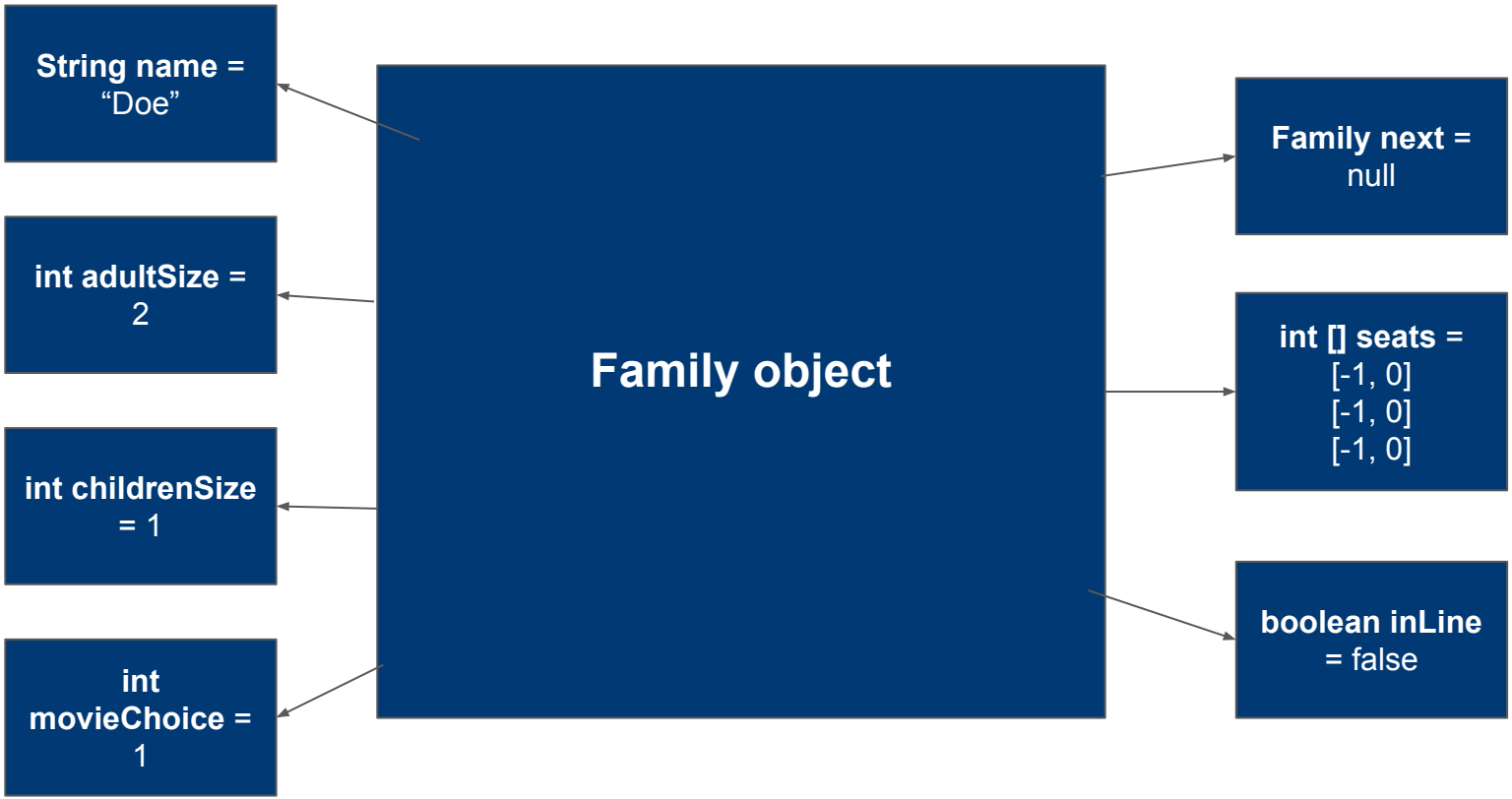
VILLAPLEX THEATERS

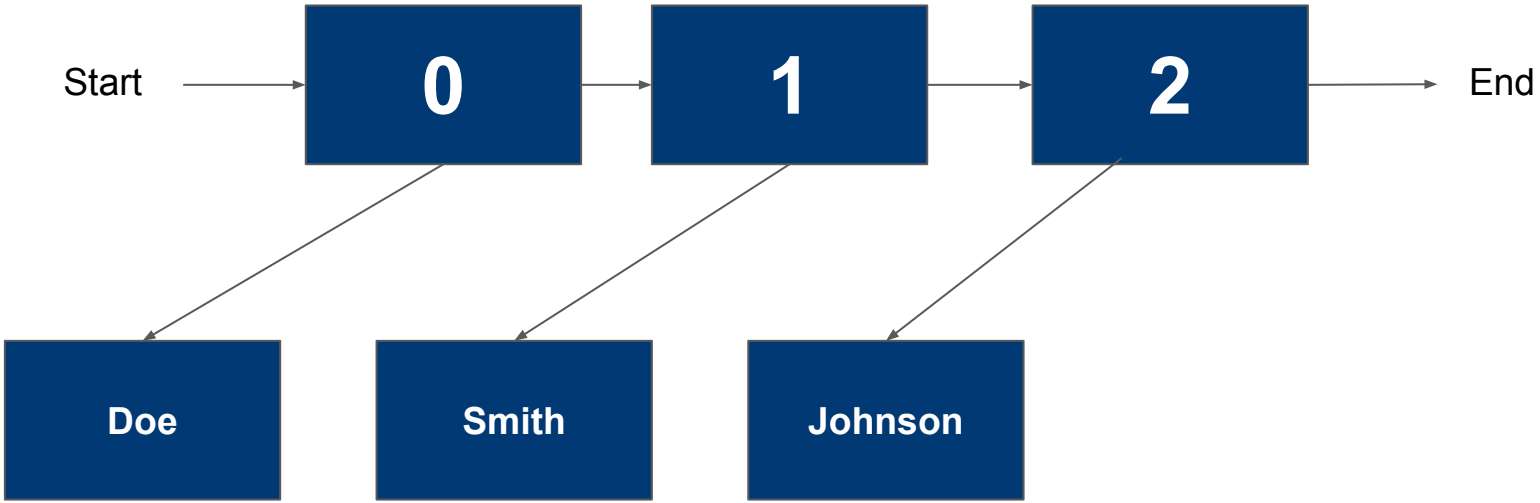
Rina Do, Tim Fan, Maria Fanelle

SYSTEM DESIGN

IDEA	DATA STRUCTURE	EFFICIENCY (TRAVERSING)	SPACE	TOTAL QUANTITY
RegTicket Line	Queue	Worst: O(n) Best: O(1)	Unlimited	3
Receipts	Stack	Worst: O(n) Best: O(1)	Unlimited	1
Family	Object			n
ArrayList	ArrayList	Worst: O(n) Best: O(1)	Unlimited	1
