServed++;
                System.out.println("Customer "
                        + nextCustomer.getCustomerNumber()
                        + " begins service at time " + clock
                        + ". Time waited is " + timeWaited);
            }
        } // end if
    } // end for
} // end simulate
```

```
public void displayResults()    {
    System.out.println();
    System.out.println("Number served = " + numberServed);
    System.out.println("Total time waited = " + totalTimeWaited);
    double averageTimeWaited = ((double)totalTimeWaited) /numberServed;
    System.out.println("Average time waited = " + averageTimeWaited);
    int leftInLine = numberArrivals - numberServed;
    System.out.println("Number left in line = " + leftInLine);
}
```

Pseudo-random numbers

no true random number, because it is determined by an initial value, seed, but it is close to truly random.

Math.random()

uniformly distributed over the interval from 0 to 1

Execution

```
1 public class SimulationQueue {
2     public static void main(String[] args) {
3         WaitingLine customerLine = new WaitingLine();
4         customerLine.simulate(10, 0.5, 3);
5         customerLine.displayResults();
6     }
7 }
```

Problems Javadoc Declaration Console

<terminated> SimulationQueue [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2020. 10. 6

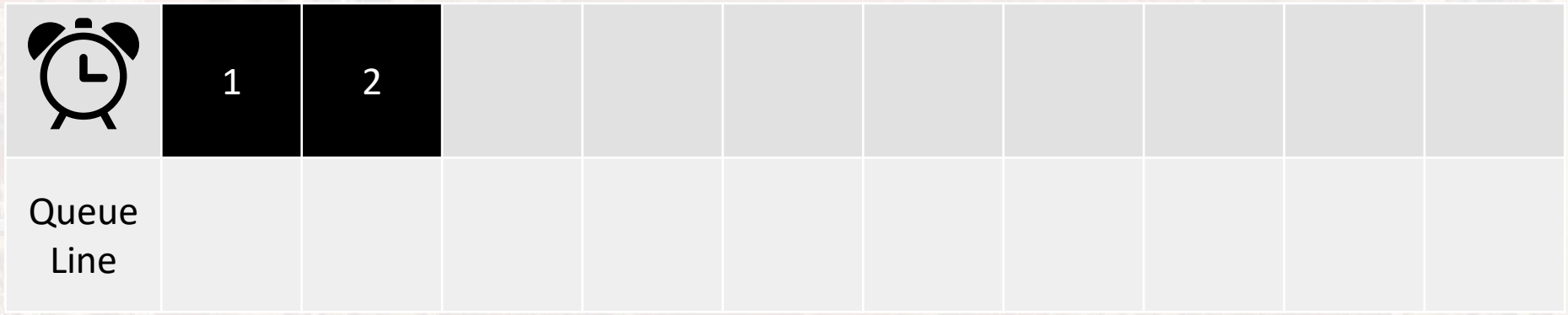
```
Customer 1 enters line at time 3. Transaction time is 3
Customer 1 begins service at time 3. Time waited is 0
Customer 2 enters line at time 4. Transaction time is 1
Customer 3 enters line at time 6. Transaction time is 3
Customer 2 begins service at time 6. Time waited is 2
Customer 4 enters line at time 7. Transaction time is 2
Customer 3 begins service at time 7. Time waited is 1
Customer 5 enters line at time 9. Transaction time is 3

