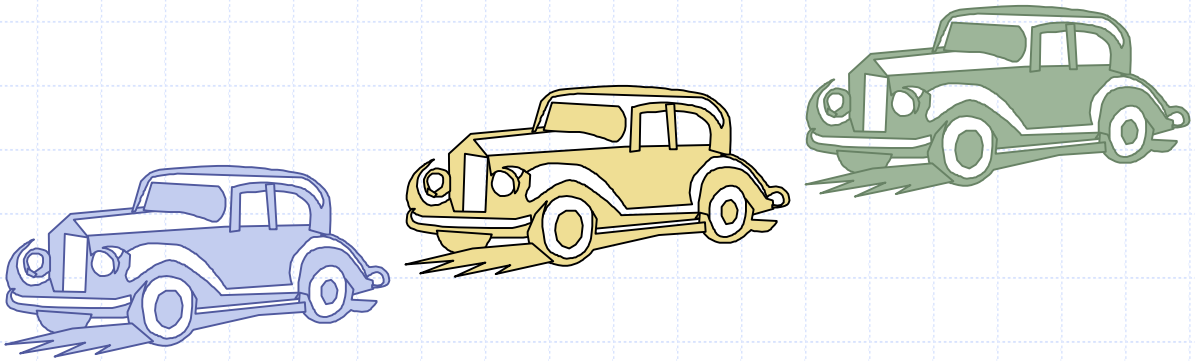
Queues



Queue ADT (§4.3)

- Queue ADT speichert bel. Objekte
- Einfügen und Löschen gemäß first-in first-out
- Einfügen am Ende
- Löschen am Anfang
- Hauptoperationen auf Queue:
  - enqueue(object): Einfügen eines Elementes am Ende der Queue
  - object dequeue(): löschen und Rückgabe des Elementes am Anfang der Queue
- Hilfsoperationen auf Queue:
  - object front(): Rückgabe des Elementes am Kopf der Queue (ohne Löschen)
  - integer size(): Rückgabe der Anzahl gespeicherter Elemente
  - boolean isEmpty(): Indikator, ob keine Elemente gespeichert sind
- Exceptions
  - Versuch eines dequeue or front auf einer leeren Queue erzeugt EmptyQueueException

Queue Example

Operation	*Output*	*Q*
enqueue(5)	–	(5)
enqueue(3)	–	(5 3)
dequeue()	*5*	(3)
enqueue(7)	–	(3 7)
dequeue()	*3*	(7)
front()	*7*	(7)
dequeue()	*7*	()
dequeue()	*"error"*	()
isEmpty()	*true*	()
enqueue(9)	–	(9)
enqueue(7)	–	(9 7)
size()	*2*	(9 7)
enqueue(3)	–	(9 7 3)
enqueue(5)	–	(9 7 3 5)
dequeue()	*9*	(7 3 5)

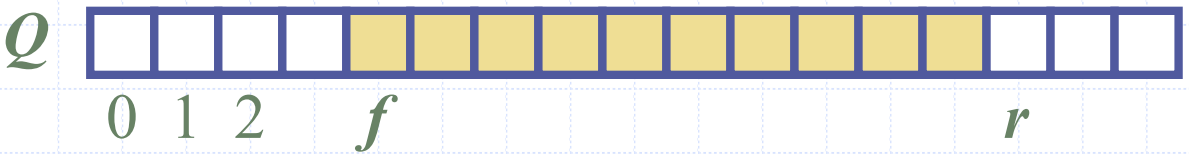
Anwendungen von Queues

- Direkte Anwendungen
  - Warteschlangen, Bürokratie
  - Zugriff auf geteilte Resourcen (z.B., Drucker)
  - Multiprogramming
- Indirekte Anwendungen
  - Hilfsdatenstrukturen für Algorithmen
  - Bestandteil anderer Datenstrukturen

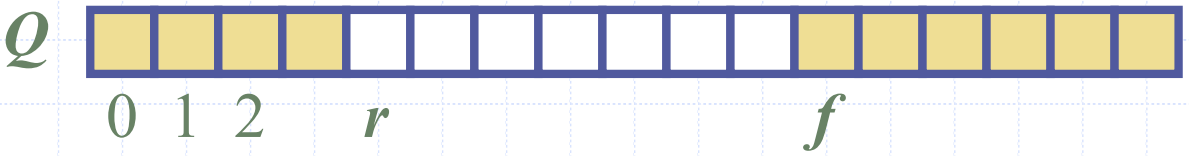
Array-basierte Queue

- Nutze array der Größe N in circulärem Modus
- Zwei Variablen zum Merken des Anfangs und Endes
  - f Index des ersten Elementes
  - r Index direkt nach letztem Element (erste freie Pos.)
- Arrayposition r is immer leer!

