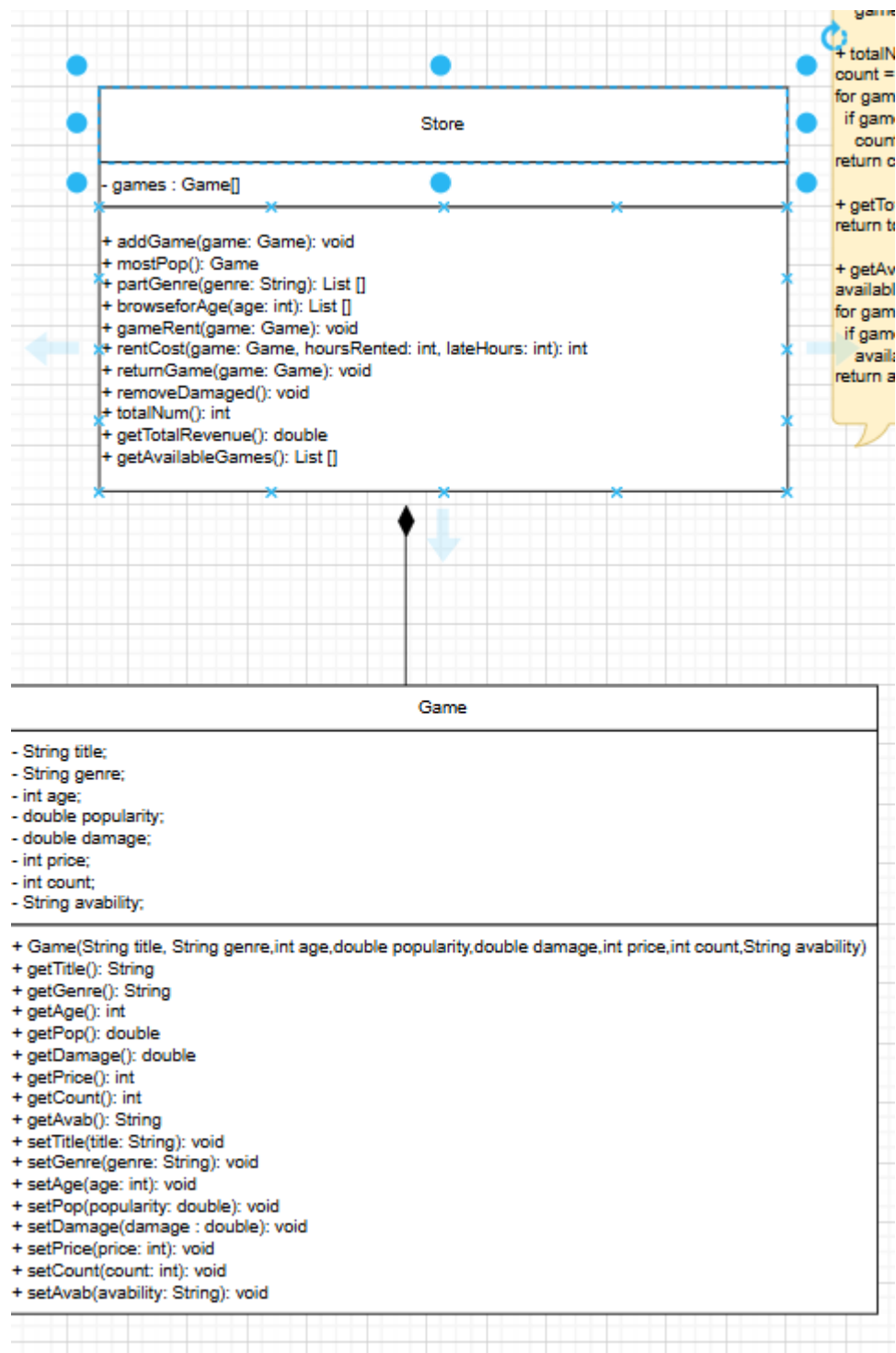


6 Video Game Rental Store A video game rental store manages a collection of video games, each with distinct attributes that determine its availability and demand. Every game has a title, a genre (such as action, puzzle, or simulation), an age rating that indicates suitability for players, a popularity score based on previous rentals, a damaged status indicating whether the game is still functional, a rental price per hour, and a total rent count that tracks how many times the game has been rented. Each game also has an availability status showing whether it is currently rented or in stock. Customers visiting the store may have different preferences when selecting a game. Some may look for the most popular game based on past rentals, while others may search for a game within a particular genre or browse games suitable for their age group. The store must efficiently manage game availability to ensure that customers can find the games they want. Games become damaged. Every time a game is rented, its total rent count increases, and with repeated use, it may sustain wear and tear. A game is considered damaged and permanently removed from circulation if it has been rented more than 20 times, marking it as too worn out for further use. The store also enforces a rental pricing system. Each game has a fixed price per hour, and rental costs are calculated based on the total hours rented. Late returns result in an additional fine, which is calculated as twice the hourly rental rate for each overdue hour. To maintain an organized rental system, the store tracks the number of games rented, their due dates, and applies late fees if they are returned past the deadline. Additionally, the system keeps records of the total revenue from rentals, ensuring that the store can analyze which genres and titles are most profitable. Over time, games that are damaged are removed from circulation. By keeping detailed records of rental statistics, customer preferences, and game performance, the store ensures that only high-demand and properly functioning games remain available for rent, creating a better player experience.



```

+ addGame(game: Game) : void
games.add(game)

+ mostPop() : Game
max = 0
g = null
for game in games:
    if game.getPop() > max:
        max = game.getPop()
