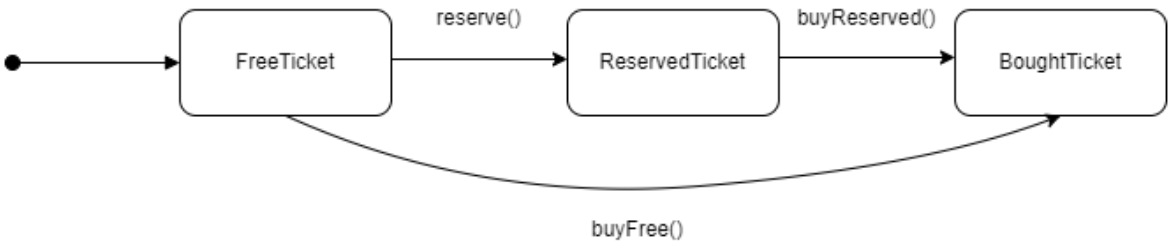
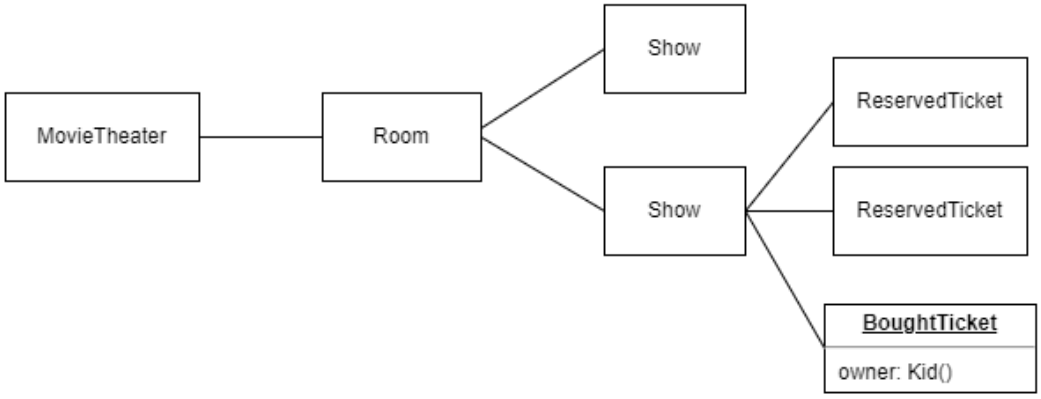
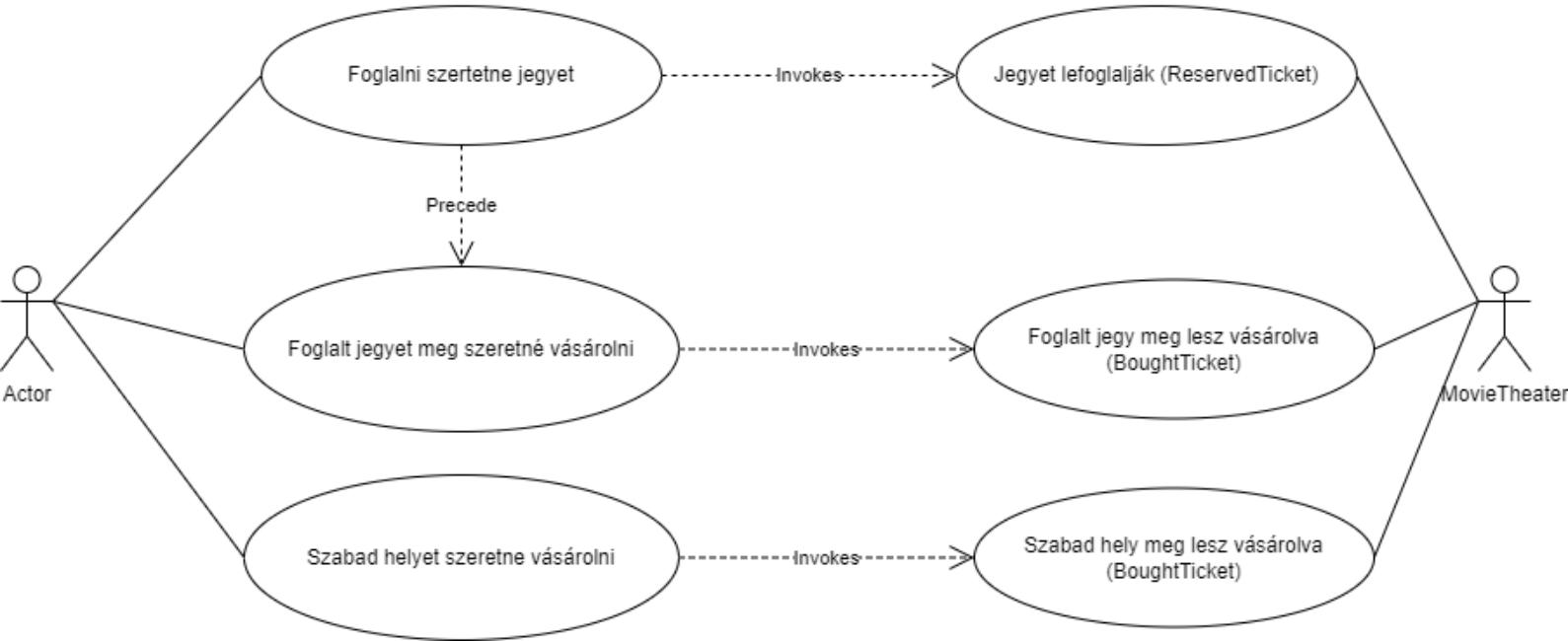
Állapotgép



