

Thiết kế xây dựng phần mềm



Project: Ecopark

Nhóm 16 thực hiện

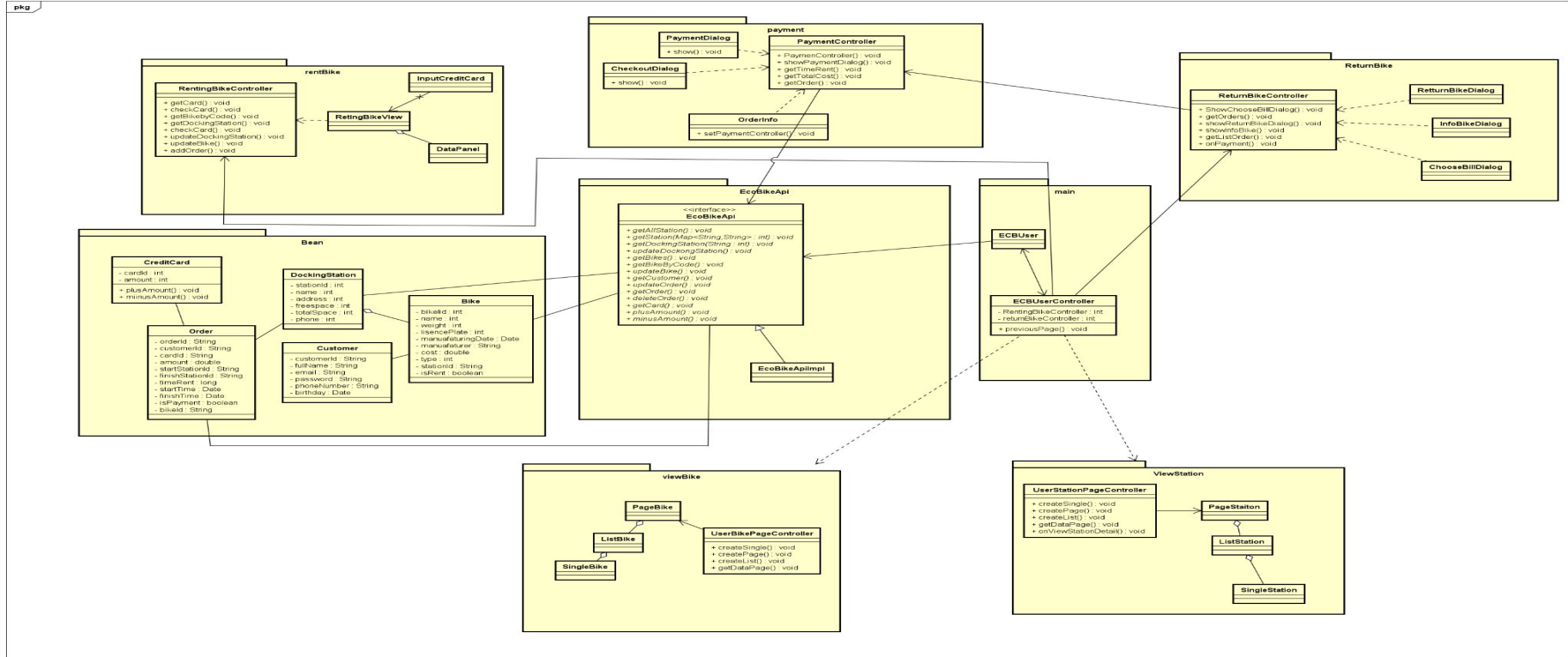
GV hướng dẫn: TS.Nguyễn Thị Thu Trang

Phân công công việc

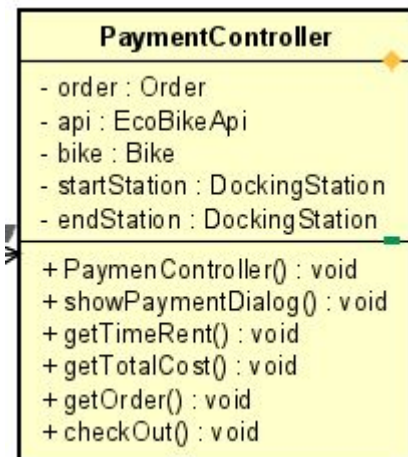
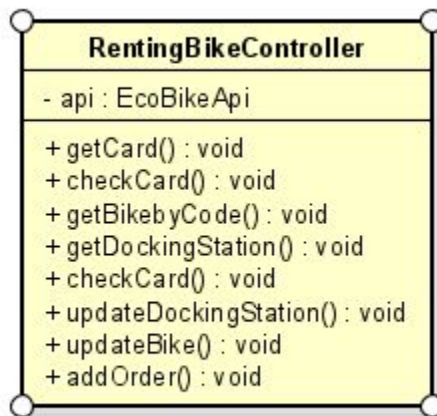
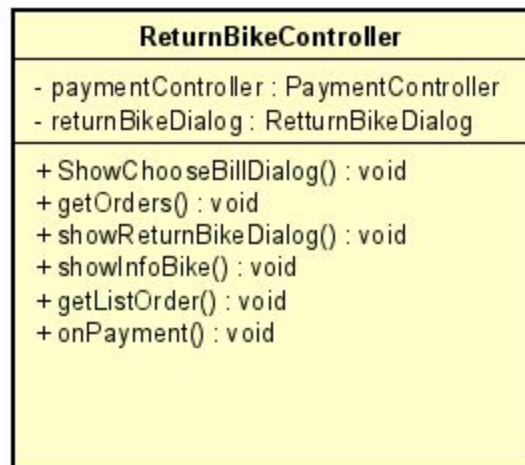


| Họ và tên | MSSV | Công việc hoàn thành |
