Design a platform for buying tickets of local events

User can

1. Login App(platform)
2. Search ticket for the local event
3. Choose date and price for the local event
4. Confirm then place the order
5. Get ticket(E-ticket/Printed-ticket)
6. Cancel order

Class User

Information:

```
String name[ ];
String UserId;
String Password;
int TicketNumber;
String Email;
String ShippingAddress;
```

Behavior:

```
LoginApp(UserId, Password ){
}
SearchTicketForLocalEvent( ){
}
ChooseDateAndPriceForLocalEvent(TicketNumber ){
    If(App.CheckStock(Date,Price)<TicketNumber)
        System.out.println("The chosen ticket is unavailable!");
    Else
        App.EnterTicketInfo( );
}
ConfirmAndPlaceOrder( ){
    Payment.ChooseCard( );
    App.DeductStock( );
}
GetTicket( )
    App.SendTicket( );
}
```

```
    CancelOrder( ){
        App.IncreaseStock( );
        Payment.Refund( );
    }
```

```
Class App
Information:
    int Stock;
    LocalDate Date;
    int Price;
    String TicketInfo[];
Behavior:
    CheckStock(Date,Price){
    }
    DeductStock(TicketNumber){
        Stock-=TikcetNumber;
    }
    IncreaseStock (TicketNumber){
        Stock+=TikcetNumber;
    }
    EnterTicketInfo ( ){
        For(int i=0,i<TikcetNumber,i++)
            TicketInfo[i]=Name[i];
    }
    SendTicket(Email, ShippingAddress){
        If(Email||ShippingAddress!=null)
            Continue;
        Else
            System.out.println(“Please choose one shipping method.”);
    }
```

```
Class Payment
Information:
    String Type;
    int CardNumber;
    String NameOnCard;
```

```
        int SecurityCode;
        Float ExpireDate;
Behavior:
        Date CurrentDate=this.date;
        ChooseCard (Type, CardNumber, NameOnCard, SecurityCode, ExpireDate){
                If(ExpireDate>=CurrentDate)
                        Continue;
                Else
                        End;
        }
        Refund ( ){
        }
```