Create Chaincode for Car Showroom

Steps to be followed:

1. Creating Car.java file
2. Creating CarTransfer.java file
3. Compiling the chaincode in the Terminal

Step 1: Creating Car.java file

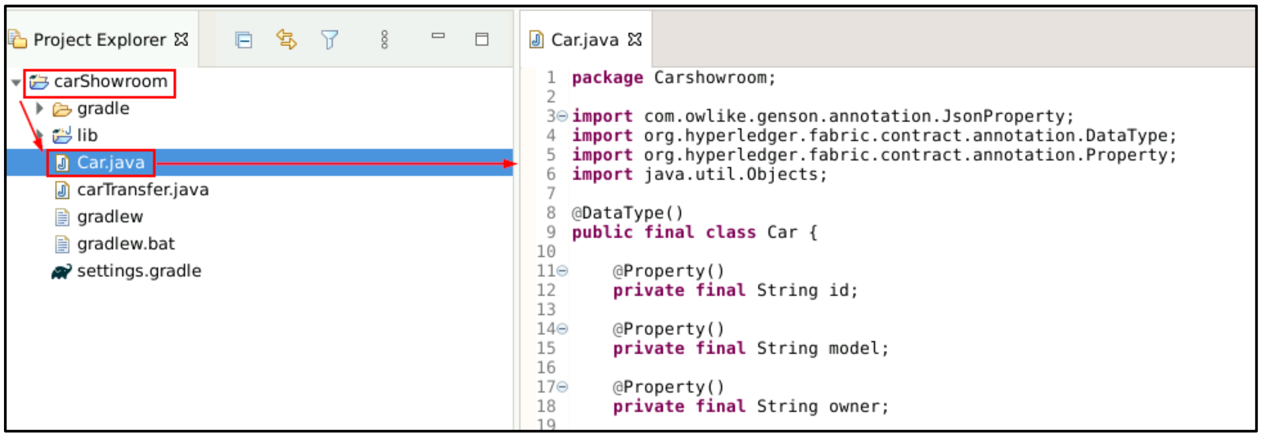
1.1 Right-click on the project and navigate to **New >> Class** to create **Car.java** class

1.2 Add the following code in **Car.java** file:

```
package CarShowroom;

import com.owlike.genson.annotation.JsonProperty;
import org.hyperledger.fabric.contract.annotation.DataType;
import org.hyperledger.fabric.contract.annotation.Property;
import java.util.Objects;

@DataType()
public final class Car {

        @Property()
        private final String id;

        @Property()
        private final String model;

        @Property()
        private final String owner;

        @Property()
        private final String value;
```

```
	public String getId() {
		return id;
	}

	public String getModel() {
		return model;
	}
	public String getOwner() {
		return owner;
	}

	public String getValue() {
		return value;
	}

	public Car(@JsonProperty("id") final String id, @JsonProperty("model") final
String model, @JsonProperty("owner") final String owner,
			@JsonProperty("value") final String value) {
		this.id = id;
		this.model = model;
		this.owner = owner;
		this.value = value;
	}

	@Override
	public boolean equals(final Object obj) {
		if (this == obj) {
			return true;
		}

		if ((obj == null) || (getClass() != obj.getClass())) {
			return false;
		}

		Car other = (Car) obj;
```

```
                return Objects.deepEquals(new String[] { getId(), getModel(), 
getOwner(), getValue() },
                                new String[] { other.getId(), other.getModel(), 
other.getOwner(), other.getValue() });
        }

        @Override
        public int hashCode() {
                return Objects.hash(getId(), getModel(), getOwner(), getValue());
        }

        @Override
        public String toString() {
                return this.getClass().getSimpleName() + "@" + 
Integer.toHexString(hashCode()) + " [id=" + id + ", model=" + model
                                + ", owner=" + owner + ", value=" + value + "]";
        }

}
```