|------------------|----------|----------------------|
| Lê Đức Hải | 20173094 | Module Thanh toán |
| Lê Thị Mai Hương | 20173164 | Module Trả xe |
| Dương Hồng Tuấn | 20173439 | Module Thuê xe |
| Lý Trung Kiên | 20173207 | Module Tìm kiếm |

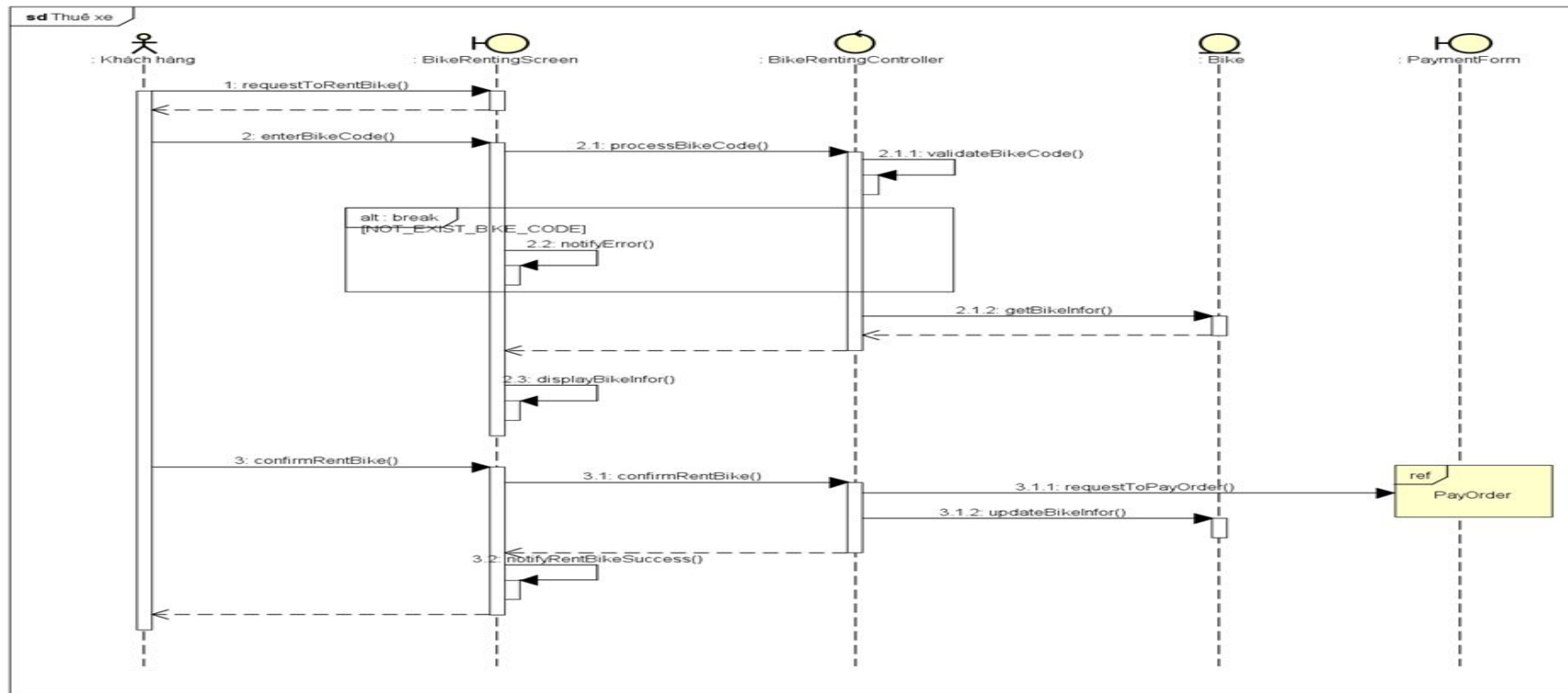
Biểu đồ lớp (General)



