Java

```java
class RideSharingSystem {

  public RideSharingSystem() {

  }

  public void addRider(int riderId) {

  }

  public void addDriver(int driverId) {

  }

  public int[] matchDriverWithRider() {

  }

  public void cancelRider(int riderId) {

  }
}

/**
 * Your RideSharingSystem object will be instantiated and called as such:
 * RideSharingSystem obj = new RideSharingSystem();
 * obj.addRider(riderId);
 * obj.addDriver(driverId);
 * int[] param_3 = obj.matchDriverWithRider();
 * obj.cancelRider(riderId);
 */
```

JavaScript

```javascript
var RideSharingSystem = function() {

};

/**
 * @param {number} riderId
 * @return {void}
 */
RideSharingSystem.prototype.addRider = function(riderId) {

};

/**
 * @param {number} driverId
 * @return {void}
 */
```

```javascript
RideSharingSystem.prototype.addDriver = function(driverId) {

};

/**
 * @return {number[]}
 */
RideSharingSystem.prototype.matchDriverWithRider = function() {

};

/**
 * @param {number} riderId
 * @return {void}
 */
RideSharingSystem.prototype.cancelRider = function(riderId) {

};

/**
 * Your RideSharingSystem object will be instantiated and called as such:
 * var obj = new RideSharingSystem()
 * obj.addRider(riderId)
 * obj.addDriver(driverId)
 * var param_3 = obj.matchDriverWithRider()
 * obj.cancelRider(riderId)
 */
```

TypeScript

```typescript
class RideSharingSystem {
  constructor() {

  }

  addRider(riderId: number): void {

  }

  addDriver(driverId: number): void {

  }

  matchDriverWithRider(): number[] {

  }

  cancelRider(riderId: number): void {

  }
}
```

```
/**
 * Your RideSharingSystem object will be instantiated and called as such:
 * var obj = new RideSharingSystem()
 * obj.addRider(riderId)
 * obj.addDriver(driverId)
 * var param_3 = obj.matchDriverWithRider()
 * obj.cancelRider(riderId)
 */
```

C++

```
class RideSharingSystem {
public:
  RideSharingSystem() {

  }

  void addRider(int riderId) {

  }

  void addDriver(int driverId) {

  }

  vector<int> matchDriverWithRider() {

  }

  void cancelRider(int riderId) {

  }
};

/**
 * Your RideSharingSystem object will be instantiated and called as such:
 * RideSharingSystem* obj = new RideSharingSystem();
 * obj->addRider(riderId);
 * obj->addDriver(driverId);
 * vector<int> param_3 = obj->matchDriverWithRider();
 * obj->cancelRider(riderId);
 */
```

C#

```
public class RideSharingSystem {

  public RideSharingSystem() {

  }
```

```
  public void AddRider(int riderId) {

  }

  public void AddDriver(int driverId) {

  }

  public int[] MatchDriverWithRider() {

  }

  public void CancelRider(int riderId) {

  }
}

/**
 * Your RideSharingSystem object will be instantiated and called as such:
 * RideSharingSystem obj = new RideSharingSystem();
 * obj.AddRider(riderId);
 * obj.AddDriver(driverId);
 * int[] param_3 = obj.MatchDriverWithRider();
 * obj.CancelRider(riderId);
 */
```

Kotlin

```
class RideSharingSystem() {

  fun addRider(riderId: Int) {

  }

  fun addDriver(driverId: Int) {

  }

  fun matchDriverWithRider(): IntArray {

  }

  fun cancelRider(riderId: Int) {

  }

}

/**
 * Your RideSharingSystem object will be instantiated and called as such:
 * var obj = RideSharingSystem()
 * obj.addRider(riderId)
```

```
 * obj.addDriver(driverId)
 * var param_3 = obj.matchDriverWithRider()
 * obj.cancelRider(riderId)
 */
```

Go

```
type RideSharingSystem struct {

}


func Constructor() RideSharingSystem {

}


func (this *RideSharingSystem) AddRider(riderId int)  {

}


func (this *RideSharingSystem) AddDriver(driverId int)  {

}


func (this *RideSharingSystem) MatchDriverWithRider() []int {

}


func (this *RideSharingSystem) CancelRider(riderId int)  {

}


/**
 * Your RideSharingSystem object will be instantiated and called as such:
 * obj := Constructor();
 * obj.AddRider(riderId);
 * obj.AddDriver(driverId);
 * param_3 := obj.MatchDriverWithRider();
 * obj.CancelRider(riderId);
 */
```
