1. **Coding Standards**

1. java

1. class, interface, enums

- pascal case
    - eg - GuestDao, HostDto

2. Enums

- should be appended with Enum.
- should be decalred as seprate files

3. method,interface,fields

- camel case
    - eg - guestDao, hostDto

4. final fields , final static fields, enum constants

- SCREAMING_SNAKE_CASE
    - eg- public enum RoleEnum{HOST, GUEST, ADMIN}

5. packages

- package name - snakecase and all lower and singular
    - base_package - com.rentmycar
    - **all packages**

    ```
    com.rentmycar.exception_handler
    com.rentmycar.security
    com.rentmycar.custom_exception
    com.rentmycar.dao
    com.rentmycar.dto
    com.rentmycar.entity
    com.rentmycar.aspect
    com.rentmycar.service
    ```

6. Exceptions and status codes

- **if already registerd** -
    - SC 409 , conflict
    - **exception** -
        - ConflictException("user with given email and mobile already registered");
        - ConflictException("user with given email already registered");

        - ConflictException("user with given mobile already registered");
- **if password and confirm password doesn't match** -
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("password mismatch", null);
- **if expiry date is before creation date**
    - SC 400 , bad request
    - **exception** - ConstraintViolationException("expiry should come after issue date of dl ", null);

2. mysql

- all table name should be snake case and lower and singular
- all column names must be snake case and lower
    - eg- user_id
- **foreign key must be 'table_name' + '_' + 'id'**
- 'user' table -> id (from mapped super class)
- 'adress' table -> user_id (foreign key to user(id))
    - user 1 <------ * address

3. git

1. to careate a branch and switch to that branch

```
        git switch -C <feature-name>
        git switch -C feature/create-guest-entity
```

branch creation coding standard

- Feature Branches (feature/):
    - Purpose: For developing new features or enhancements.
        - Naming Example: feature/add-user-profile, feature/implement-searchfunctionality
- Bugfix Branches (bugfix/ or fix/):
    - Purpose: For fixing bugs or issues in the codebase.
        - Naming Example: bugfix/fix-login-error, fix/missing-footer-on-homepage
- Documentation Branches (docs/):
    - Purpose: For updates or changes to documentation.
        - Naming Example: docs/update-readme, docs/improve-api-docs
- Chore Branches (chore/):
    - Purpose: For maintenance tasks that don’t directly affect functionality, such as refactoring or updating dependencies.
        - Naming Example: chore/update-dependencies, chore/refactor-codebase

2. to check the all the branches and current branch

```
    git branch
```

3. to add the changes to local repo

```
    git add .
```

4. to commit the changes to local repo

```
        git commit -m "feat: add guest entity"
```

1. commit creation coding standard

- **feat** : New features or enhancements
  - **Example** - feat: add user profile page
- **fix** : Bug fixes
  - **Example** - fix: correct header image rendering
- **docs**: Documentation changes
  - **Example** - docs: update README with new setup instructions
- **style**: Formatting, missing semi-colons, etc. (no code change)
  - **Example** - style: format code according to eslint
- **refactor**: Code changes that neither fix bugs nor add features
  - **Example** - refactor: improve variable naming in utils.js
- **test**: Adding or updating tests
  - **Example** - test: add unit tests for user login functionality
- **chore**: Changes to the build process or auxiliary tools/libraries
  - **Example** - chore: update dependencies

5. to push the changes to remote branch

```
     git push -u origin feature/create-guest-entity
```

6. to pull the changes to main branch

```
git switch main
git fetch
git pull origin main
```

7. what if i have edited files in main branch

```
## This will revert all changes in the working directory to the last committed
state.
git restore .
```

8. what if i have staged files in main branch

```
#1. Unstage the Changes
## To unstage all changes that have been added to the staging area, use:
git reset
## This command will move all staged changes back to the working directory.

#2. Discard Changes in Working Directory
##To discard changes in all tracked files:
git restore .
##This will revert the changes in your working directory to match the last commit.
```