Number served = 3
Total time waited = 3
Average time waited = 1.0
Number left in line = 2
```

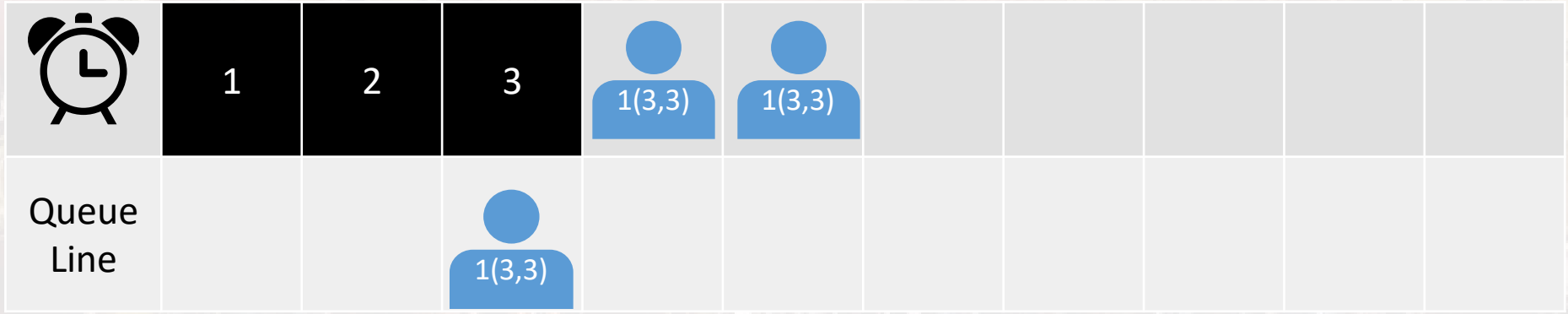
Customer number	Arrival time	Service (Transaction) time
1(3,3)	3	3
2(4,1)	4	1
3(6,3)	6	3
4(7,2)	7	2
5(9,3)	9	3

Simulation

NumberArrivals:0
numberServed:0
totalTimeWaited:0

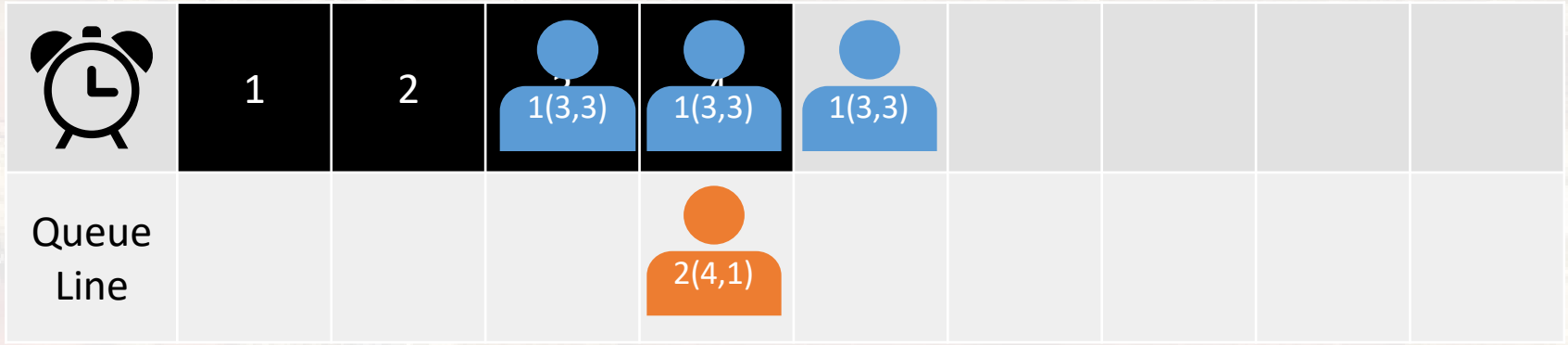


NumberArrivals:0
numberServed:0
totalTimeWaited:0

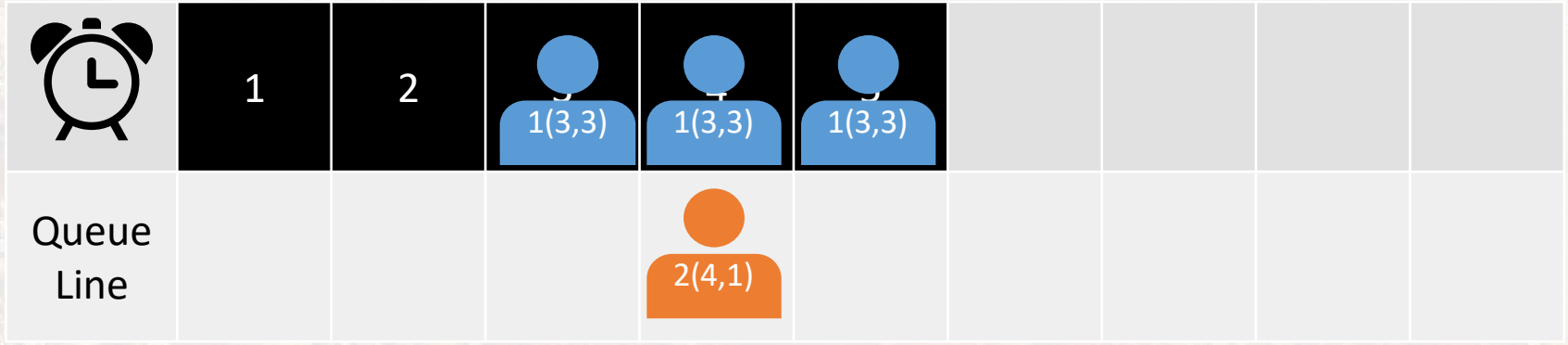


NumberArrivals:1
numberServed:0
totalTimeWaited:0

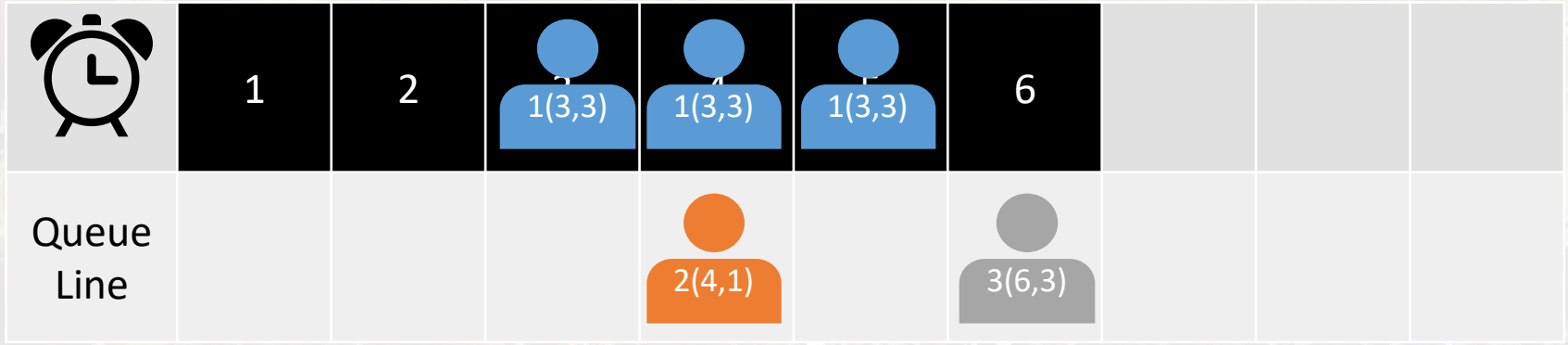
Simulation



NumberArrivals:2
numberServed:0
totalTimeWaited:0

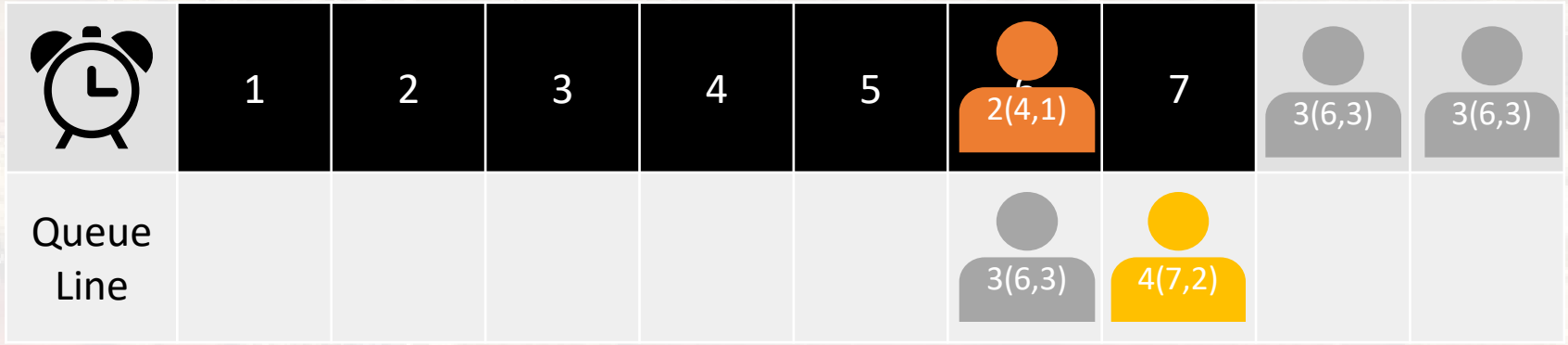


NumberArrivals:2
numberServed:0
totalTimeWaited:0

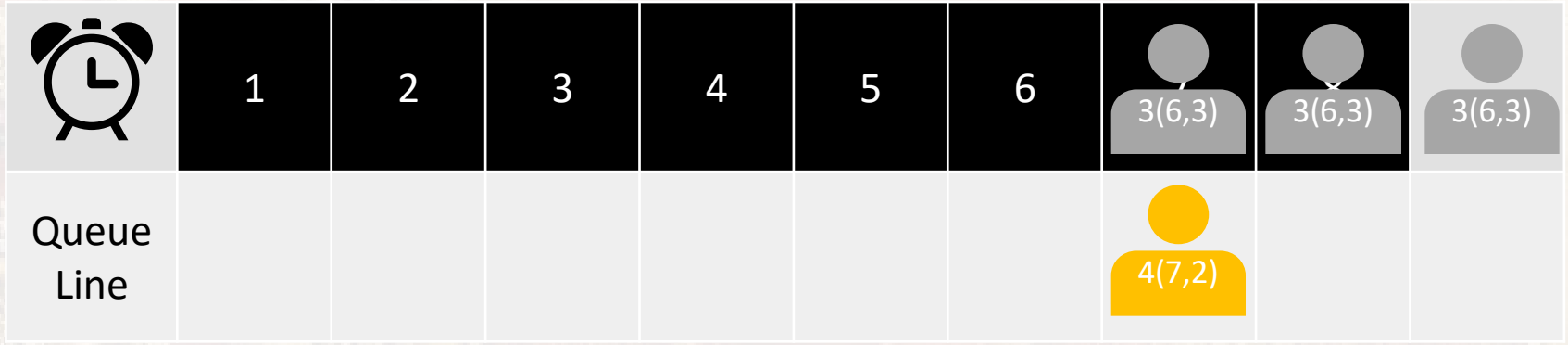


NumberArrivals:3
numberServed:1
totalTimeWaited:2

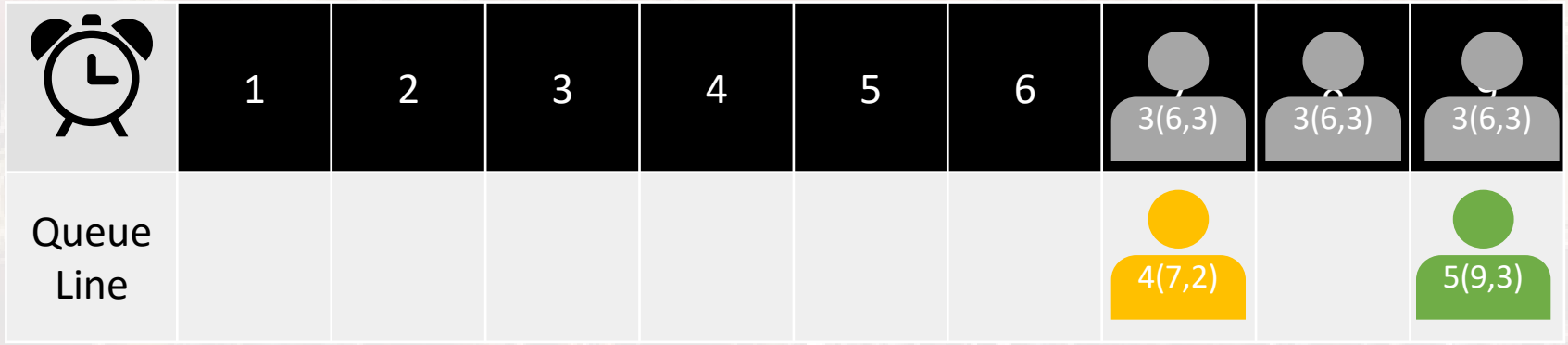
Simulation



NumberArrivals:4
numberServed:2
totalTimeWaited:3



NumberArrivals:4
numberServed:2
totalTimeWaited:3



NumberArrivals:5
numberServed:2
totalTimeWaited:3

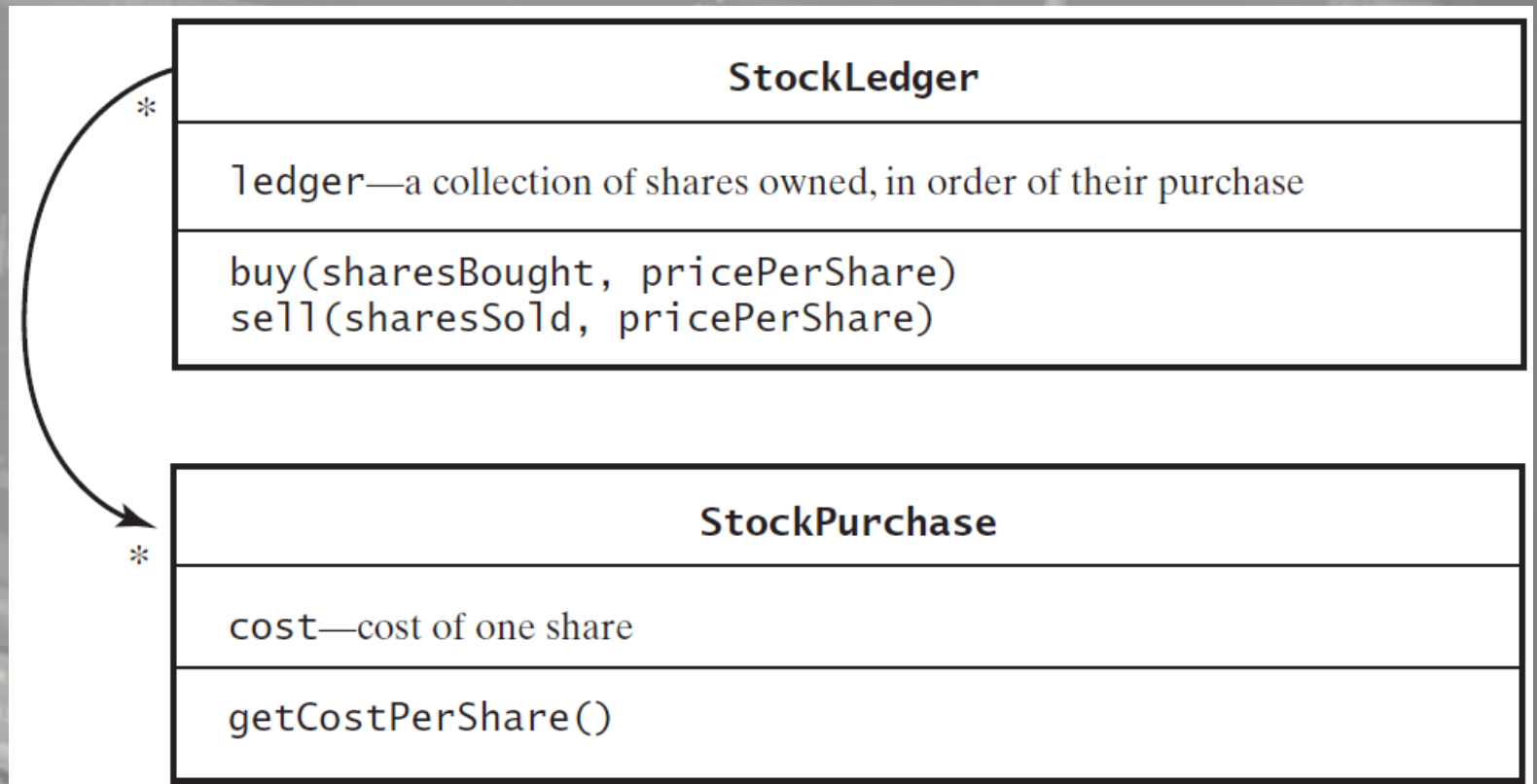
Example2-Computing the Capital Gain in a Sale of Stock

- Capital gain: a profit that you have made if the sale price exceeds the purchase price
  - stock sales are a first-in, first-out application
- Example: Presto Pizza
  - Last year: buy 6/$45
  - Last month; buy 6/$75
  - Today: sell 9/$65

6/$65-$45	=6*$20	=$120
3/$65-$75	=3*-$10	=-$30
		$90

Design of Stock

- Respoisibility of StockLedger:
  - Record the shares of a stock purchased, in chronological order
  - Remove the shares of a stock sold, beginning with the ones held the longest
  - Compute the capital gain (loss) on shares of a stock sold



Example2-Computing the Capital Gain in a Sale of Stock