Theater	Matrices	Worst: O(n) Best: O(1)	n x n (as designated by user)	2


String name = "Doe"
int adultSize = 2
int childrenSize = 1
int movieChoice = 1
Family object
Family next = null
int [] seats = [-1, 0] [-1, 0] [-1, 0]
boolean inLine = false

ARRAYLIST<Family>



REGTICKETLINE.JAVA (LINK-BASED QUEUE)

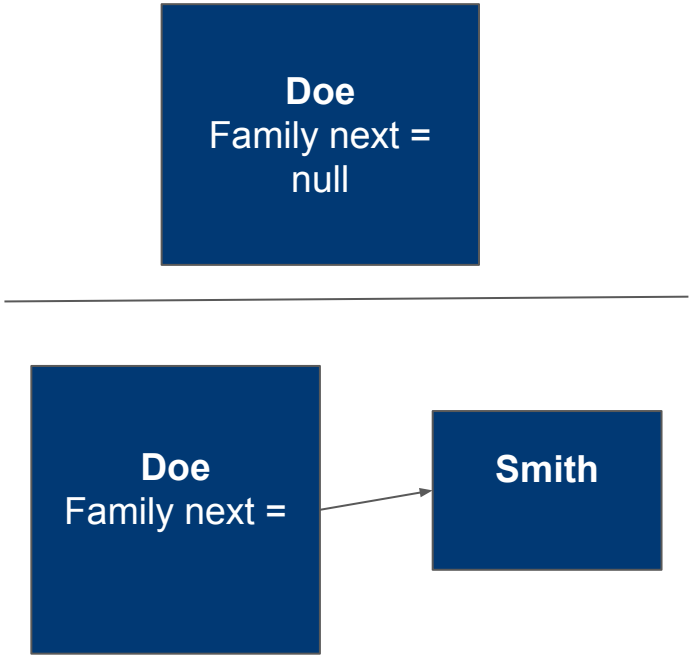
METHODS	RETURN TYPE
enqueue(Family family)	null
dequeue()	Family
isEmpty()	Boolean
size()	int
printQueue()	void