9. what if i have made commit in main branch

```
#If you've committed changes to the main branch and you want to undo those
commits, there are several approaches you can take, depending on whether you want
to keep the changes or discard them completely.

##1. Undo the Last Commit but Keep Changes
###If you want to undo the last commit but keep the changes in your working
directory (so you can adjust them or re-commit):

git reset --soft HEAD~1

###This command moves the branch pointer back one commit but leaves your changes
in the working directory and staging area.

##2. Undo the Last Commit and Discard Changes
###If you want to undo the last commit and discard the changes entirely:
```

```
git reset --hard HEAD~1

###This command moves the branch pointer back one commit and discards all changes
in the working directory.

##3. Undo Multiple Commits

###To undo more than one commit, adjust the number in HEAD~n where n is the number
of commits you want to undo. For example, to undo the last 3 commits:

####Keep Changes:
git reset --soft HEAD~3

####Discard Changes:
git reset --hard HEAD~3
```

2. Api Descptions

1. guest APIs

1. `UserController (@RequestMapping="/user")`

1. login `User`

- **URL** - http://host:port/user/login
- **Method** - POST
- **payload** - `SignInRequestDto`
- **Successful Resp** - SC 200 - `HttpStatus.OK` - `SignInResponseDto`
- **Error resp**
  - **if isdeleted = 0** - 403 - `HttpStatus.FORBIDDEN` - `CustomAuthorizationException("User is Deactivated!")`
  - **if user is not found** - `CustomAuthenticationException("Invalid Email or Password !")`

2. update `User` Basic Details By `UserId`

- **URL** - http://host:port/guest/update/{guestId}
- **Method** - PUT
- **payload** - `UpdateBasicUserDetailsDto`
- **Successful Resp** - SC 201 - `UpdateBasicUserDetailsDto`
- **Error resp** - SC 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("Invalid User Id !")`

3. get `User` details (basic details + driving license)

- **URL** - http://host:port/user/profile/{guestId}
- **Method** - Get

- **Successful Resp** - SC 201 `UserDetailsResponseDto`
- **Error resp** -
  - SC 400 `ConstraintViolationException("issue date is before expiry date")`
  - SC 404 `ResourceNotFoundException("invalid user id")`

4. delete `User`

- **URL** - http://host:port/user/delete/{userId}
- **Method** - PATCH
- **Successful Resp** - SC 201 `ApiResponseDto`
- **Error resp** -
  - SC 404 - `ResourceNotFoundException("Invalid User Id")`
  - SC 400 - `CustomBadRequestException("Admin users cannot be deleted.")`

2. `GuestController (@RequestMapping="/guest")`

3. `CarController (@RequestMapping="/car")`

1.Get CarCards By City,pickupDateTime,dropOffDateTime - public api

- **URL** - http://host:port/car/get_cars_by_city?city=value,pickupDateTime,dropOffDateTime
- **Method** - GET
- **Successful Resp** - SC 201 * Successful Resp - SC 201 ArrayList CarsCardDetailsDto(brand,model,transmissionTypeEnum,fuelTypeEnum,seatingCapacity,noOfTrips,carPricePerHr,carPricePerDay)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

2.Get CarCards Details By CarId(Car + Features)

- **URL** - http://host:port/car/get_specific_car_details/{carId}
- **Method** - GET
- **Successful Resp** - SC 201 CompleteCarDetailsDto + mesg (ApiResponse) *ArrayList CarsCardDetailsDto(brand,model,transmissionTypeEnum,fuelTypeEnum,seatingCapacity,noOfTrips,carPricePerHr,carPricePerDay)
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

4. `BookingController (@RequestMapping="/booking")`

1. add `Booking` by `guestId`, `CarId`, `guestAddressId` of guest

- **URL** - http://host:port/booking/{guestId}/{carId}/{guestAddressId}
- **Method** - POST
- **payload** - `BookingDto`
- **Successful Resp** - SC 201 `BookingResponseDto`
- **Error resp** - SC 400 `RuntimeException("Pickup Date is invalid !")`
  - SC 400 `RuntimeException("Guest Not Found!") ;`
  - SC 400 `RuntimeException("Address not found!") ;`
  - SC 400 `RuntimeException("Car Listing not found!") ;`