        g = game
return g

+ partGenre(genre: String) : Game[]
gamesofGenres = []
for game in games:
    if game.getGenre() == genre:
        gamesofGenres.add(game)
return gamesofGenres

+ browseforAge(age: int) : Game[]
gamesforAge = []
for game in games:
    if game.getAge() == age:
        gamesforAge.add(game)
return gamesforAge

+ gameRent(game: Game) : void
if game.getAvab().equals("in stock"):
    game.setCount(game.getCount() + 1)
    game.setDamage(game.getDamage() + 1)
    game.setAvab("rented")

+ rentCost(game: Game, hoursRented: int, lateHours: int) : int
rentalCost = game.getPrice() * hoursRented
lateFee = 2 * game.getPrice() * lateHours
totalCost = rentalCost + lateFee
totalRevenue = totalRevenue + totalCost
return totalCost

+ returnGame(game: Game) : void
if game.getAvab().equals("rented"):
    game.setAvab("in stock")

+ removeDamaged() : void
for game in games:
    if game.getDamage() > 20:
        games.remove(game)

+ totalNum() : int
count = 0
for game in games:
    if game.getAvab().equals("rented"):
        count = count + 1
return count

+ getTotalRevenue() : double
return totalRevenue

+ getAvailableGames() : Game[]
availableGames = []
for game in games:
    if game.getAvab().equals("in stock"):
        availableGames.add(game)
return availableGames

```

```
+ Game(String title, String genre, int age, double popularity, double damage, int price, int count, String
availability): void
this.title = title
this.genre = genre
this.age = age
this.popularity = popularity
this.damage = damage
this.price = price
this.count = count
this.avability = availability

+ getTitle(): String
return this.title

+ getGenre(): String
return this.genre

+ getAge(): int
return this.age

+ getPop(): double
return this.popularity

+ getDamage(): double
return this.damage

+ getPrice(): int
return this.price

+ getCount(): int
return this.count

+ getAvab(): String
return this.avability

+ setTitle(String title): void
this.title = title

+ setGenre(String genre): void
this.genre = genre

+ setAge(int age): void
this.age = age

+ setPop(double popularity): void
this.popularity = popularity

+ setDamage(double damage): void
this.damage = damage

+ setPrice(int price): void
this.price = price

+ setCount(int count): void
this.count = count

+ setAvab(String availability): void
this.avability = availability
```