Objektum diagram



Use-Case Diagram



```
lRoom = false
for each r in rooms:
    if requestedRoom.id == r.id:
        lRoom = true
        break
if lRoom == false:
    throw RoomNotFoundException

lShow = false
requestedShow = null
for each s in requestedRoom.shows:
    if requestedInterval == s.interval:
        requestedShow = s
        lShow = true
        break
if lShow == false:
    throw ShowNotFoundException

lTicket = false
for each t in requestedShow.tickets:
    if t.isFree():
        lTicket = true
        requestedShow.tickets.add(new ReservedTicket(g, t.row, t.column))
        requestedShow.tickets.remove(t)
        break
if lTicket == false:
    throw FullHouseException
```

```
lRoom = false
foundRoom = null
for each r in rooms:
    if requestedRoom.id == r.id:
        foundRoom = r
        lRoom = true
        break
if lRoom == false:
    throw RoomNotFoundException

lShow = false
requestedShow = null
for each s in requestedRoom.shows:
    if requestedInterval == s.interval:
        requestedShow = s
        lShow = true
        break
if lShow == false:
    throw ShowNotFoundException

lTicket = false
for each t in requestedShow.tickets:
    if t.isFree():
        lTicket = true
        requestedShow.tickets.add(new BoughtTicket(g, t.row, t.column,
foundRoom.discountedPrice(g)))
        requestedShow.tickets.remove(t)
        break
if lTicket == false:
    throw FullHouseException
```

```
lRoom = false
foundRoom = null
for each r in rooms:
    if requestedRoom.id == r.id:
        foundRoom = r
        lRoom = true
        break
if lRoom == false:
    throw RoomNotFoundException

lShow = false
requestedShow = null
for each s in requestedRoom.shows:
    if requestedInterval == s.interval:
        requestedShow = s
        lShow = true
        break
if lShow == false:
    throw ShowNotFoundException

lTicket = false
for each t in requestedShow.tickets:
    if t.isReserved() and t.owner == g:
        lTicket = true
        requestedShow.tickets.add(new BoughtTicket(g, t.row, t.column,
foundRoom.discountedPrice(g)))
        requestedShow.tickets.remove(t)
        break
if lTicket == false:
    throw NoReservationException
```

```
moviesNumbers = new List<movieAndViewers>()
movies = new List<string>()

for each r in rooms:
    for each s in r.shows:
        moviesNumbers.add(new movieAndViewers(s.movie,
s.countBoughtTickets()))
        if movies does not contain s.movie:
            movies.add(s.movie)

counter = 0
mostWatchedNumber = 0
mostWatched = ""

for i = 0 to movies.count() - 1:
    for j = 0 to moviesNumbers.count() - 1:
        if moviesNumbers[j].movie == movies[i]:
            counter += moviesNumbers[j].viewers

    if counter > mostWatchedNumber:
        mostWatchedNumber = counter
        mostWatched = movies[i]

    counter = 0

return mostWatched
```

```
counter = 0
for each t in tickets:
    if t.isFree() and !t.isReserved():
        counter += 1
return counter
```

```
for i = 0 to rowNum - 1:
    for j = 0 to colNum - 1:
        tickets.add(new FreeTicket(i, j))
```

```
counter = 0
for each t in tickets:
    if t.isFree():
        counter += 1
return counter
```

```
counter = 0
for each t in tickets:
    if t.isReserved():
        counter += 1
return counter
```