2. update `Booking` after payment is made

- **URL** - http://host:port/confirm_booking/{booking_id}
- **Method** - Patch
- **payload** - `PaymentRequestDto`
- **Successful Resp** - SC 201 - `PaymentResponseDto`
- **Error resp**
  - SC 400 - `RuntimeException("Booking not found")`
  - SC 400 - `ApiException("Internal Server Error")`

3. get upcomming `Bookings` by `UserId`

- **URL** - http://host:port/booking/upcomming_booking/{userId}
- **Method** - Get
- **Successful Resp** - SC 201 - `List<BookingCardDto>`
- **Error resp**
  - SC 400 - `ApiException("Internal Server Error")`
  - SC 400 - `RuntimeException("User not found")`

4. get past `Bookings` by `UserId`

- **URL** - http://host:port/past_booking/{userId}
- **Method** - Get
- **Successful Resp** - SC 201 BookingCardDto
- **Error resp**
  - SC 400 - `ApiException("Internal Server Error")`
  - SC 400 - `RuntimeException("User not found")`

5. `DrivingLicenseController(@RequestMapping="/dl")`

1. update `DrivingLicense` By `UserId`

- **URL** - http://host:port/dl/{userId}/update/{dlId}
- **Method** - PUT
- **payload** - `DrivingLicenseDto`
- **Successful Resp** - SC 200 - `DrivingLicenseDto`
- **Error resp**
  - SC 400 - `ResourceNotFoundException("invalid user id")`
  - SC 409 - `ConflictException`

2. host APIs

1. `UserController (@RequestMapping="/user")`

1. register `User` with basic details

- **URL** - http://host:port/user/register_basic
- **Method** - POST

- **payload** - `RegisterUserReqDto`
- **Successful Resp** - SC 201 HttpStatus.CREATED `RegisterUserResDto`
- **Error resp** -
  - **if already registerd** -
    - SC 409 , conflict
    - **exception** -
      - `ConflictException("user with given email and mobile already registered");`
      - `ConflictException("user with given email already registered");`
      - `ConflictException("user with given mobile already registered");`
  - **if password and confirm password doesn't match** -
    - SC 400 , bad request
    - **exception** - `ConstraintViolationException("password mismatch", null);`
  - **if expiry date is before creation date**
    - SC 400 , bad request
    - **exception** - `ConstraintViolationException("expiry should come after issue date of dl ", null);`

2. register `User` with `DrivingLicense`

- **URL** - http://host:port/user/register_with_dl
- **Method** - POST
- **payload** - `RegisterUserWithDlReqDto`
- **Successful Resp** - SC 201 `HttpStatus.CREATED` - `RegisterUserWithDlResDto`
- **Error resp** -
  - **if already registerd** -
    - SC 409 , conflict
    - **exception** -
      - `ConflictException("user with given email and mobile already registered");`
      - `ConflictException("user with given email already registered");`
      - `ConflictException("user with given mobile already registered");`
    - **if password and confirm password doesn't match** -
      - SC 400 , bad request
      - **exception** - `ConstraintViolationException("password mismatch", null);`
    - **if expiry date is before creation date**
      - SC 400 , bad request
      - **exception** - `ConstraintViolationException("expiry should come after issue date of dl ", null);`

1. `HostController(@RequestMapping="/host")`

2. `CarListingController(@RequestMapping=/car_listing)`

1. add `CarListing` by `hostId` and `hostAddressId`

- **URL** - http://host:port/car_listing/{hostId}/{hostAddressId}

- **Method** - POST
- **payload** - `AddCarListingDto`
- **Successful** Resp - SC 201 - `HttpStatus.CREATED` - `AddCarListingResponseDto`
- **Error resp**
  - 409 - `HttpStatus.CONFLICT` - `ConflictException("registration no alreday exists")`
  - 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("invalid host id")`
  - 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("invalid address id")`