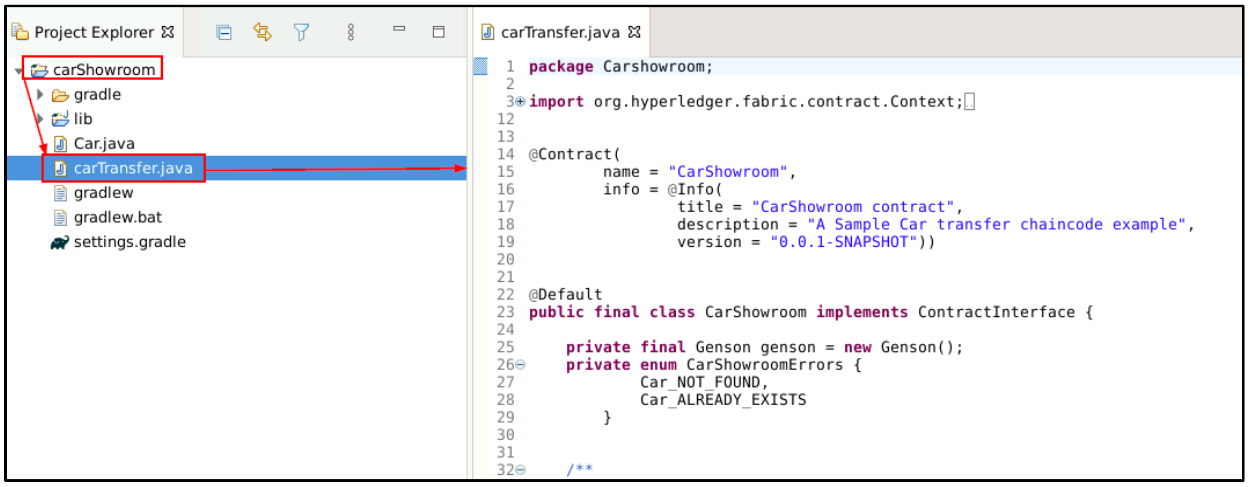
```
package Carshowroom;

import com.owlike.genson.annotation.JsonProperty;
import org.hyperledger.fabric.contract.annotation.DataType;
import org.hyperledger.fabric.contract.annotation.Property;
import java.util.Objects;

@DataType()
public final class Car {

    @Property()
    private final String id;

    @Property()
    private final String model;

    @Property()
    private final String owner;
```

Step 2: Creating CarTransfer.java file

2.1 Follow Stap 1.1 to create **CarTransfer.java** class and add the following code in it:

```
package CarShowroom;


import org.hyperledger.fabric.contract.Context;
import org.hyperledger.fabric.contract.ContractInterface;
import org.hyperledger.fabric.contract.annotation.Contract;
import org.hyperledger.fabric.contract.annotation.Default;
import org.hyperledger.fabric.contract.annotation.Info;
import org.hyperledger.fabric.contract.annotation.Transaction;
import org.hyperledger.fabric.shim.ChaincodeException;
import org.hyperledger.fabric.shim.ChaincodeStub;
import com.owlike.genson.Genson;


@Contract(
      name = "CarShowroom",
      info = @Info(
            title = "CarShowroom contract",
            description = "A Sample Car transfer chaincode example",
            version = "0.0.1-SNAPSHOT"))


@Default
public final class CarTransfer implements ContractInterface {

         private final Genson genson = new Genson();
         private enum CarShowroomErrors {
               Car_NOT_FOUND,
               Car_ALREADY_EXISTS
            }



         /**
     * Add some initial properties to the ledger
     *
     * @param ctx the transaction context
     */
    @Transaction()
    public void initLedger(final Context ctx) {

       ChaincodeStub stub= ctx.getStub();
```