```java
public class StockMain {
    public static void main(String[] args) {
        StockLedger book = new StockLedger();
        book.buy(6,45);
        book.buy(6,75);
        System.out.println("My capital gain is: "+book.sell(9, 65));
    }
}

```

Problems Javadoc Declaration Console

<terminated> StockMain [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2020. 10. 6. 오전 2:19:49 – 오전 2:19:51)

```
My capital gain is: 90.0
```

buy

Client program

```
book.buy(6,45);
```

StockLodger

```
public void buy(int sharesBought, int pricePerShare){
        for (; sharesBought > 0; sharesBought--){
                ledger.offer(new StockPurchase(pricePerShare));
        }
}
```

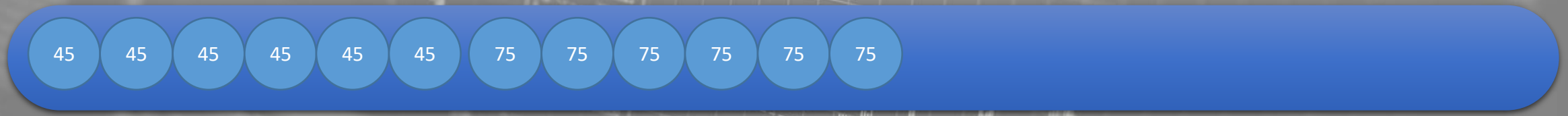
buy

Client program