REGTICKETLINE.JAVA

```
private Family first;
private Family last;
private int n;

public
RegTicketLine()
{
     first = null;
     last = null;
     n = 0;
}
```

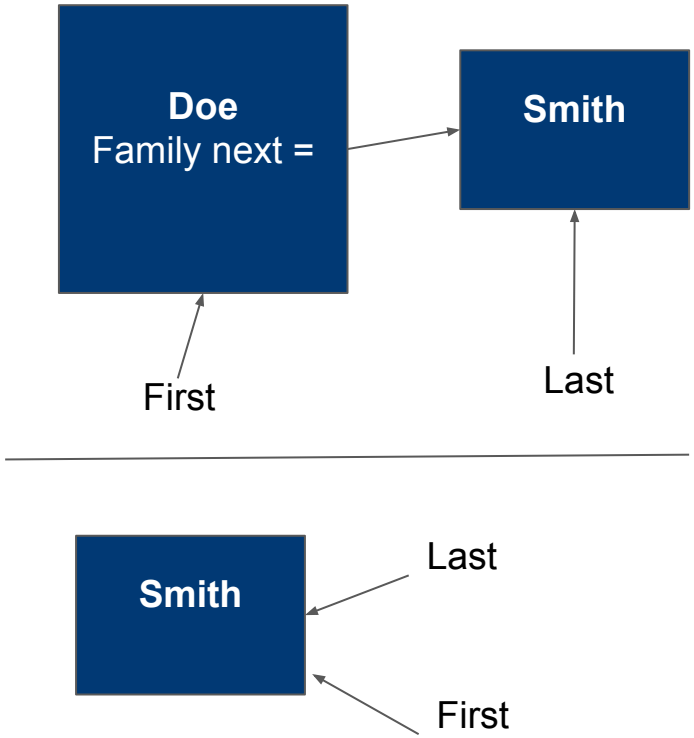
REGTICKETLINE.JAVA: enqueue(Family family)




```
public void enqueue(Family
family)
{
    if (last == null)
     first = family;
    else
    {
       Family temp = last;
      temp.setLink(family);
    }

    last = family;
    n++;
```

REGTICKETLINE.JAVA: dequeue()




```
public Family dequeue()
{
     Family temp = first;

     if (!isEmpty())
     {
          first = temp.getLink();
          if (first == null)
               last = null;
          n--;
     }
     return temp;
}
```

REGTICKETLINE.JAVA: isEmpty(), size(), peek()

```
public boolean isEmpty()
 {
      return first == null;
 }

 public int size()
 {
      return n;
 }

 public Family peek()
 {
      return first;
 }
```