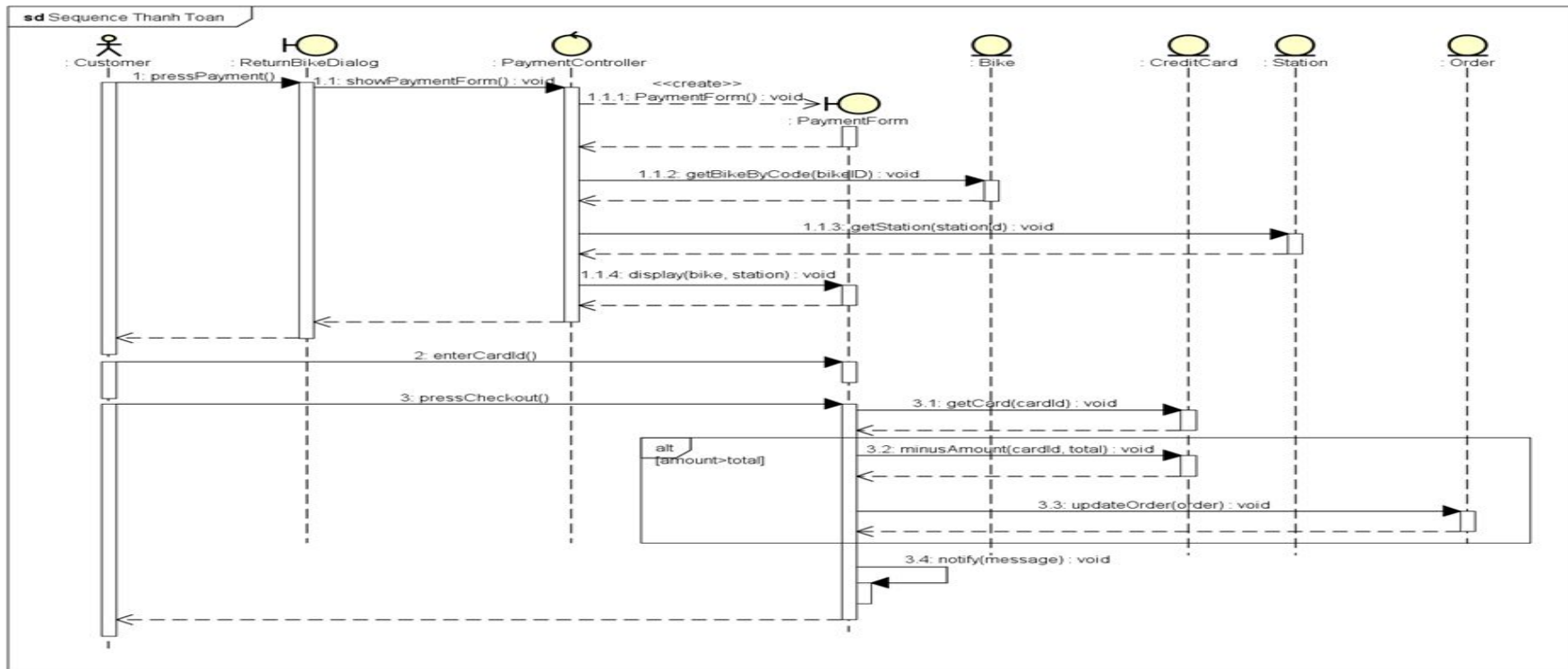
Biểu đồ lớp (Detailed)



