CS302 - Data Structures *using C++*

Topic: Queues and Priority Queues

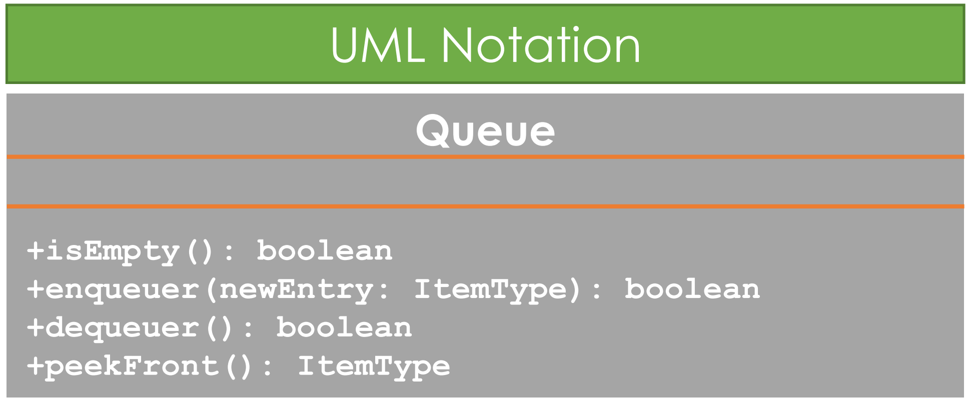
Kostas Alexis

The ADT Queue

- Like a line of people
    - First person in line is first person served
    - New element of queue enter at its back
    - Items leave the queue from its front
- Called FIFI behavior
    - First In First Out

The ADT Queue

- UML diagram for the ADT queue


UML Notation

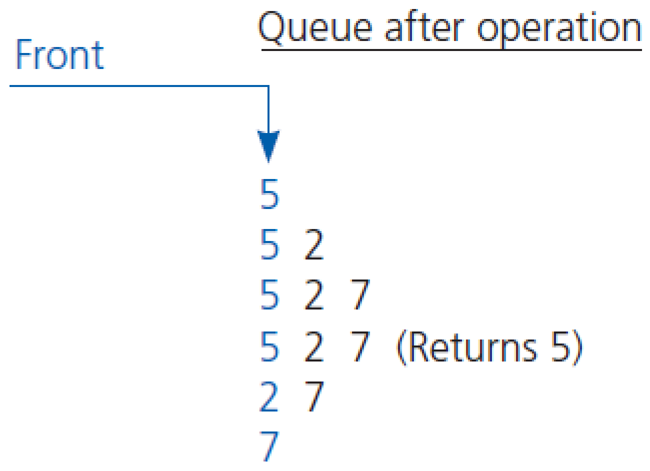
Queue

```
+isEmpty(): boolean
+enqueuer(newEntry: ItemType): boolean
+dequeuer(): boolean
+peekFront(): ItemType
```

The ADT Queue

- Example queue operations

```
aQueue = an empty queue
aQueue.enqueue(5)
aQueue.enqueue(2)
aQueue.enqueue(7)
aQueue.peekFront(5)
aQueue.dequeue()
aQueue.dequeue()
```



The ADT Queue

- C++ Interface for queues

```
#ifndef QUEUE_INTERFACE_
#define QUEUE_INTERFACE _

template<class ItemType>
class QueueInterface
{
public:
      virtual bool isEmpty() const = 0;
      virtual bool enqueue(const ItemType& anEntry) = 0;
      virtual bool dequeue() = 0;
      virtual ItemType dequeue() const 0;

      virtual ~QueueInterface() { }
}; // end QueueInterface
#endif
```

Applications Reading a String of Chars

- Pseudocode to read a string of characters into a queue

```
// Reade a string of characters from a single line of input into a queue
aQueue = a new empty queue
while(not end of line)
{
    Read a new character into ch
    aQueue.eqnueue(ch)
}
```