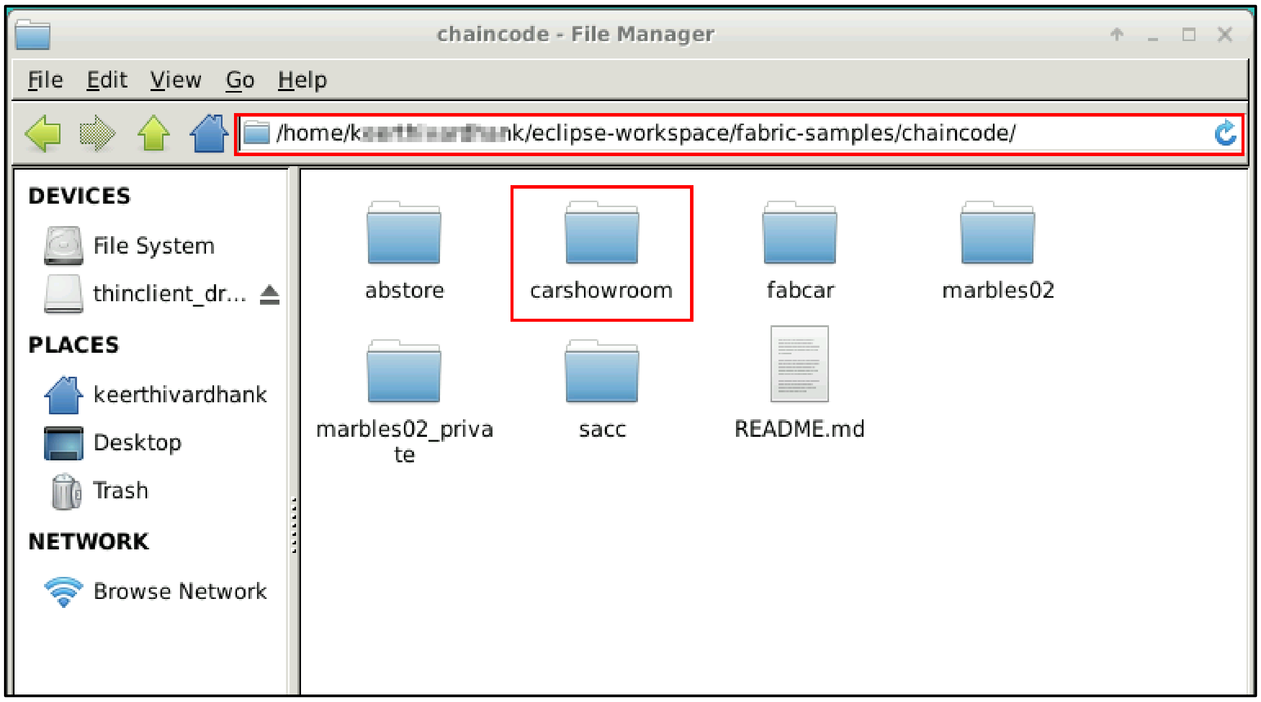
```
        Car Car = new Car("1", "Maruti","Mark","6756");

        String CarState = genson.serialize(Car);

        stub.putStringState("1", CarState);
    }



    /**
     * Add new Car on the ledger.
     *
     * @param ctx the transaction context
     * @param id the key for the new Car
     * @param model the model of the new Car
     * @param ownername the owner of the new Car
     * @param value the value of the new Car
     * @return the created Car
     */

    @Transaction()
    public Car addNewCar(final Context ctx, final String id, final String model,
            final String ownername, final String value) {

             ChaincodeStub stub = ctx.getStub();

        String CarState = stub.getStringState(id);

        if (!CarState.isEmpty()) {
            String errorMessage = String.format("Car %s already exists", id);
            System.out.println(errorMessage);
            throw new ChaincodeException(errorMessage,
CarShowroomErrors.Car_ALREADY_EXISTS.toString());
        }

        Car Car = new Car(id, model, ownername,value);

        CarState = genson.serialize(Car);
        stub.putStringState(id, CarState);

        return Car;
    }



            /**
                * Retrieves a Car based upon Car Id from the ledger.
                *
                * @param ctx the transaction context
                * @param id the key
                * @return the Car found on the ledger if there was one
                */
            @Transaction()
                public Car queryCarById(final Context ctx, final String id) {
                    ChaincodeStub stub = ctx.getStub();
                    String CarState = stub.getStringState(id);

                    if (CarState.isEmpty()) {
                        String errorMessage = String.format("Car %s does not exist", id);
                        System.out.println(errorMessage);
                        throw new ChaincodeException(errorMessage,
CarShowroomErrors.Car_NOT_FOUND.toString());
                    }

                    Car Car = genson.deserialize(CarState, Car.class);
                    return Car;
                }

        /**
                * Changes the owner of a Car on the ledger.
                *
                * @param ctx the transaction context
                * @param id the key
                * @param newOwner the new owner
                * @return the updated Car
                */
                @Transaction()
        public Car changeCarOwnership(final Context ctx, final String id, final String
newCarOwner) {
          ChaincodeStub stub = ctx.getStub();

          String CarState = stub.getStringState(id);

          if (CarState.isEmpty()) {
            String errorMessage = String.format("Car %s does not exist", id);
            System.out.println(errorMessage);
            throw new ChaincodeException(errorMessage,
CarShowroomErrors.Car_NOT_FOUND.toString());
          }

          Car Car = genson.deserialize(CarState, Car.class);

          Car newCar = new Car(Car.getId(), Car.getModel(), newCarOwner,
Car.getValue());

          String newCarState = genson.serialize(newCar);

          stub.putStringState(id, newCarState);

          return newCar;
        }
}
```



```java
package Carshowroom;

import org.hyperledger.fabric.contract.Context;


@Contract(
        name = "CarShowroom",
        info = @Info(
                title = "CarShowroom contract",
                description = "A Sample Car transfer chaincode example",
                version = "0.0.1-SNAPSHOT"))


@Default
public final class CarShowroom implements ContractInterface {

    private final Genson genson = new Genson();
    private enum CarShowroomErrors {
            Car_NOT_FOUND,
            Car_ALREADY_EXISTS
        }


    /**
```