2. get `CarListing` by `carListingId`

- **URL** - http://host:port/car_listing/{carListingId}
- **Method** - GET
- **payload** -
- **Successful** Resp - SC 200 - `HttpStatus.OK` - `GetCarListingResponseDto`
- **Error resp**
  - 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("car_listing_doesn't exist")`

3. update `CarListing` by `carListingId`

- **URL** - http://host:port/car_listing/{carListingId}
- **Method** - PATCH
- **payload** - `UpdateCarListingDto`
- **Successful** Resp - SC 200 - `HttpStatus.OK` - `GetCarListingResponseDto`
- **Error resp**
  - 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("car_listing_doesn't exist")`

3. `AddressController(@RequestMapping=/address)`

1. add `Address` By `UserId`

- **URL** - http://host:port/address/{userId}
- **Method** - POST
- **payload** - `AddressDto`
- **Successful** Resp - SC 201 - `HttpStatus.CREATED` - `AddressResDto`
- **Error resp** - SC 400 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("invalid id")`

2. get `Address` by `addressId`

- **URL** - http://host:port/address/{addressId}
- **Method** - GET
- **payload** -
- **Successful** Resp - SC 200 OK, `AddressDto`
- **Error resp** - 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("invalid address id")`

3. get `AddressList` by `UserId`

- **URL** - http://host:port/address/get_all/{userId}
- **Method** - GET
- **payload** -
- **Successful** Resp - SC 201 `List<AddressDto>`
- **Error resp** - SC 400 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("invalid user id")`

4. update `Address` by `addressId`

- **URL** - http://host:port/user/{userId}/address/{addressId}
- **Method** - PUT
- **payload** -
- **Successful** Resp - SC 201 `AddressDto`
- **Error resp**
  - 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("invalid address id")`
  - 400 - `HttpStatus.BAD_REQUEST` - `ConstraintViolationException("address id mismatch", null)`

4. delete `Address` by `AddressId`

- **URL** - http://host:port/user/{userId}/address/{addressId}
- **Method** - PATCH
- **payload** -
- **Successful** Resp - 200 - `HttpStatus.OK` - `DeleteAddressResDto`
- **Error resp**
  - 404 - `HttpStatus.NOT_FOUND` - `ResourceNotFoundException("invalid address id")`

4. `BookingController(@RequestMapping=/booking)`

3. admin APIs

1. `AdminController(@RequestMapping="/admin")`

2. `GuestController (@RequestMapping="/guest")`

1. get all guests

- **URL** - http://host:port/guest/get_all_guests
- **Method** - Get
- **Successful Resp** - SC 200 - `ArrayList<GetAllGuestDto>`
- **Error resp** - SC 400 , error mesg -wrapped in DTO(ApiResponse)

3. `HostController(@RequestMapping="/host")`

4. `CarController(@RequestMapping="/car")`

3. **Anootated Entities**

0. BaseEntity

~~~
    ```java
            package com.rentmycar.entity;

            @Setter
            @Getter
            @ToString
            @MappedSuperclass
            public class BaseEntity {

                @Id
                @GeneratedValue(strategy = GenerationType.IDENTITY)
                private Long id;

                @Column(updatable = false)
                @CreationTimestamp
                private LocalDate createdOn;

                @UpdateTimestamp
                private LocalDateTime updatedOn;

            }
    ```
~~~

1. User

~~~
```java
    package com.rentmycar.entity;

    @Entity
    @Getter
    @Setter
    @ToString
    @Table(name = "user", uniqueConstraints = {
            @UniqueConstraint(columnNames = {"email","mobile","roleEnum"})})
    public class User extends BaseEntity {

        @Column(length = 25, nullable = false)
        private String firstName;

        @Column(length = 25, nullable = false)
        private String lastName;

        @Column(length = 12, nullable = false)
        private String mobile;

        @Column(length = 20, nullable = false)
        private String email;
~~~

```
        @Column(length = 70, nullable = false)
        private String password;

        @Column(columnDefinition = "boolean default false", nullable = false)
        private Boolean isDeleted;