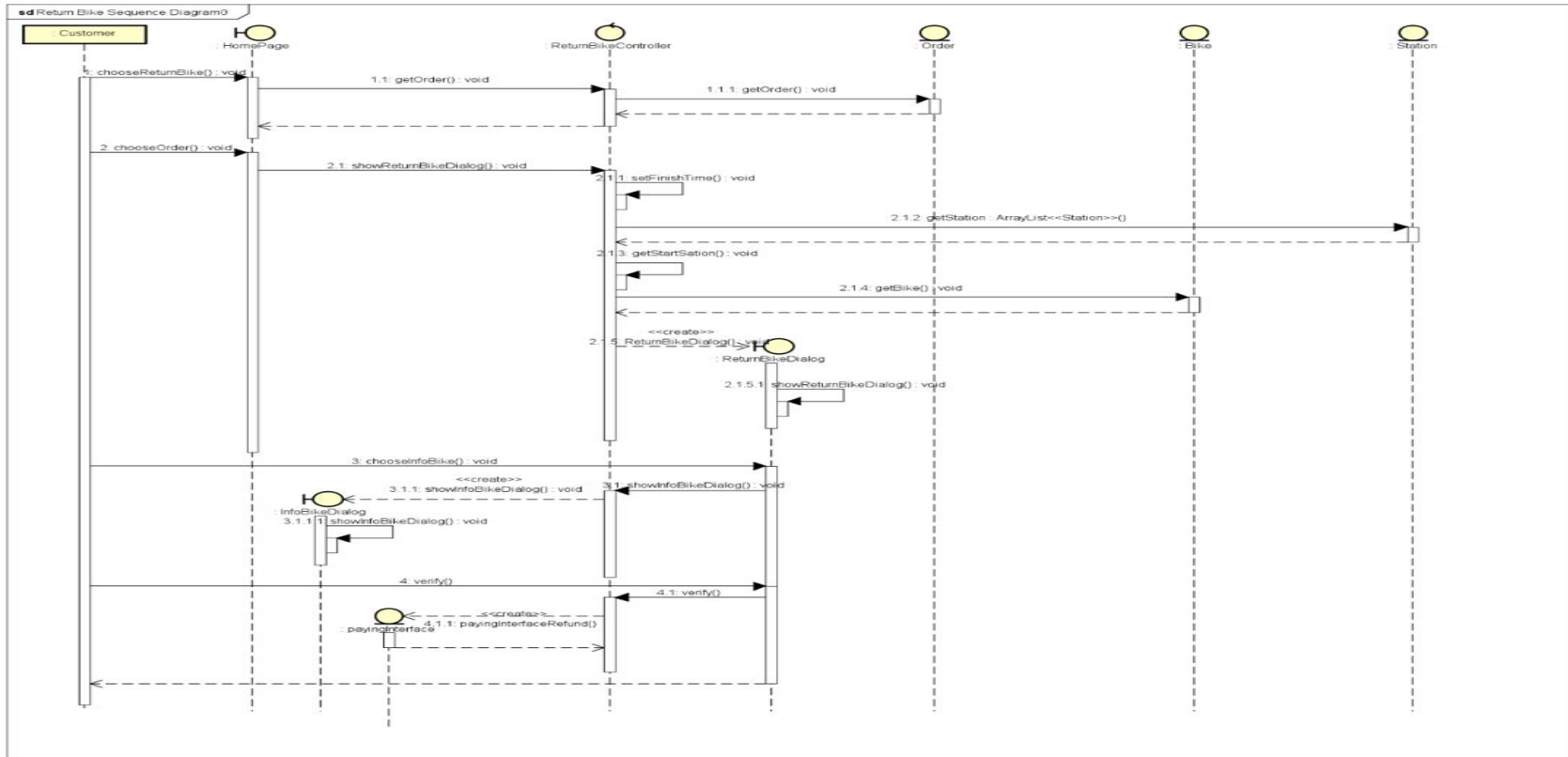
Biểu đồ trình tự: Thuê xe



Biểu đồ trình tự: Thanh toán



Biểu đồ trình tự: Trả xe



Design Considerations

1. Coupling

- Stamp coupling : Dữ liệu truyền vào phương thức không ở dạng data ví dụ như object Order, Card

```
public HashMap<String, Order> getOrdersNotPayment(String customerId, ArrayList<Order> orders) {  
    if (orders == null) {  
        paymentController.openCheckoutDialog("Hiện không có hóa đơn nào!");  
    } else {  
        Iterator<Order> iter = orders.iterator();  
        while (iter.hasNext()) {  
            Order order = iter.next();  
            if (order.getCustomerId().equals(customerId)) {  
                if (order.isPayment() == false) {  
                    listOrder.put(order.getBikeId(), order);  
                }  
            }  
        }  
    }  
}
```


Design Considerations

1. Coupling

- Stamp coupling : Dữ liệu truyền vào phương thức không ở dạng data ví dụ như object Order, Card

```
public HashMap<String, Order> getOrdersNotPayment(String customerId, ArrayList<Order> orders) {  
    if (orders == null) {  
        paymentController.openCheckoutDialog("Hiện không có hóa đơn nào!");  
    } else {  
        Iterator<Order> iter = orders.iterator();  