MovieTheater

```
+ reserveTicket(requestedRoom:
Room, requestedInterval: string, g:
Guest)
+ buyFreeTicket(requestedRoom:
Room, requestedInterval: string, g:
Guest)
+ buyReservedTicket(requestedRoom:
Room, requestedInterval: string, g:
Guest)
+ mostWatchedMovie(): string
```

Room

```
# simplePrice: int
# id : int
+ discountedPrice(g : Guest) {virtual}
```

Show

```
- interval : string
- movie : string
+ countFreeTickets(): int
+ countReservedTickets(): int
+ countBoughtTickets(): int
+ fillWithFreeTickets(rowNum: int, colNum: int)
```

Ticket

```
# row: int
# column: int
+ isFree(): Bool {virtual}
+ isReserved(): Bool
{virtual}
```

Guest

```
# name : string
+ Discount(k: Medium) {virtual}
+ Discount(k: Large) {virtual}
+ Discount(k: VIP) {virtual}
+ reserve(requestedRoom: Room,
requestedInterval: string)
+ buyFree(requestedRoom: Room,
requestedInterval: string)
+ buyReserved(requestedRoom:
Room, requestedInterval: string)
```

BoughtTicket

```
- owner: Guest
- price: double
+ isFree(): Bool {override}
+ isReserved(): Bool {override}
```

ReservedTicket

```
- owner: Guest
+ isFree(): Bool {override}
+ isReserved(): Bool
{override}
```

FreeTicket

```
+ isFree(): Bool {override}
+ isReserved(): Bool
{override}
```

Medium

```
+ discountedPrice(g :
Guest) {override}
```

Large

```
+ discountedPrice(g :
Guest) {override}
```

VIP

```
+ discountedPrice(g :
Guest) {override}
```

Kid

```
+ Discount(k: Medium)
{override}
+ Discount(k: Large)
{override}
+ Discount(k: VIP)
{override}
```

Student

```
+ Discount(k: Medium)
{override}
+ Discount(k: Large)
{override}
+ Discount(k: VIP)
{override}
```

Adult

```
+ Discount(k: Medium)
{override}
+ Discount(k: Large)
{override}
+ Discount(k: VIP)
{override}
```

Old

```
+ Discount(k: Medium)
{override}
+ Discount(k: Large)
{override}
+ Discount(k: VIP)
{override}
```

Frequent

```
+ Discount(k: Medium)
{override}
+ Discount(k: Large)
{override}
+ Discount(k: VIP)
{override}
```

```
theater.reserveTicket(requestedRoom, requestedInterval, this)
```

```
theater.buyFreeTicket(requestedRoom, requestedInterval, this)
```

```
theater.buyReservedTicket(requestedRoom, requestedInterval,
this)
```

```
return (simplePrice * (100 - g.Discount(this) / 100))
```

```
return (simplePrice * (100 - g.Discount(this) / 100))
```

```
return (simplePrice * (100 - g.Discount(this) / 100))
```

```
return 40
return 40
return 0
```

```
return 30
return 20
return 0
```

```
return 10
return 0
return 0
```

```
return 30
return 20
return 0
```

```
return 30
return 0
```

```
return true
return false
```

```
return false
return true
```

```
return false
return false
```

0...* rooms

* shows

* tickets

1 theater

0...1 owner