        @Enumerated(EnumType.STRING)
        @Column(length = 5)
        private UserRoleEnum roleEnum;

    //
************************************************************************************
*******

        // User 1<---->* Address
        @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval =
true) // mandatory , otherwise

// ,MappingException !
        private List<Address> addressList = new ArrayList<>();

        public void addAddress(Address address) {
            addressList.add(address);
            address.setUser(this);
        }

        public void deleteAddress(Address address) {
            addressList.remove(address);
            address.setUser(null);
        }

    //
************************************************************************************
*********

        // Guest 1<---->* Booking
        @OneToMany(mappedBy = "guest", cascade = CascadeType.ALL, orphanRemoval =
true)
        private List<Booking> bookingList = new ArrayList<>();

        public void addBooking(Booking booking) {
            bookingList.add(booking);
            booking.setGuest(this);
        }

    //
************************************************************************************
**********

        // Guest 1 <------> *Review
        @OneToMany(mappedBy = "guest", cascade = CascadeType.ALL, orphanRemoval =
true)
        private List<Review> reviewList = new ArrayList<Review>();

        public void addReview(Review review) {
            reviewList.add(review);
            review.setGuest(this);
        }


//*********************************************************************************
**************

        // User 1 <------> 1 DrivingLicense
        @OneToOne(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval =
true)
        private DrivingLicense drivingLicense;

        public void adddrivingLicense(DrivingLicense drivingLicense) {
            if (drivingLicense == null) {
                throw new RuntimeException("driving license is null");
            }
            this.drivingLicense = drivingLicense;
            drivingLicense.setUser(this);
        }

    //
***********************************************************************************
**********

        // Host 1 <---------> * CarHostAddressPricing
        @OneToMany(mappedBy = "host", cascade = CascadeType.ALL, orphanRemoval =
true)
        private List<CarHostAddressPricing> carHostAddressPricingList = new
ArrayList<CarHostAddressPricing>();

        public void addCarHostAddressPricing(CarHostAddressPricing
carHostAddressPricing) {
            this.carHostAddressPricingList.add(carHostAddressPricing);
            carHostAddressPricing.setHost(this);

//*********************************************************************************
***************
        }
```

1.1. UserRoleEnum

~~~
```java
    package com.rentmycar.entity;
~~~

~~~

    public enum UserRoleEnum {
        GUEST,ADMIN,HOST
    }

```
~~~

2. Address

~~~
```java
        package com.rentmycar.entity;

        @Entity
        @Table
        @Getter
        @Setter
        @ToString

        public class Address extends BaseEntity {

            @Column(name = "adr_line1", length = 200, nullable = false)
            private String adrLine1; // varchar(100) not null

            @Column(name = "adr_line2", length = 200)
            private String adrLine2; // varchar(100)

            @Column(length = 20, nullable = false)
            private String country; // varchar(20),not null

            @Column(length = 20, nullable = false)
            private String state; // varchar(20),not null

            @Column(length = 20, nullable = false)
            private String city; // varchar(20),not null

            @Column(nullable = false, columnDefinition = "char(10)")
            private String pincode; // char(10),not null

            @Column(columnDefinition = "boolean default false", nullable = false)
            private Boolean isDeleted;


//*********************************************************************************
*******************
            // many -> Address * <----> 1 User
            @ManyToOne(fetch = FetchType.LAZY) // mandatory
            @JoinColumn(name = "guest_id", nullable = false)
            private User user;
~~~

```
//***
****

        }
```
```

3. Booking