        while (iter.hasNext()) {  
            Order order = iter.next();  
            if (order.getCustomerId().equals(customerId)) {  
                if (order.isPayment() == false) {  
                    listOrder.put(order.getBikeId(), order);  
                }  
            }  
        }  
    }  
}
```

Design Considerations



1. Coupling

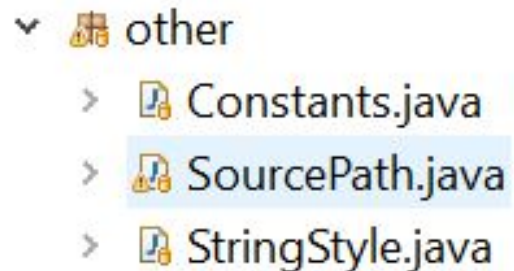
- Control coupling (VD: Hình dưới, phương thức truyền vào obj Station, mà obj này liên quan đến việc đề điều khiển tính toán các loại xe có trong bãi)

```
public BikePageHeader(DockingStation station) {  
    // TODO Auto-generated constructor stub  
    labelStation = new JLabel();  
    labelStation.setFont(StringStyle.BIG_FONT);  
    labelAddress = new JLabel();  
  
    int bike=0, ebike=0, twinbike=0, edbike=0;  
  
    for (com.ecb.bean.Bike _bike: station.getListBike()) {  
        if(_bike.getType() == Constants.BIKE.bike)  
            bike ++;  
        if(_bike.getType() == Constants.BIKE.ebike)  
            ebike ++;  
        if(_bike.getType() == Constants.BIKE.twinBike)  
            twinbike ++;  
        if(_bike.getType() == Constants.BIKE.edBike)  
            edbike ++;  
    }  
}
```