```
book.buy(6,75);
```

StockLodger

```
public void buy(int sharesBought, int pricePerShare){
        for (; sharesBought > 0; sharesBought--){
                ledger.offer(new StockPurchase(pricePerShare));
        }
}
```



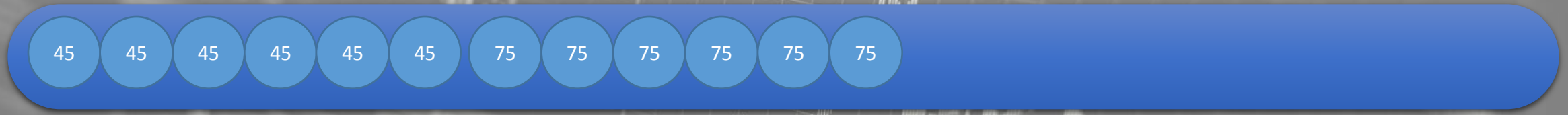
sell

Client program

```
System.out.println("My capital gain is: "+
book.sell(9, 65));
```

StockLodger

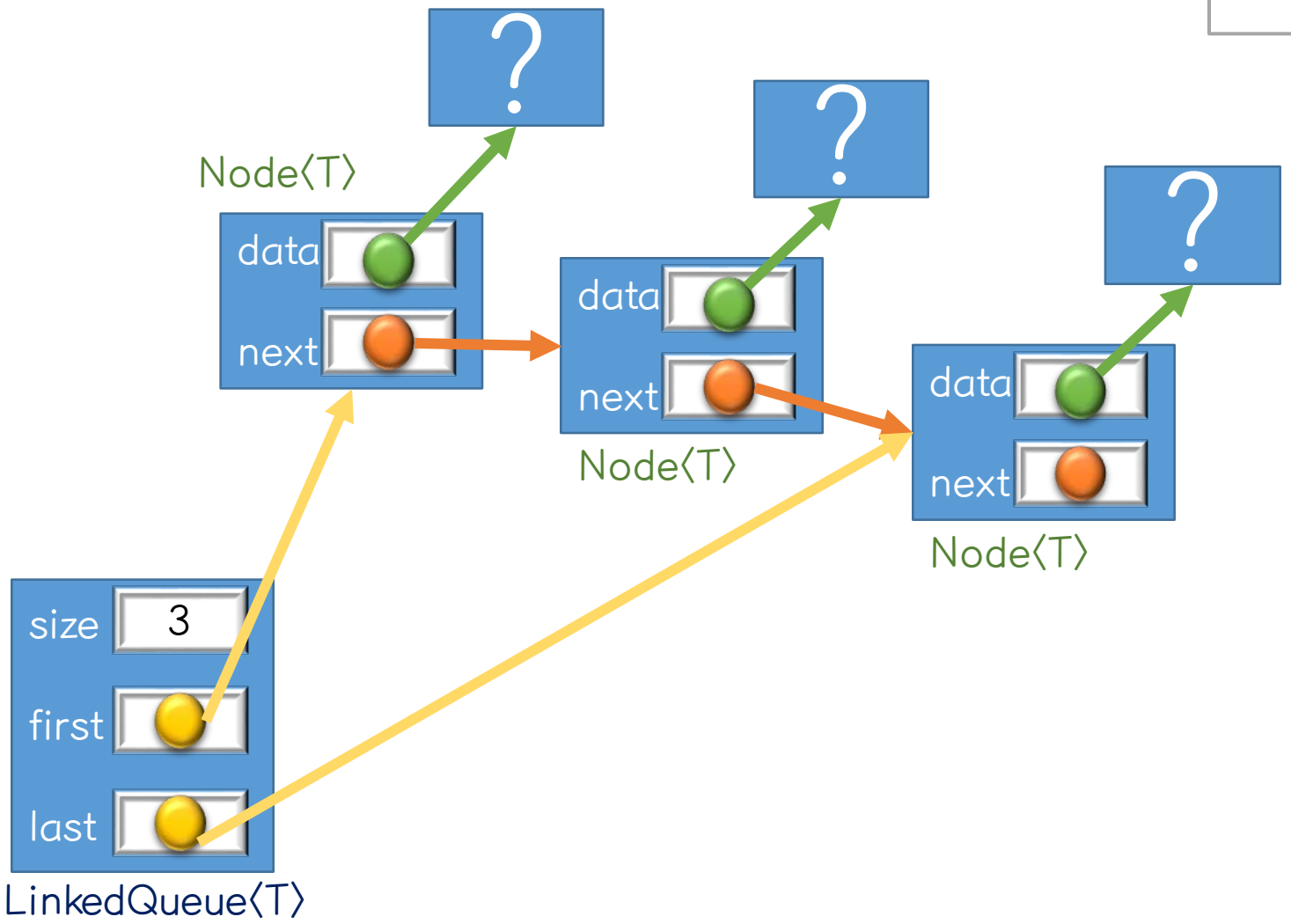
```
public double sell(int sharesSold, int pricePerShare) {
     int saleAmount = sharesSold * pricePerShare;
     int totalCost = 0;
     for (;sharesSold > 0;sharesSold--) {
          StockPurchase share = ledger.poll();
          int shareCost = share.getCostPerShare();
          totalCost = totalCost + shareCost;
     } // end while
          return saleAmount - totalCost;
} // end sell
```



Result:

Linked Queue

Outline of the class



the size of the stack

LinkedQueue<T>

- size: int
- first: Node<T>
- last: Node<T>

+ enqueue(T newItem)
+ dequeue(): T
+ getFirst(): T
+ isEmpty(): Boolean
+clear(): void
+size():int
+toString(): String

Node<T>

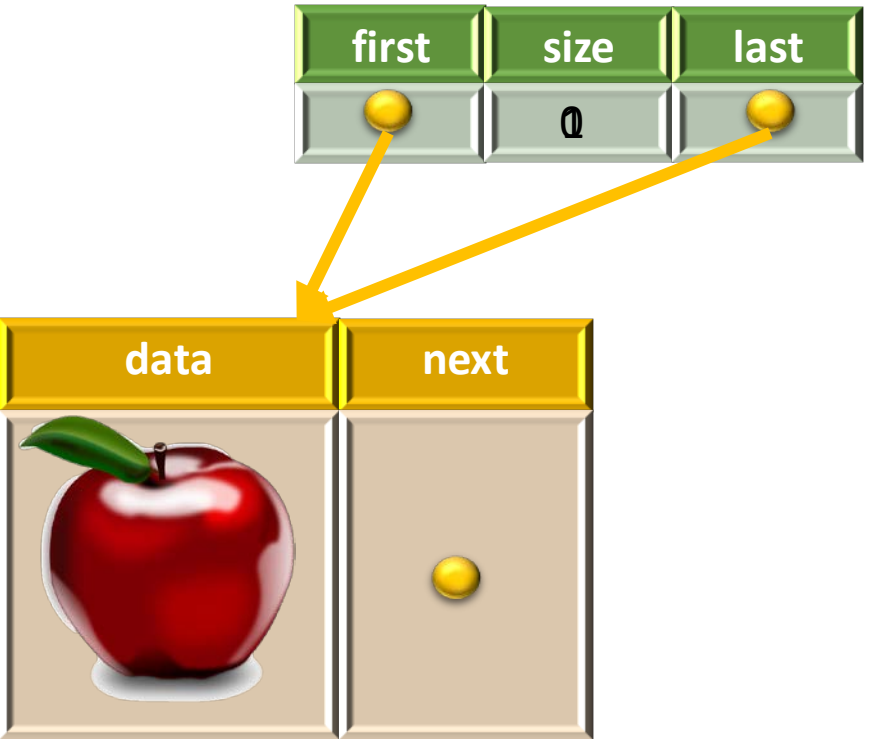
- data: T
- next: Node<T>

+ Node<T>(T data, Node<T> next)

Method enqueue

- Client program

crate.enqueue("Apple");



- In class LinkedQueue

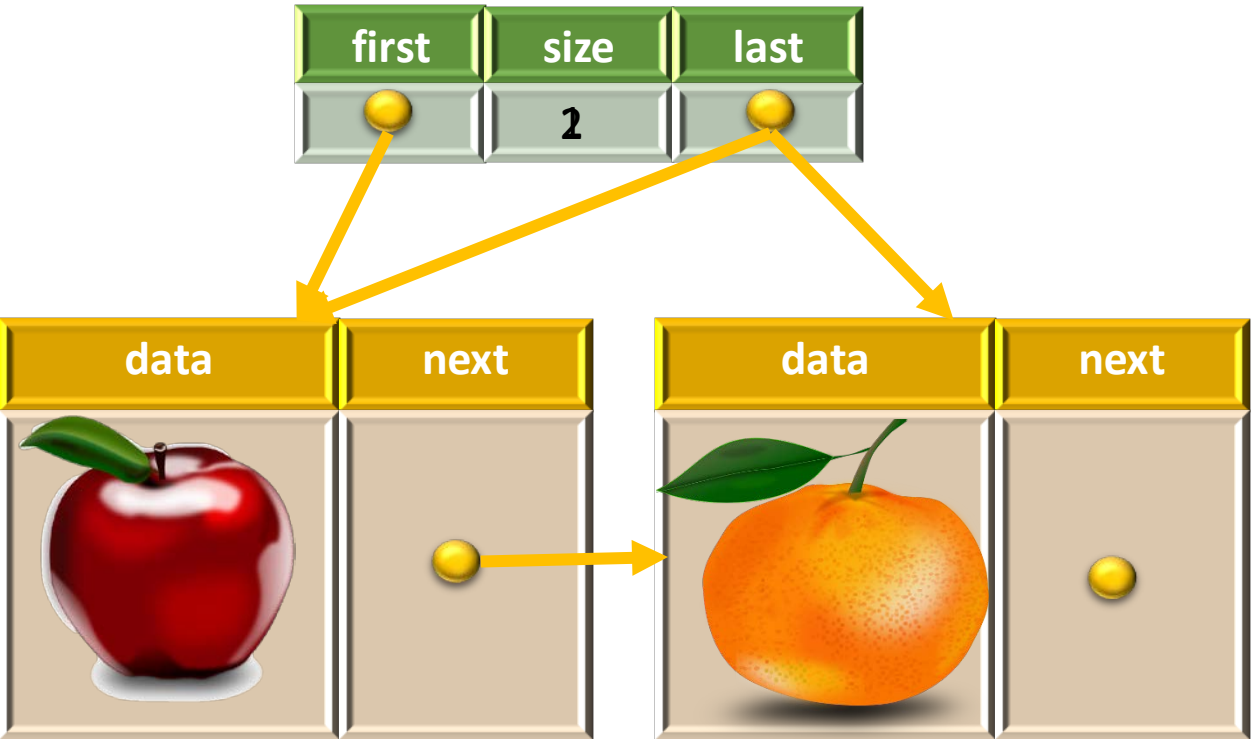
```
public void enqueue(T newEntry) {
     if (isEmpty())
          first = last = new Node(newEntry, null);
     else
          last = last.next = new Node(newEntry, null);
          size++;
}
```

O(1)

Method enqueue

- Client program

crate.enqueue("Orange");



- In class LinkedQueue

