System Description: The system simulates multiple cars arriving simultaneously and requesting parking. A shared parking pool manages these requests, while parking agents process and park the cars safely.

Class Description:-

i) RegisteredParking: Represents a parking request made by a car.

ii) ParkingPool: A shared, synchronized queue that stores parking requests.

iii) Parking Agent: A thread that continuously retrives and parks cars from the pool.

iv) MainClass: Simulates multiple cars arriving concurrently.

#Implementation

```
import java.util.LinkedList;
import java.util.Queue;
class RegistrarParking{
        private String carNumber;
        public RegistrarParking(String carNumber){
            this.carNumber = carNumber;
        }
        public String getCarNumber(){
            return carNumber;
        }
    }
class ParkingPool{
    private Queue<RegistrarParking> queue = new LinkedList();
    private public synchronized void addCar(RegistrarParking
                                            car){
        queue.add(Car);
        notify();
    }
public synchronized RegistrarParking getCar() throws
                InterruptedException{
        while(queue.isEmpty()){
            wait();
        }
        return queue.poll();
    }
}
```

```
class ParkingAgent extends Thread{
    private ParkingPool pool;
    public ParkingAgent(ParkingPool pool){
        this.pool = pool;
    }
    public void run(){
        try{
        while (true){
            RegistroParking car = pool.getCar();
            System.out.println("Parking Car: "+car.getNumber());
            Thread.sleep(500);
        }
        }catch (InterruptedException e){
        System.out.println("Parking Agent Stopped");
        }
    }
}

public class MainClass{
    public static void main (String[] args){
    ParkingPool pool = new ParkingPool();
    ParkingAgent agent = new ParkingAgent(pool);
    agent.start();
    for(int i=1; i<=5; i++){
    pool.addCar(new RegistroParking("CAR-"+i));
    }
}}
```