Design Considerations

1. Cohesion








- **Coincidental cohesion:** Trong package **other**, các sub component Constants.java (chứa các hằng số), SourcePath.java (chứa đường dẫn) và StringStyle.java (chứa định dạng xâu) được đặt trong cùng 1 package vì tính ngẫu nhiên



Design Considerations

1. Cohesion

- **Logical cohesion:** Các thành phần trong package **api** về mặt logic đều dùng để gọi API
 - + Chia thành 2 package: **api** (gọi api dữ liệu) và **api.bank** (gọi api liên quan đến ngân hàng)

- ▼  **api**
 - >  EcoBikeApi.java
 - >  EcoBikeApiFactory.java
 - >  EcobikeApimpl.java
- ▼  **api.bank**
 - >  BankApimpl.java
 - >  IBankApi.java

Design Considerations

1. Cohesion

- **Procedural cohesion:** Trong class `PaymentController` của package `payment`, các phương thức liên quan đến dialog được gọi theo thứ tự từng bước một (show, open, close)

```
PaymentController.java x
43
44 public void showPaymentDialog() {
45     if (order.isPayment()) {
46         openCheckoutDialog("Đơn đã được thanh toán!");
47         return;
48     }
49     startStation = api.getDockingStation(order.getStartStationId());
50     endStation = api.getDockingStation(order.getFinishStationId());
51     bike = api.getBikeByCode(order.getBikeId());
52
53     paymentDialog = new PaymentDialog(orderInfoPanel, paymentInfoPanel);
54     paymentDialog.setLocationRelativeTo(SwingUtilities.getWindowForComponent(paymentDialog));
55     paymentDialog.update(bike, startStation, endStation);
56     paymentDialog.setVisible(true);
57
58 }
59
60 public void openCheckoutDialog(String message) {
61     AlertCheckoutDialog alertCheckoutDialog = new AlertCheckoutDialog(message);
62     alertCheckoutDialog.setLocationRelativeTo(SwingUtilities.getWindowForComponent(alertCheckoutDialog));
63     alertCheckoutDialog.setVisible(true);
64 }
65
66 public void closePaymentDialog() {
67     paymentDialog.setVisible(false);
68     paymentDialog.dispose();
69 }
70
```

Design Considerations

1. Cohesion

- Communication cohesion:

output của getTimeRent() là input của getTotalCost()

```
70
71Ⓢ public long getTimeRent() {
72     long millis = Math.abs(order.getFinishTime().getTime()
73         - order.getStartTime().getTime());
74     return millis;
75 }
76
77Ⓢ public double getTotalCost(int bikeType, long minutes) {
78     double total = 0;
79
80     // count total price
81     if (minutes <= 10) {
82         total = 0;
83     } else if (minutes <= 30) {
84         total = 10000;
85     } else {
86         if (bikeType != 0) {
87             total = 10000 + (Math.ceil((minutes - 30) * 1.0 / 15) * 3000) * 1.5;
88         }
89         else {
90             total = 10000 + Math.ceil((minutes - 30) * 1.0 / 15) * 3000;
91         }
92     }
93
94     return total;
95 }
96
```

Design Principles



1. Single Responsibility Principle (SRP)

Hầu hết các lớp sẽ chỉ thực hiện một chức năng duy nhất

2. Open Closed Principle (OCP)

Sử dụng các lớp cha để tổng quát hóa, ví dụ như các lớp trong `abstractdata` package để cho phần `viewBike`, `viewStation` kế thừa

Design pattern



- Sử dụng Singleton:

Lớp seed để đọc dữ liệu từ file json mà hệ thống mong muốn chỉ có duy nhất 1 lớp để thực hiện việc này -> giải pháp sử dụng singleton

```
public class Seed {  
  
    private ArrayList<Order> orders;  
    private ArrayList<Bike> bikes;  
    private ArrayList<DockingStation> stations;  
    private ArrayList<CreditCard> cards;  
    private ArrayList<Customer> customers;  
  
    private static Seed singleton = new Seed();  
  
    private Seed() {  
        start();  
    }  
}
```