```
public void enqueue(T newEntry) {
     if (isEmpty())
          first = last = new Node(newEntry, null);
     else
          last = last.next = new Node(newEntry, null);
          size++;
}
```

O(1)

Method enqueue

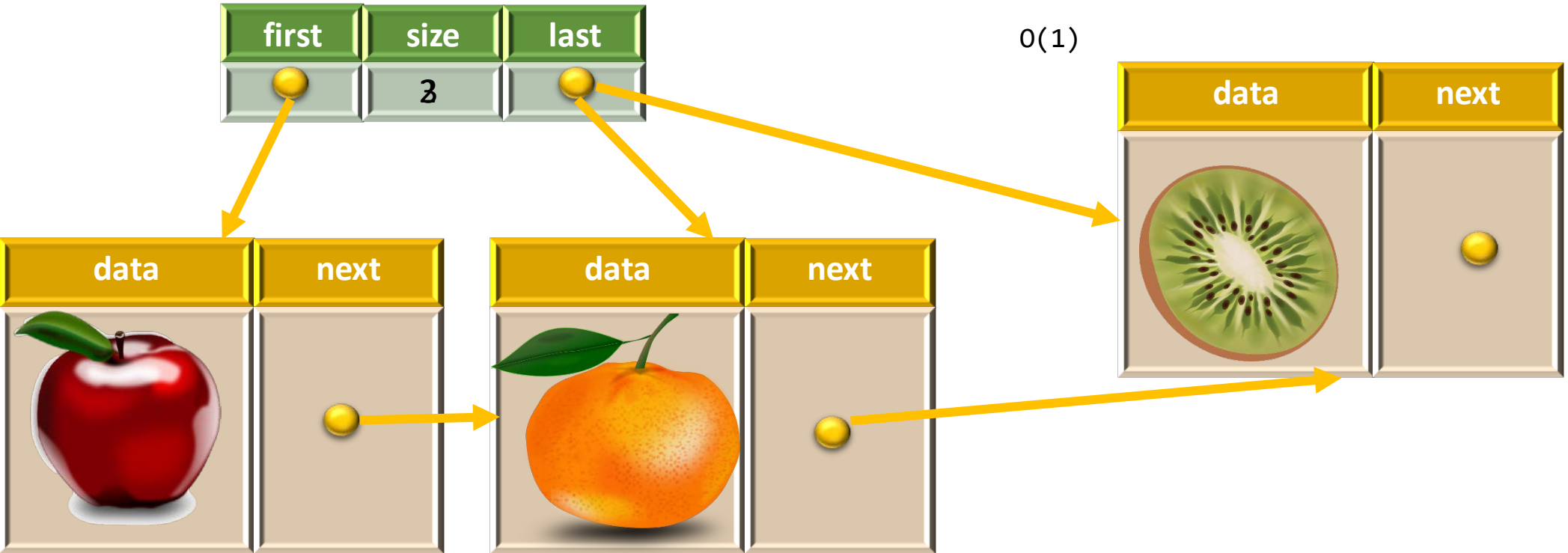
- Client program

crate.enqueue("Kiwi");

- In class LinkedQueue