Step 3: Compiling the project

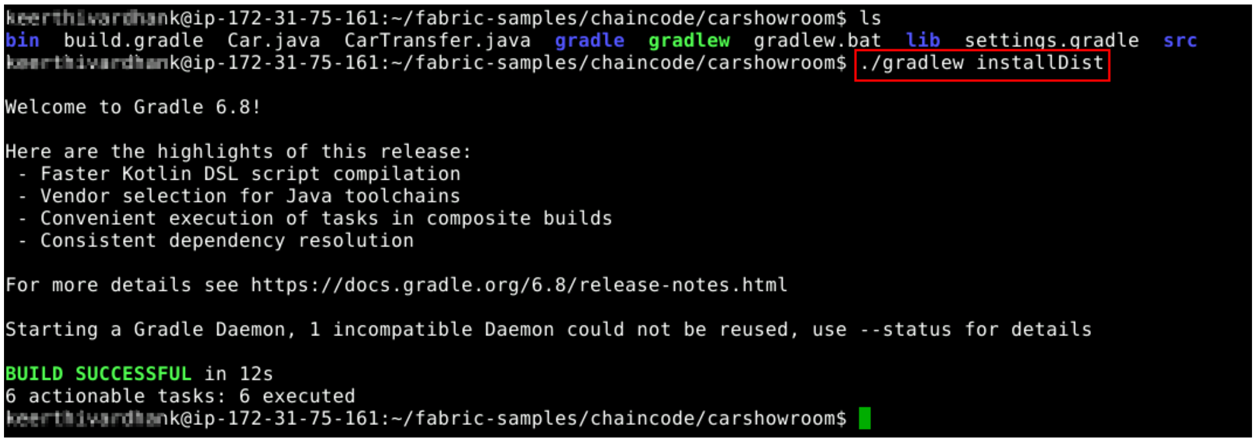
3.1 To compile the project, ensure that the **CarShowroom** project is under the **eclipse-workspace/fabric-samples/chaincode** folder. If not, then move it under the **eclipse-workspace/fabric-samples/chaincode** folder



3.2 To compile the chaincode, run the command on the terminal:

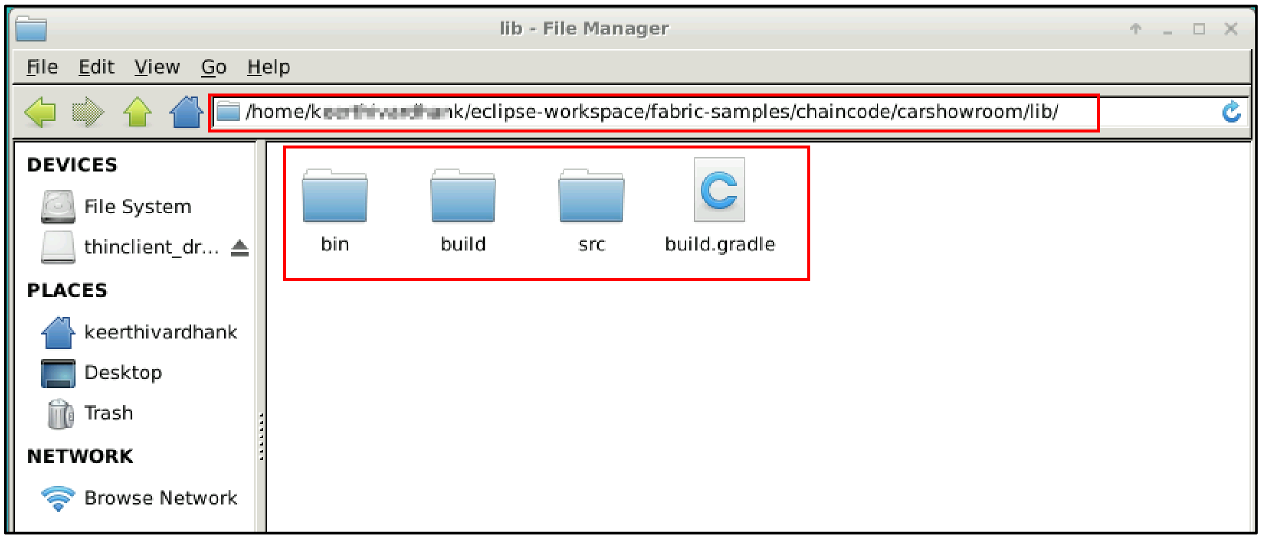
cd eclipse-workspace/fabric-samples/chaincode/Carshowroom

