Создать набор классов (**Reservator и ReservatorApiPartner**), который позволит зарезервировать билет (сгенерировать случайный ticketId внутри класса) для разных типов событий и получить номер заказа (сгенерировать случайный orderId внутри класса). Соответствующий объект создается отдельно для каждого события (через фабрику, либо статичный фабричный метод основываясь на типе события). Всю логику специфичную для реализации партнерского события вынести в отдельный класс избегая дублирования кода.

Все реальные действия по работе с базой, отсылкой почты и тд заменить просто выводом сообщений типа ***'reserving ticket #64244'***.

Есть **два типа** событий, одни **локальные**, вторые – берутся **по апи** и для резервации билета на них необходимо вызвать команду стороннего партнерского апи и получить результат (для упрощения достаточно заменить это все на вывод сообщения ***'reserving ticket #34580 VIA PARTNER API CALL' –*** *реализовать как метод в классе ReservatorApiPartner*)

Для простоты, все **четные** ID событий пускай будут **локальными**, **нечетные – партнеские.**

То есть для набора событий $eventIds = [14, 21, 587, 82]**; вывод должен быть соответствующим тому, который показан на скриншоте** на последней странице документа (за исключением случайных чисел).

Вызывающий код приблизительно такой:

```
$eventIds = [14, 21, 587, 82];
foreach ($eventIds as $eventId) {
    // creating reservator object $o depending on current event id
    // your code for creating appropriate object here
    $o->reserveRandomTicket();
    echo "<h5>--------------------------------------------------</h5>\n";
}
```

В зависмости от типа текущего обрабатываемого события все сообщения должны иметь префикс [local] | [partnerApi] (как на скриншоте).

Для локальных событий все случайные числа генерировать в диапазоне 99 – 3000 (создать отдельный метод),

для партнерских 80000 – 90000 (переопределить этот метод в *ReservatorApiPartner*)

Алгоритм приблизительно следующий:

- По id события определяем его тип
- Генерируем случайный ticketId
- Резервируем сгенереный ticketId (заглушка – вывод сообщения)
- Создаем заказ orderId для билета ticketId (генерим случайное число и заглушка – вывод сообщения)
  * если событие локальное, то просто выводим **'orderId #xxx created'**
  ***** если партнерское – симулируем вызов по апи, выводя сообщение '**reserving ticket #xxx VIA PARTNER API CALL'**
- Эмулирем отсылку сообщения админам о создании заказа такого вида
  'Sending admin notification: Order #yyy created for event #zzz'

В результате обработки указанных событий ожидаем вывод соответствующий тому, что показан на скрине ниже (за исключением случайных чисел).

Цель задания – проверить навыки применения наследования и полиморфизма на практике, умение МАКСИМАЛЬНО ИЗБЕГАТЬ ДУБЛИРОВАНИЯ КОДА. По задумке класс ReserveApiPartner наследует Reserve и переопределяет в нем нужную функциональность, если у вас есть свой подход – посмотрим на него.

[local] | creating object for event #14

[local] | reserving ticket #308

[local] | orderId #365 created

[local] | Sending admin notification: Order #365 created for event #14

[partnerApi] | creating object for event #21

[partnerApi] | reserving ticket #83826

[partnerApi] | reserving ticket #83826 VIA PARTNER API CALL

[partnerApi] | orderId #88144 created

[partnerApi] | Sending admin notification: Order #88144 created for event #21

партнерские события

[partnerApi] | creating object for event #587

[partnerApi] | reserving ticket #89268

[partnerApi] | reserving ticket #89268 VIA PARTNER API CALL

[partnerApi] | orderId #82806 created

[partnerApi] | Sending admin notification: Order #82806 created for event #587

[local] | creating object for event #82

[local] | reserving ticket #2359

[local] | orderId #1783 created

[local] | Sending admin notification: Order #1783 created for event #82