```
public void enqueue(T newEntry) {
     if (isEmpty())
          first = last = new Node(newEntry, null);
     else
          last = last.next = new Node(newEntry, null);
          size++;
}
```

O(1)



Method dequeue

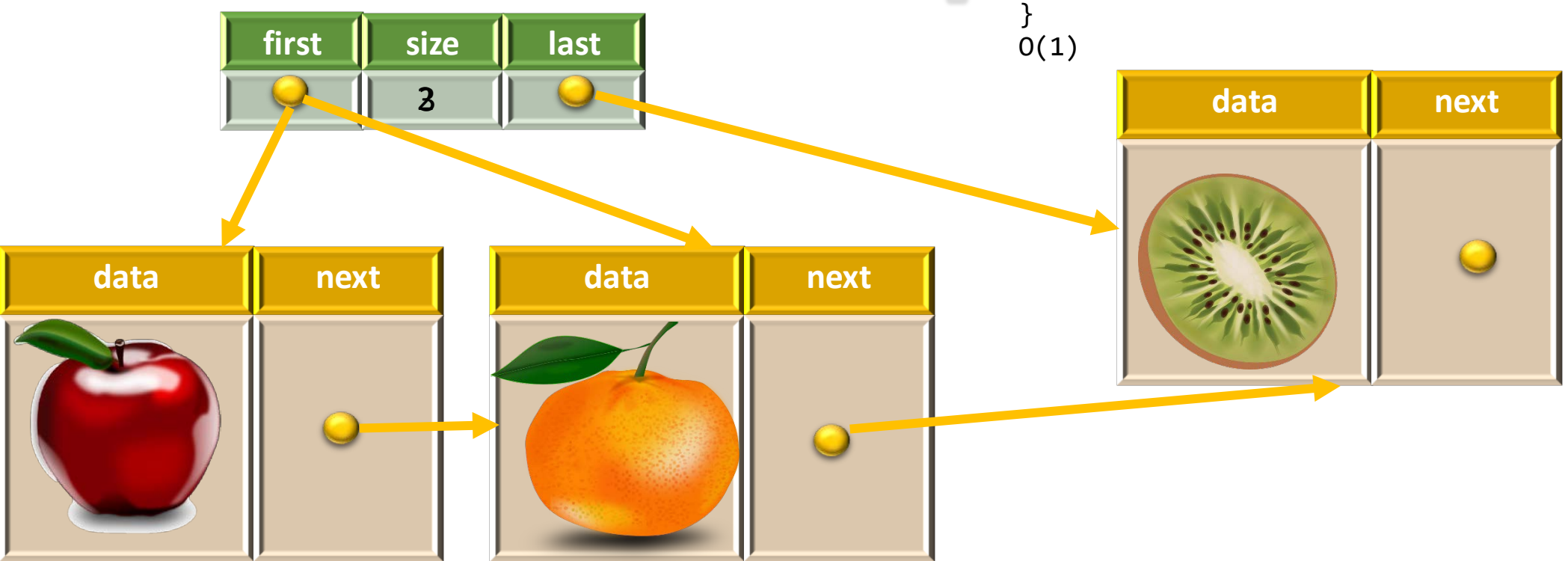
- Client program

System.*out.println("dequeue: "+ crate.dequeue());*

- In class LinkedQueue

```
public T dequeue() {
     if(isEmpty())return null;
     T result = first.data;
     if(first == last) first = last = null;
     else first = first.next;
     size --;
     return result;
}
```

O(1)



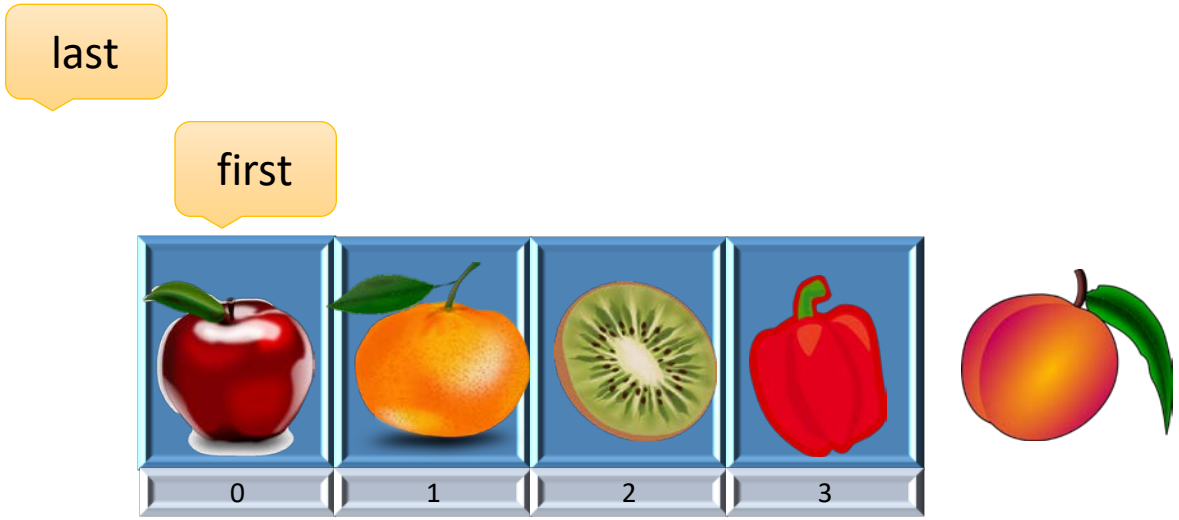
result:

Array Queue

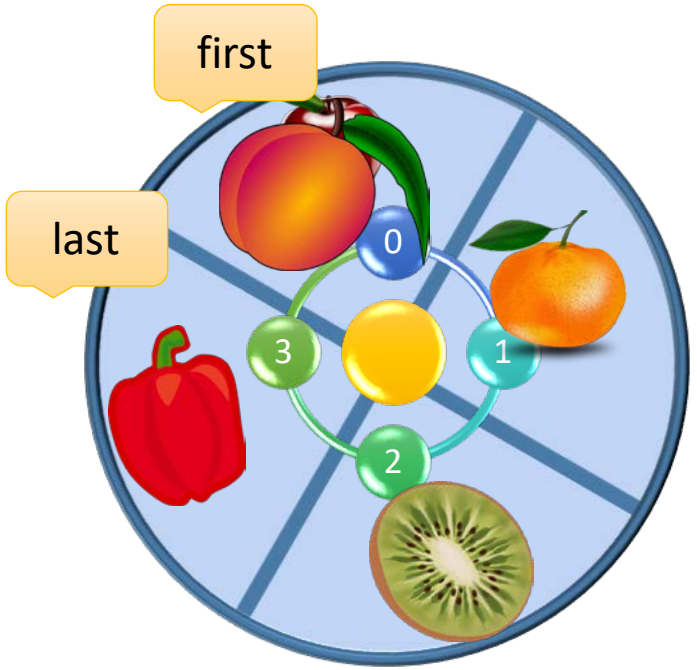
Design Issue: linear queue vs circular queue

- Linear Queue