```
```java
    package com.rentmycar.entity;

    @Getter
    @Setter
    @ToString
    @Entity
    @Table
    public class Booking extends BaseEntity {
        @Column(nullable = false)
        LocalDateTime pickUp; // DateTime,not null

        @Column(nullable = false)
        LocalDateTime dropOff; // DateTime,not null

        @Enumerated(EnumType.STRING)
        @Column(nullable = false, length = 10)
        @ColumnDefault("'PENDING'")
        BookingStatusEnum bookingStatusEnum; // not null,default - PENDING

        @Column(nullable = false)
        Double amount; // double,not null

        @Column(nullable = false)
        LocalDateTime paymentDateTime; // DateTime

        @Column(nullable = false, columnDefinition = "char(20)")
        String transactionId; // char (#### 20.


//*********************************************************************************
********************

        // Booking * <---> 1 Guest
        @ManyToOne(fetch = FetchType.LAZY)
        @JoinColumn(name = "guest_id", nullable = false)
        private User guest;


//*********************************************************************************
********************

        // Booking * -------> 1 Address
        @ManyToOne
        @JoinColumn(nullable = false)
        private Address guestAddress;


//*********************************************************************************
********************

        // Booking * <-----------> 1 CarHostAddrssPricing
        @ManyToOne(fetch = FetchType.LAZY)
        private CarHostAddressPricing carHostAddressPricing;


//*********************************************************************************
********************
    }
```
```

3.1. BookingStatusEnum

```
package com.rentmycar.entity;

public enum BookingStatusEnum {
    CANCEL, PENDING, SUCCESS,FAILED
}
```

4. Car

```
```java
    package com.rentmycar.entity;

    @Table(name = "car")
    @Entity
    public class Car extends BaseEntity {

        @Column(length = 20, nullable = false)
        private String brand; // varchar(20),not null

        @Column(length = 20, nullable = false)
        private String model; // varchar(20),not null

        @Enumerated(EnumType.STRING)
        @Column(length = 10)
```

~~~
        private FuelTypeEnum fuelTypeEnum; // varchar(10), not null

        @Column(columnDefinition = "TINYINT", nullable = false )
        private int seatingCapacity; // tinyint(1-6),not null

    //*************************************************************
        // Car * ------> 1 CarFeatures
        // many cars can have same features
        @ManyToOne
        private CarFeatures carFeatures;

    //*************************************************************
    }
```
~~~

4.1. FuelTypeEnum

~~~
```java

    package com.rentmycar.entity;

    public enum FuelTypeEnum {
        PETRO,DIESEL,ELECTRIC,CNG
    }
```
~~~

5. CarFeatures

~~~
```java
    package com.rentmycar.entity;

    @ToString
    @Setter
    @Getter
    @Entity
    @Table(name = "car_features")
    public class CarFeatures extends BaseEntity{

        @Column(columnDefinition = "boolean default false")
        private Boolean hasUsbCharger;  // Boolean,default-false

        @Column(columnDefinition = "boolean default false")
        private Boolean hasBluetooth;   // Boolean,default-false

        @Column(columnDefinition = "boolean default false")
        private Boolean hasPowerSteering;   // Boolean,default-false

~~~

~~~
        @Column(columnDefinition = "boolean default false")
        private Boolean hasAirBags;     // Boolean,default-false

        @Column(columnDefinition = "boolean default false")
        private Boolean hasAbs;     // Boolean,default-false

        @Column(columnDefinition = "boolean default false")
        private Boolean hasAc;  // Boolean,default-false
    }
```
~~~

6. CarHostAddressPricing