./gradlew installDist



```
[redacted]k@ip-172-31-75-161:~/fabric-samples/chaincode/carshowroom$ ls
bin  build.gradle  Car.java  CarTransfer.java  gradle  gradlew  gradlew.bat  lib  settings.gradle  src
[redacted]k@ip-172-31-75-161:~/fabric-samples/chaincode/carshowroom$ ./gradlew installDist

Welcome to Gradle 6.8!

Here are the highlights of this release:
 - Faster Kotlin DSL script compilation
 - Vendor selection for Java toolchains
 - Convenient execution of tasks in composite builds
 - Consistent dependency resolution

For more details see https://docs.gradle.org/6.8/release-notes.html

Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for details

BUILD SUCCESSFUL in 12s
6 actionable tasks: 6 executed
[redacted]k@ip-172-31-75-161:~/fabric-samples/chaincode/carshowroom$
```

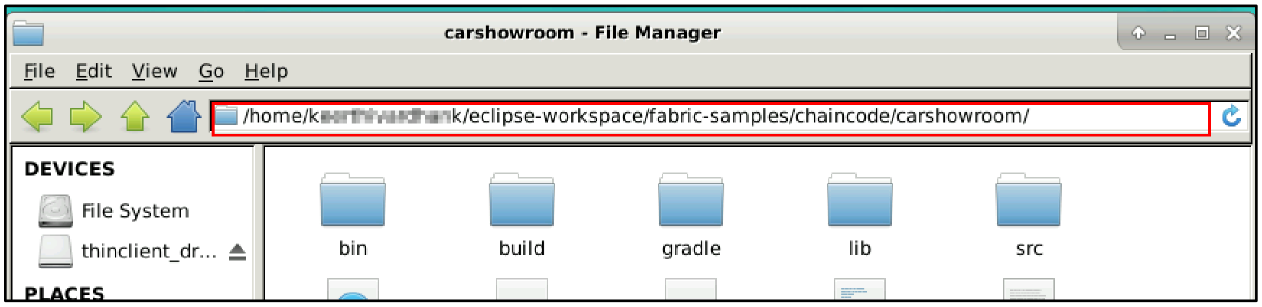
3.3 Copy the folders and files from the library **lib**



lib - File Manager

/home/k[redacted]k/eclipse-workspace/fabric-samples/chaincode/carshowroom/lib/

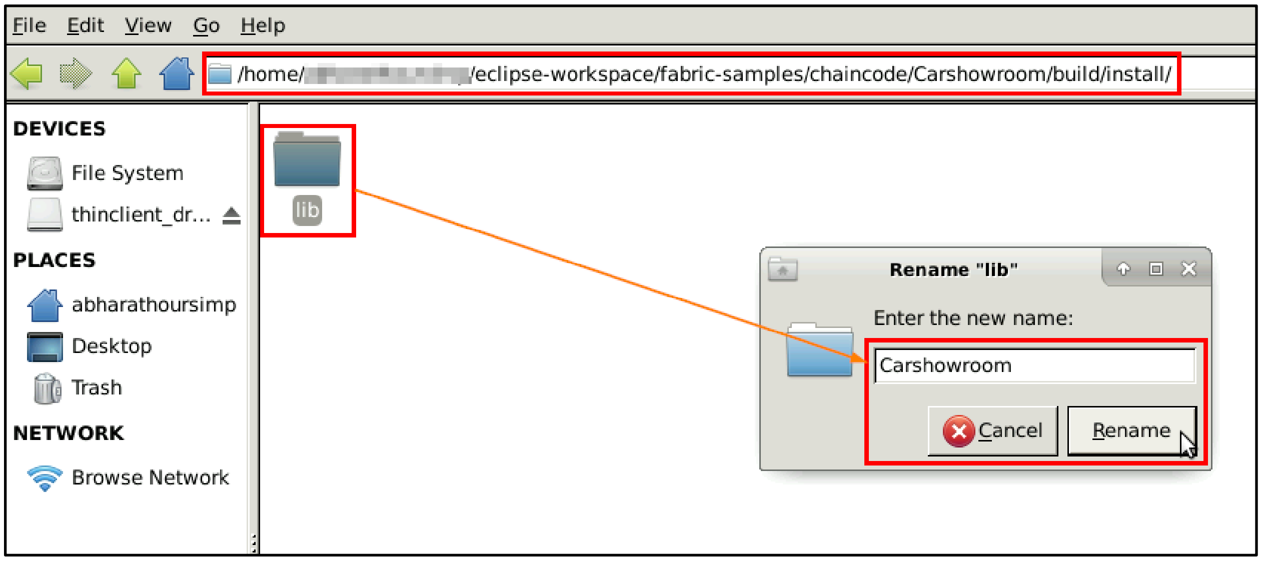
3.4 Paste it in the main folder **Carshowroom**