```
crate.enqueue("Apple");
crate.enqueue("Orange");
crate.enqueue("Kiwi");
System.out.println("dequeue: "+ crate.dequeue());
crate.enqueue("Paprika");
crate.enqueue("Peach");
```



- Circular queue



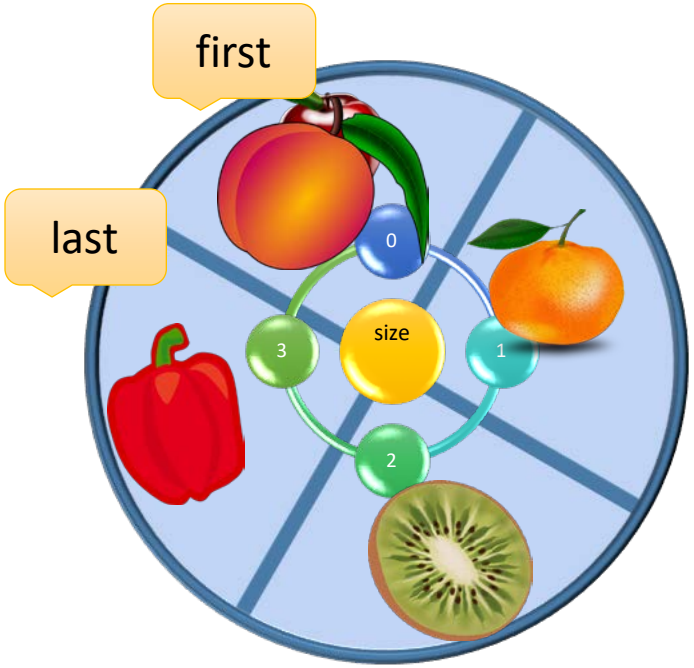
Design Issue: size vs unused element

- Empty:

  (last-first+q.size)% q.size = q.size-1
- Full:

  (last-first +q.size)% q.size = q.size-1

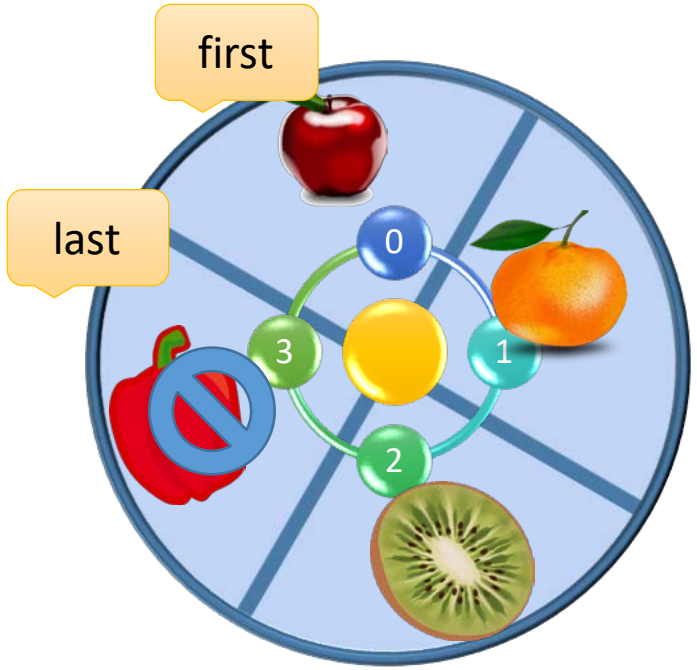
➔Using size variable



- Empty:

  (last-first+q.size)% q.size = q.size-1
- Full:

  (last-first +q.size)% q.size = q.size-2



Outline of an ArrayQueue

the index for the data that will be dequeued
= the index for the data enqueued firstly

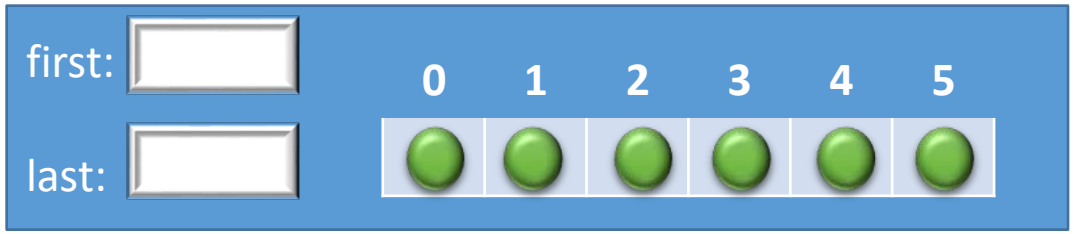
the index for the data enqueued lastly

ArrayStack<T>

- first: int
- last:int
- queue: T[]
- DEFAULT_INITIAL_CAPACITY: int
- integrityOK: boolean
- MAX_CAPACITY: int

the array for the data

+ enqueue(T newItem)
+ dequeue(): T
+ getFirst(): T
+ isEmpty(): Boolean
+clear(): void
+size():int
+toString(): String



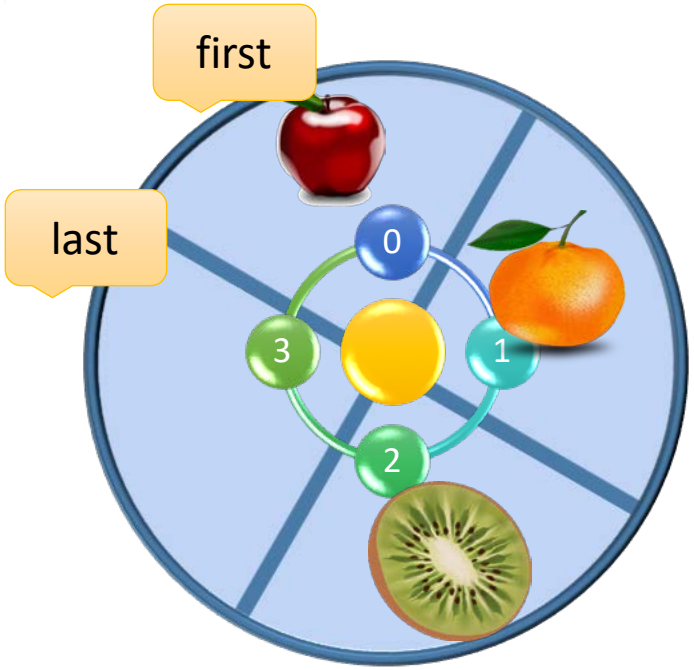
Method enqueue

```
1 public class QueueTest {
2     public static void main(String[] args) {
3         QueueInterface<String> crate = new ArrayQueue<>(3);
4         System.out.println(crate.toString());
5         crate.enqueue("Apple");
6         System.out.println(crate.toString());
7         crate.enqueue("Orange");
8         System.out.println(crate.toString());
9         crate.enqueue("Kiwi");
10        System.out.println(crate.toString());
11    }
12 }
```

Problems Javadoc Declaration Console

<terminated> QueueTest [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2020. 10. 7. 오후 11:0

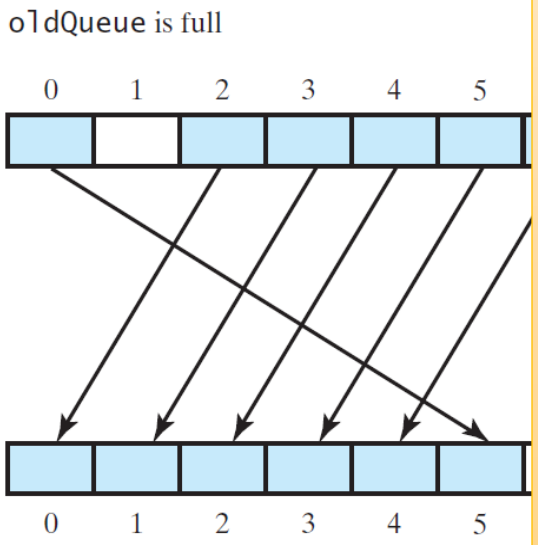
```
Queue[0, 0, 3]: [null, null, null, null]
Queue[1, 0, 0]: [Apple, null, null, null]
Queue[2, 0, 1]: [Apple, Orange, null, null]
Queue[3, 0, 2]: [Apple, Orange, Kiwi, null]
```



- In class stack