~~~
```java
    package com.rentmycar.entity;

    @Table
    @Entity
    @Getter
    @Setter
    public class CarHostAddressPricing extends BaseEntity {
        int kmDriven; // int,not null

        @Column(columnDefinition = "TINYINT")
        int fuelMeter; // tinyint(1-10),not null

        @Column(length = 20)
        String carImage; // varchar(20),

        @Column
        int noOfTrips;

        @Column(columnDefinition = "BOOLEAN DEFAULT TRUE")
        Boolean isAvailable; // Boolean,default-true

        @Column(columnDefinition = "BOOLEAN DEFAULT FALSE")
        Boolean isDeleted; // Booleane,default-false

        @Column(columnDefinition = "BOOLEAN DEFAULT FALSE")
        Boolean isApproved; // Boolean,default-false

        @Column(length = 20, nullable = false)
        String registrationNo; // varchar(20),not null

        @Column(columnDefinition = "TINYINT NOT NULL")
        int spareTyreCount; // tinyint,not null

    //****************************************************************************
~~~

~~~
        // CarHostAddressPricing * ---------> 1 Car
        @ManyToOne
        @JoinColumn(nullable = false)
        private Car car;

    //**************************************************************************

        // CarHostAddressPricing * -------> 1 CarPricing
        @ManyToOne
        @JoinColumn(nullable = false)
        private CarPricing carPricing;;

    //**************************************************************************

        // CarHostAddressPricing * --------> 1 Address
        @ManyToOne
        @JoinColumn(nullable = false)
        private Address address;

    //**************************************************************************

        // CarHostAddressPricing * <------------> 1 Host
        @ManyToOne
        @JoinColumn(nullable = false)
        private User host;

    //***************************************************************************

        // CarHostAddressPricing 1 <-------------> * Booking
        @OneToMany(mappedBy = "carHostAddressPricing", cascade = CascadeType.ALL,
orphanRemoval = true)
        private List<Booking> bookingList = new ArrayList<Booking>();

        public void addBooking(Booking booking) {
            bookingList.add(booking);
            booking.setCarHostAddressPricing(this);
        }

    //***************************************************************************

    }
```
~~~

7. CarPricing

~~~
```java
    package com.rentmycar.entity;

    @ToString
~~~

```
    @Getter
    @Setter
    @Entity
    @Table(name = "car_pricing")
    public class CarPricing extends BaseEntity{

        @Column(nullable = false)
        Double pricePerHr;  // Boolean,not null

        @Column(nullable = false)
        Double pricePerDay;     // Boolean,not null

        @Column(nullable = false)
        Double securityDeposit;     // double,not null
    }
```
```

## 8. DrivingLicense

```

    ```java
        package com.rentmycar.entity;

        @Getter
        @Setter
        @ToString
        @Entity
        @Table(name = "driving_license")
        public class DrivingLicense extends BaseEntity {

            @Column(name = "license_no", length = 25, unique = true, nullable =
false)
            private String drivingLicenseNo;

            @Column(name = "license_issue_date", nullable = false)
            private LocalDate issueDate;

            @Column(name = "license_expiry_date", nullable = false)
            private LocalDate expirationDate;

            @Enumerated(EnumType.STRING)
            @Column(length = 10, nullable = false)
            private LicenseClassEnum licenseClassEnum;


//*********************************************************************************
******************
            // DrivingLicense 1 <----> 1 User
            @OneToOne(fetch = FetchType.LAZY) // mandatory
            @JoinColumn(name = "user_id", nullable = false)
```

```
            private User user;


  //*********************************************************************************
  *******************

        }
    ```
```

### 8.1. LicenseClassEnum

```
    ```java
        package com.rentmycar.entity;

        public enum LicenseClassEnum {

            A, // ("Light Motor Vehicle"),
            B, // ("Motorcycles up to 500cc"),
            C, // ("Heavy Motor Vehicle"),
            D// ("Transport Vehicle")
        }
    ```
```

### 9. Review

```
    ```java
            package com.rentmycar.entity;

            @Entity
            @Table(name = "review")
            @Data
            public class Review extends BaseEntity{
                @Column(nullable = false)
                private byte rating; // tinyint,not-null

                @Column(length = 50)
                private String reviewText; // varchar(50)

            //
  ***************************************************************************
                // Review * <-----------> 1 Guest
                @ManyToOne(fetch = FetchType.LAZY)
                @JoinColumn(name = "guest_id", nullable = false)
                private User guest;

            //
  **************************************************************************
```

```
                // Review * -----------> 1 CarHostAddressPricing
                @ManyToOne
                @JoinColumn(nullable = false)
                private CarHostAddressPricing carHostAddressPricing;


//******************************************************************************

            }
    ```
```

10. Transaction

```
    ```java
        // YET TO BE DECIDED
        @Entity
        public class Transaction {
            Double ammount; // double
            PaymentStatusEnum paymentStatusEnum;
            DateTime paymentDate; // DateTime
        }
    ```
```

## 4. **ERDiagram**
```