carshowroom - File Manager

/home/k[redacted]k/eclipse-workspace/fabric-samples/chaincode/carshowroom/

3.5 In the root project folder, navigate to **build >> install** and rename lib folder to **Carshowroom**



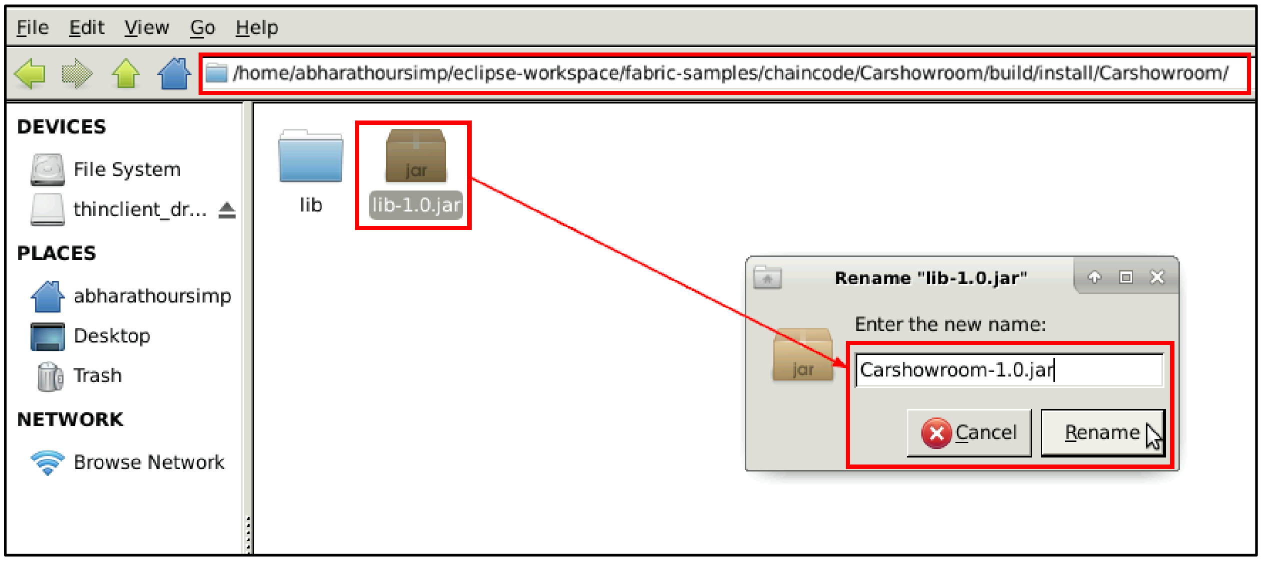
/home/[redacted]/eclipse-workspace/fabric-samples/chaincode/Carshowroom/build/install/

Rename "lib"

Enter the new name:

Carshowroom

3.6 In the root project folder, go to **build >> install >> Carshowroom** folder and rename **lib.tar.gz file** to **Carshowroom-1.0.jar**



/home/abharathoursimp/eclipse-workspace/fabric-samples/chaincode/Carshowroom/build/install/Carshowroom/

Rename "lib-1.0.jar"

Enter the new name:

Carshowroom-1.0.jar