Normale Konfiguration



wrapped-around Konfiguration

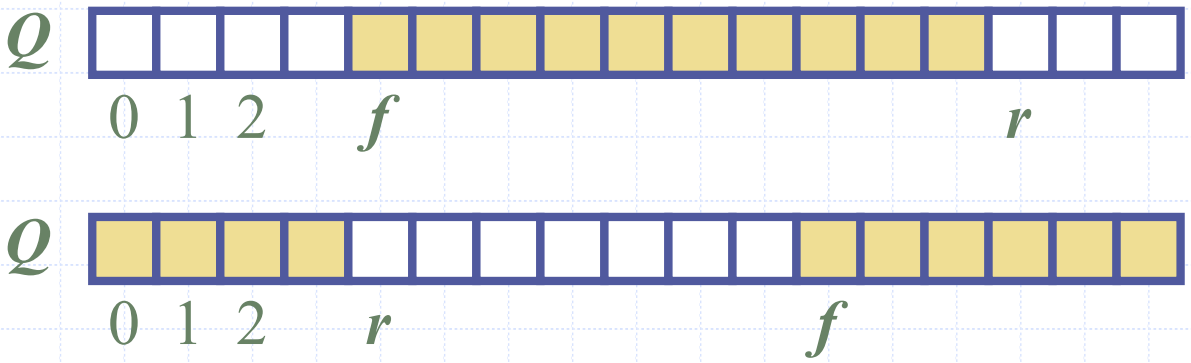


Queue Operationen

- Modulo-Operator nutzen (Divisionsrest)

Algorithm *size*()
  return $(N - f + r) \bmod N$

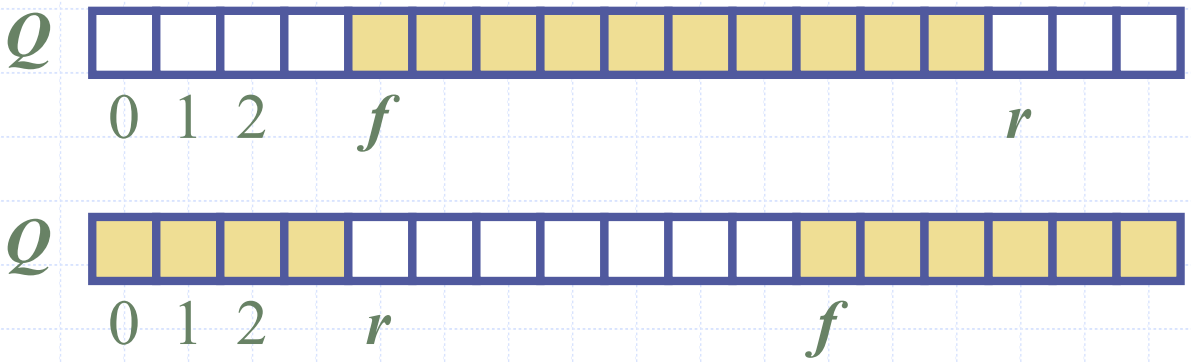
Algorithm *isEmpty*()
  return $(f = r)$



Queue Operationen II

- Operation enqueue erzeugt exception, wenn das array voll ist
- Diese exception is implementatierungs-abhängig

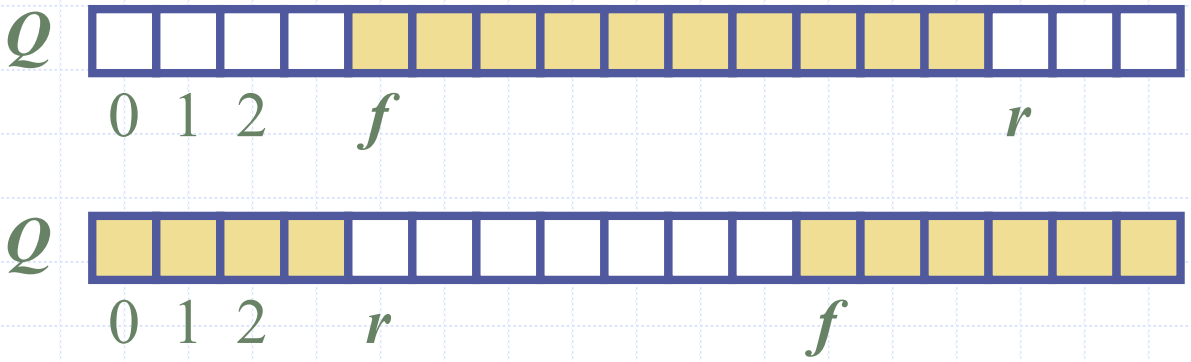
Algorithm *enqueue*(*o*)
  if *size*() = *N* – 1 **then**
    throw *FullQueueException*
  else
    Q[*r*] ← *o*
    r ← (*r* + 1) mod *N*



Queue Operationen III

- Operation dequeue erzeugt exception, falls queue leer
- Diese exception im queue ADT spezifiziert

```
Algorithm dequeue()
  if isEmpty() then
    throw EmptyQueueException
  else
    o ← Q[f]
    f ← (f + 1) mod N
    return o
```



Queue Interface in Java

- Java interface zum obigen Queue ADT
- Erfordert die Definition der class EmptyQueueException
- Nicht übereinstimmend mit der built-in Java class!

```
public interface Queue {

   public int size();

   public boolean isEmpty();

   public Object front()
         throws EmptyQueueException;

   public void enqueue(Object o);

   public Object dequeue()
          throws EmptyQueueException;
}
```

Anwendungen: Round Robin Schedulers

- Implementierung eines *round robin scheduler* über eine Queue *Q* durch die wiederholte Ausführung der folgenden Schritte:
  1. *e* = *Q*.dequeue()
  2. Service element *e*
  3. *Q*.enqueue(*e*)