```
public void enqueue(T newEntry) {
        checkIntegrity();
        ensureCapacity();
        last = (last + 1) % queue.length;
        queue[last] = newEntry;
}
```

Usually O(1), if a stack is full, O(n)




```
private void ensureCapacity() {
        if(first == (last +2) % queue.length ) {
            T[] oldQ = queue;
            int newSize = 2*oldQ.length;
            integrityOK=false;
            if(newSize-1 < MAX_CAPACITY) {
                @SuppressWarnings("unchecked")
                T[] tempQueue = (T[]) new Object[newSize];
                queue = tempQueue;
                for(int i = 0 ; i < oldQ.length -1 ; i++) {
                    queue[i]=oldQ[first];
                    first = (first+1)%oldQ.length;
                }
                first = 0;
                last = oldQ.length -2;
                integrityOK = true;
            } else
                throw new IllegalStateException("Attempt to create a queue whose "
                        + "capacity exceeds allowed maximum.");
        }
    }
```

Method dequeue

Queue/src/LinkedQueue.java

```
2   public static void main(String[] args) {
3           QueueInterface<String> crate = new ArrayQueue<>(3);
4           System.out.println(crate.toString());
5           crate.enqueue("Apple");
6           System.out.println(crate.toString());
7           crate.enqueue("Orange");
8           System.out.println(crate.toString());
9           crate.enqueue("Kiwi");
10          System.out.println(crate.toString());
11          System.out.println("dequeue: "+ crate.dequeue());
12          System.out.println("dequeue: "+ crate.dequeue());
13      }
14  }
15
```

Problems Javadoc Declaration Console

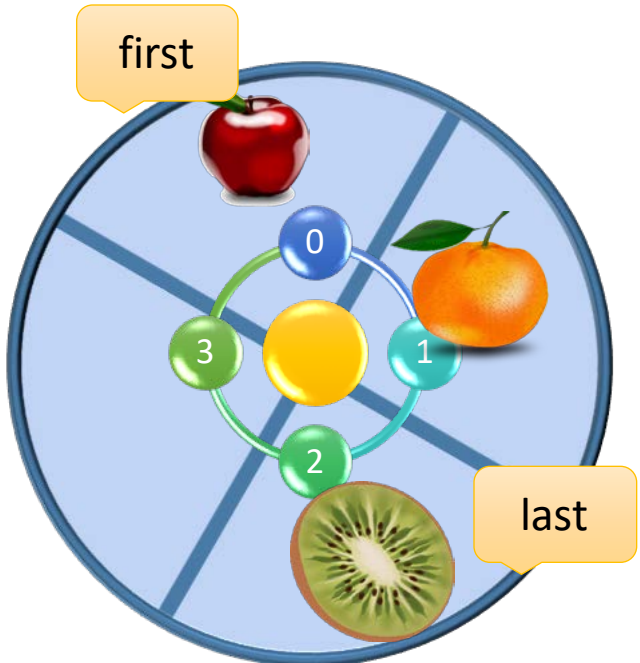
<terminated> QueueTest [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2020. 10. 7. 오후 11:0

```
Queue[0, 0, 3]: [null, null, null, null]
Queue[1, 0, 0]: [Apple, null, null, null]
Queue[2, 0, 1]: [Apple, Orange, null, null]
Queue[3, 0, 2]: [Apple, Orange, Kiwi, null]
dequeue: Apple
dequeue: Orange
```

- In class stack

```
public T dequeue() {
    checkIntegrity();
    if(isEmpty()) return null;
    else {
        T result = queue[first];
        queue[first]=null;
        first = (first+1)% queue.length;
        return result;
    }
}
```

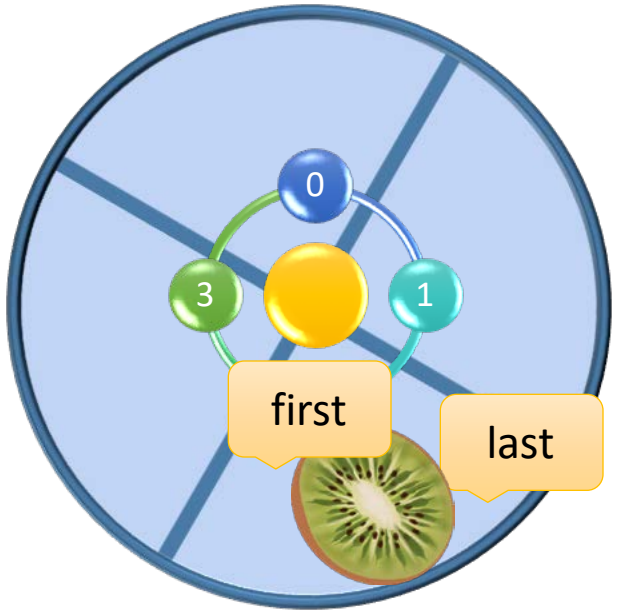
O(1)



Method size and getFront

- Client program

```
System.out.println("size: "+ crate.size()+
        ", getFront: "+ crate.getFront());
```



- In class stack

```
public T getFront() {
    checkIntegrity();
    if(isEmpty()) return null;
    else {
        T result = queue[first];
        return result;
    }
}
O(1)

public int size() {
    return (last+queue.length+1-first)%queue.length;
}
O(1)
```

Method push

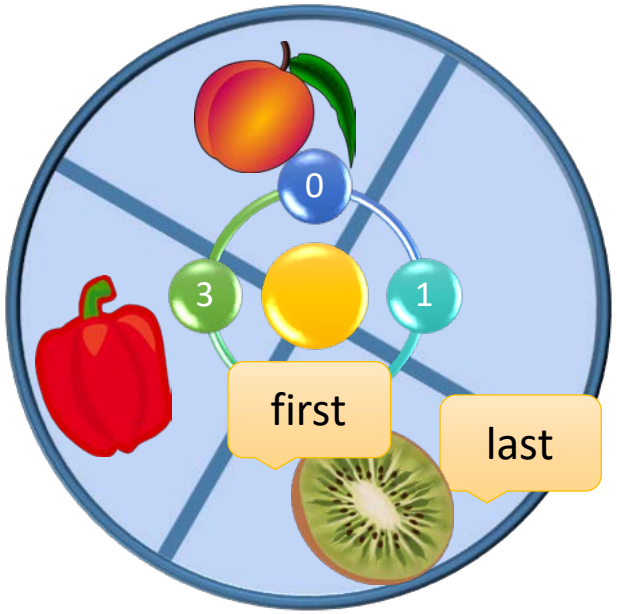
```
14          crate.enqueue("Paprika");
15          crate.enqueue("Peach");
16          System.out.println(crate.toString());
17      }
18  }
```

Problems Javadoc Declaration Console

```
<terminated> QueueTest [Java Application] C:₩Program Files₩Java₩jdk-14.0.2₩bin₩javaw.exe (2
Queue[0, 0, 3]: [null, null, null, null]
Queue[1, 0, 0]: [Apple, null, null, null]
Queue[2, 0, 1]: [Apple, Orange, null, null]
Queue[3, 0, 2]: [Apple, Orange, Kiwi, null]
dequeue: Apple
dequeue: Orange
size: 1, getFront: Kiwi
Queue[3, 2, 0]: [Peach, null, Kiwi, Paprika]
```

- In class stack

```
public void enqueue(T newEntry) {
        checkIntegrity();
        ensureCapacity();
        last = (last + 1) % queue.length;
        queue[last] = newEntry;
}
```



Thank you