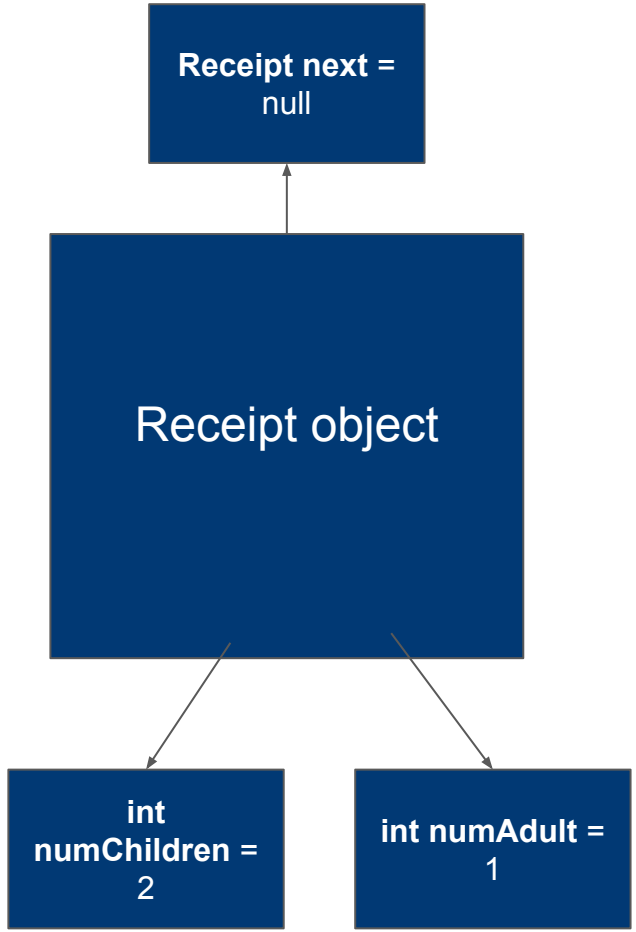
REGTICKETLINE.JAVA: printQueue

```
    public void printQueue()
     {
          Family temp = first;

          while (temp != null)
          {
               System.out.println("Name: " +
temp.getName() + ", Number of adults: " +
temp.getAdultSize() + ", Number of
children: " + temp.getChildrenSize());
               temp = temp.getLink();
          }
     }
```

RECEIPTS.JAVA: the Receipt object

METHODS	RETURN TYPE
getLink()	Receipt
setLink(Receipt nextLink)	void
getChildren()	int
getAdult()	int



RECEIPTS.JAVA

METHODS	RETURN TYPE	METHODS	RETURN TYPE
isEmpty()	boolean	soldChildren()	int
push (int numChildren, int numAdult)	void	soldAdults()	int
pop()	void		
amountOfReceipts( )	int		
printReceipt()	void		

RECEIPTS.JAVA: dequeue()

```
public Receipts()
{
    // initialize empty Stack
    top = null;
    total = 0;

}
```

RECEIPTS.JAVA: isEmpty()

```
   public boolean isEmpty()
    {
        return top == null;
    }

    // a link-based stack is never
full
    // can always take input
    public boolean isFull()
    {
        return false;
    }
```

RECEIPTS.JAVA: push(child & adult parameters)

```
public void push(int numChildren, int
numAdult)
{
     if(top == null) {
          top = new Receipt(numChildren,
numAdult);
          first = top;
     } else {
          Receipt temp = new
Receipt(numChildren, numAdult);
          temp.setLink(top);
          top = temp;
          }
     total++;
```

RECEIPTS.JAVA: pop()

```
public void pop()
{
   if(top == null) {
       System.out.println("No family
information available.");
    } else {
       Receipt temp = top;
       top = top.getLink();
    }

   total--;
}
```

RECEIPTS.JAVA: amountOfReceipts()

```
public int amountOfReceipts()
 {
      return total;
 }
```

RECEIPTS.JAVA: printReceipt()

```
    public void printReceipt() {
          Receipt temp = top;

          if (temp == null)
               System.out.println("There are no receipts.");

          while (temp != null)
          {
               System.out.println("Children: " + temp.getChildren() + ", Number of
adults: " + temp.getAdult());
               temp = temp.getLink();
          }
     }
```

RECEIPTS.JAVA: soldChildren()

```
public int soldChildren()
  {
        Receipt temp = top;

        if (temp == null)
        {
              return 0;
        }

        int childrenCnt = 0; // test numbers

        while (temp != null)
        {
              childrenCnt += temp.getChildren();
              temp = temp.getLink();
        }
        return childrenCnt;
  }
```

RECEIPTS.JAVA: soldAdults()

```
public int soldAdults()
  {
        Receipt temp = top;

        if (temp == null)
        {
              return 0;
        }

        int adultCnt = 0; // test numbers

        while (temp != null)
        {
              adultCnt += temp.getAdult();
              temp = temp.getLink();
        }
        return adultCnt;
  }
```