Bus Seats

**Problem**

A bus of our fleet has a capacity of **N** seats, **M** of these are booked and **P** are still available. **R** users are currently considering booking a seat online. How can we avoid to overbook our bus? Explain your reasoning and solutions, step by step. Try to be as clear and concise as possible. Pseudo code in the answer is considered a plus.

**Risoluzione**

Creiamo una classe Bus con i seguenti attributi:

* **seats**: rappresenta la capacità degli N posti disponibili sul nostro Bus
* **seats\_booked**: M posti già prenotati
* **seats\_available**: posti ancora disponibili nel Bus, tale variabile sarà settata come differenza tra **seats** e **seats\_booked**.

L’implementazione della classe col relativo costruttore è la seguente:

1. **class** Bus
2. {
3. **private** $seats;
4. **private** $seats\_booked;
5. **private** $seats\_available;

8. **public** **function** \_\_construct($seats = 100,$seats\_booked = 0)
9. {
10. $this->seats = $seats;
11. $this->seats\_booked = $seats\_booked;
12. $this ->updateAvailable();
14. }

Possiamo notare la presenza nel costruttore del metodo **updateAvailable()** che ai occuperà dell’aggiornamento dei posti ancora disponibili dopo una prenotazione.

1. **private** **function**  updateAvailable()
2. {
3. $this->seats\_available = $this->seats - $this->seats\_booked;
4. }

Adesso andiamo a definire la funzione **checkAvailable** che ci permette di controllare la disponibiltà di prenotazione.

1. **public** **function** checkAvailable($book = 0)
2. {
4. **switch** ($book)
5. {
6. **case** 0:
7. **if**($this->seats\_available> 0)
8. {
9. **return** true;
10. }
11. **break**;
12. **default**:
13. **if**($this->seats\_available < $book)
14. {
15. **return** false;
16. }
17. **else**
18. {
19. **return** true;
20. }
21. **break**;
23. }
24. }

Abbiamo un parametro di ingresso “book” che rappresenta il numero di posti che si stanno cercando di prenotare, il suo valore di default è 0.

Controlliamo il valore di **seats\_available** confrontandolo con **book**, se quest’ultimo risulta essere inferiore potremo procedere con la prenotazione.

La funzione principale che gestisce la prenotazione dei posti è **addBooking**.

1. **public** **function**  addBooking($num)
2. {
3. $checkAvailable=$this->checkAvailable($num);
4. **if**($checkAvailable) {
5. $this->seats\_booked += $num;
6. $this->updateAvailable();
7. }
8. }

Troviamo il parametro **num** che rappresenta il numero di posti che vorremmo prenotare.

La prima operazione che verrà eseguita è il controllo della disponibilità di posti liberi. Se la funzione **checkAvailable** avrà un valore di ritorno positivo, sarà possibile proseguire con la prenotazione aggiornando sia i posti ancora disponibili che quelli occupati.

**Richiamare la funzione**

E’ possibile testare tutto il procedimento tramite url:

[**http://localhost/api/booking.php?booking=56**](http://localhost/api/booking.php?booking=56)

Possiamo notare la presenza del parametro GET “booking”, il cui valore rappresenta il numero di posti che vorremmo andare a prenotare.

La chiamata eseguirà le seguenti istruzioni codice:

1. $booking = intval($\_GET['booking']);
2. $bus = **new** Bus(100,32);
3. $bus->addBooking($booking );

Creiamo un oggetto di tipo Bus che avrà capacità N = 100 e posti già prenotati 32.

Andremo poi ad usare la funzione **addBooking** pasando come paramtro quello recuperato dall’url in GET.

**Gestione richieste Multiple**

In un normale sistema di prenotazione, accade di dover gestire N richieste di prenotazione provenienti da dispositivi differenti come PC o smartphones.

Cosa fare nel caso in cui arrivino più richieste di prenotazione contemporaneamente?

Possiamo ipotizzare la presenza di un componente che crei una “coda” di richieste che verranno eseguite in ordine di arrivo (FIFO).

Supponiamo di avere N richieste di prenotazione, è impensabile permettere a tutte contemporaneamente di prenotare in quanto il sistema potrebbe sicuramente generare degli errori di diverso tipo.

Andremo a generare una coda di prenotazione, in cui ogni richiesta avrò il suo limite di tempo per essere eseguita, bloccando le risorse del sistema ed evitando che altre richieste possano interferire con l’operazione.

Si potrebbe ipotizzare l’utilizzo di un “Timer” impercettibile all’utente, che renda per un lasso di tempo il servizio di prenotazione bloccato per la richiesta che ha priorità nella coda generata in precedenza.