Applications Recognizing a Palindrome

- **Question:** How can we detect that a certain string of characters is a palindrome?
- Examples of palindromes
  - Anna
  - Civic
  - Kayak
  - Level
  - Madam
  - Mom
  - Noon
  - Racecar

The ADT Priority Queue

- Organize data by priorities
  - Example: weekly “to-do” list
- Priority value
  - We will say high value – high priority
- Operations
  - Test for empty
  - Add to queue in sorted position
  - Remove/get entry with highest priority

The ADT Priority Queue

- UML diagram for the class PriorityQueue

UML Notation

PriorityQueue

```
+isEmpty(): boolean
+enqueuer(newEntry: ItemType): boolean
+dequeuer(): boolean
+peekFront(): ItemType
```

Tracking Your Assignments

- UML diagram for the class Assignment

UML Notation

Assignment

```
Course – the course code
Task – a description of the assignment
Date – the due data
```

```
+getCourseCode(): string
+getTask(): string
+getDueDate(): string
```

Tracking Your Assignments

- Pseudocode organize assignments, responsibilities

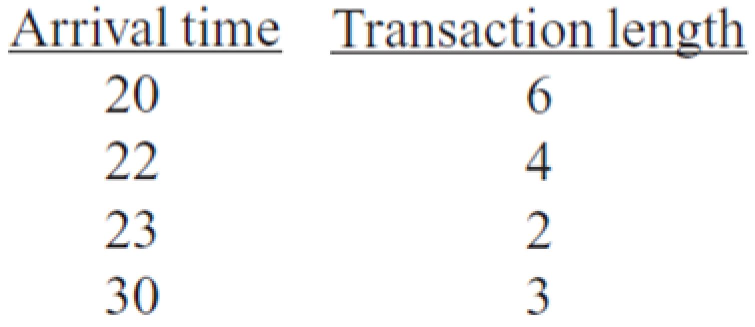
```
assignmentLog = a new priority queue using due date as the priority value
project = a new instance of Assignment
essay = a new instance of Assignment
quiz = a new instance of Assignment
errand = a new instance of Assignment
assignmentLog.enqueue(project)
assignmentLog.enqueue(essay)
assignmentLog.enqueue(quiz)
assignmentLog.enqueue(errand)
cout << "I should the following first: "
cout << assignmentLog.peekFront()
```

Application: Simulation

- Simulation models behavior of systems
- Problem to solve
  - Approximate average time bank customer must wait for service from a teller
  - Decrease in customer wait time with each new teller added

Application: Simulation

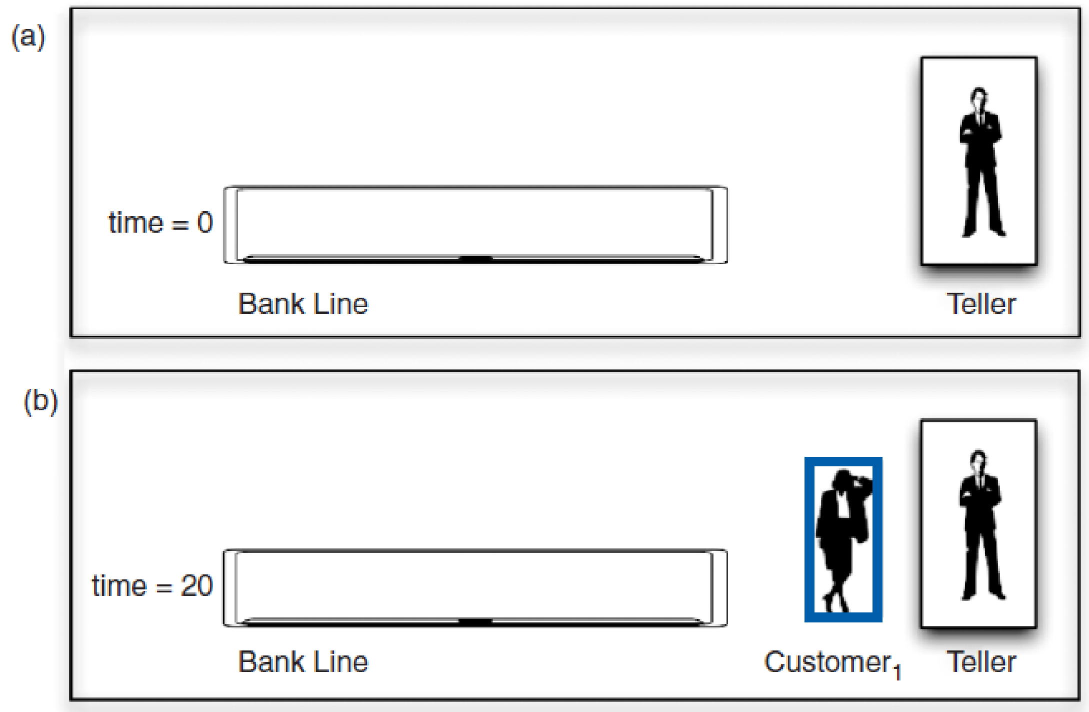
- Sample arrival and transaction times



Arrival time	Transaction length
20	6
22	4
23	2
30	3

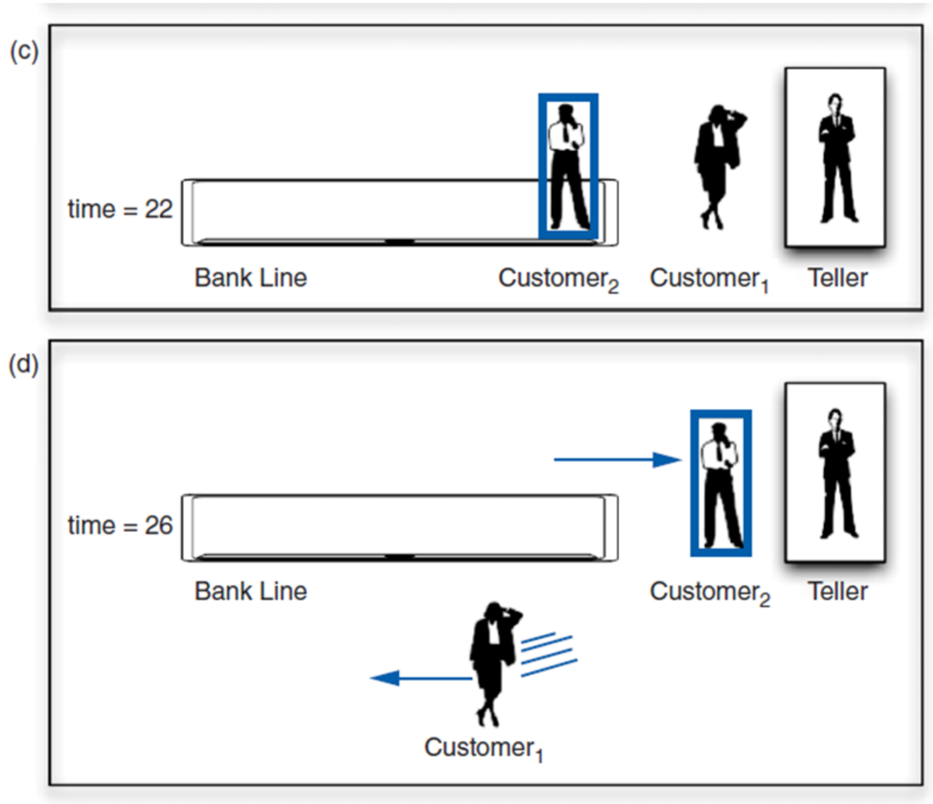
Application: Simulation

- A bank line at time (a) 0; (b) 20; (c) 22; (d) 26



Application: Simulation

- A bank line at time (a) 0; (b) 20; (c) 22; (d) 26



Application: Simulation

- Pseudocode for an event loop

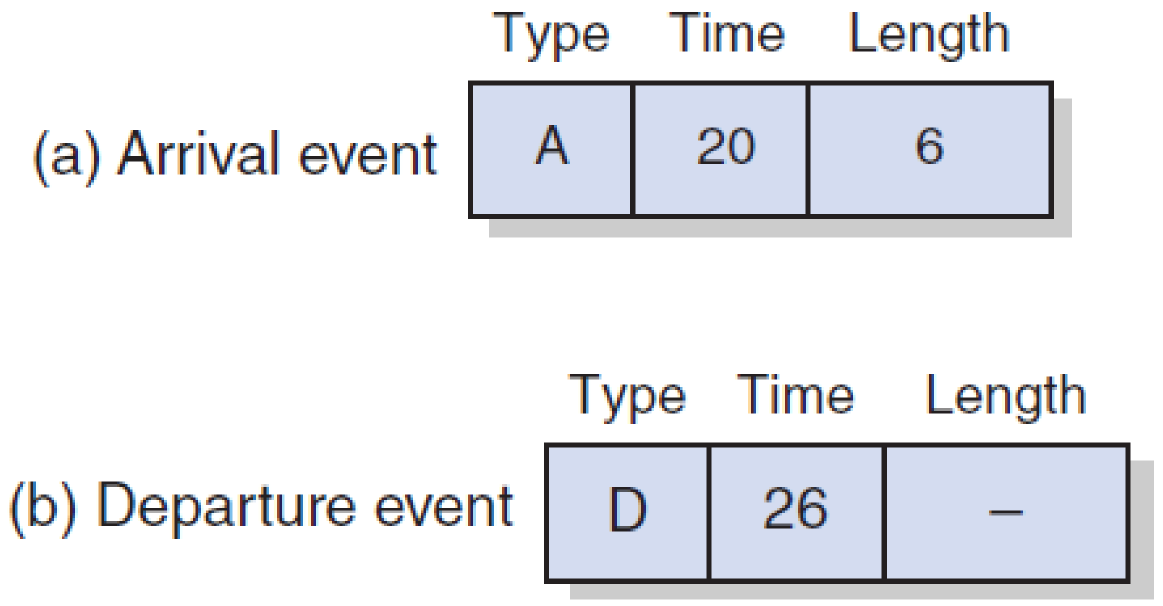
```
Initialize the line to “no customers”
while (events remain to be processed)
{
    currentTime = time of next event
    if (event is an arrival event)
        Process the arrival event
    else
        Process the departure event

    // When an arrival event and a departure event occur at the same time,
    // arbitrary process the arrival event first
}
```

Application: Simulation

- Time-driven simulation
    - Simulates the ticking of a clock
- Event-driven simulation considers
    - Only the times of certain events
    - In this case, arrival(s) and departure(s)
- Event list contains
    - All future arrival and departure events

Application: Simulation

- A typical instance of (a) an arrival event; (b) a departure event



Application: Simulation

- Two tasks required to process each event
  - Update the bank line: Add or remove customers
  - Update the event queue: Add or remove events
- New customer
  - Always enters bank line
  - Served while at the front of the line

Position-Oriented and Value-Oriented ADTs

- Position-oriented ADTs
  - Stack, list, queue
- Value-oriented ADTs
  - Sorted list

Position-Oriented and Value-Oriented ADTs

- Comparison of stack and queue operations
  - **isEmpty** for both
  - **Pop** and **dequeue**
  - **Peek** and **peekFront**

Position-Oriented and Value-Oriented ADTs

- ADT list operations generalize stack and queue operations
  - **getLength**
  - **insert**
  - **remove**
  - **getEntry**

Thank you