```java
package carsInPark;
import java.util.concurrent.Semaphore;

      // Example threads Java car Park 14.10.2021
      // inspired on work Distr Systems 1993 Leic IST (Leic94)

      // Test sample: 12 cars, 1 parking with 1 entry, 1 exit,
      // 2 main rooms, park1(with 3 places) + park2 (also 3 places),
      // separated by a narrow hall (2 places, 1 sense)


public class Car extends Thread{

      private int carId;      private Semaphore s1,s2,s3;


      public Car(int i, Semaphore s1,Semaphore s2,Semaphore s3) {

            carId=i;    this.s1=s1; this.s2=s2; this.s3=s3;
      }


      // basic display, sorry! Next version... maybe ???
      private static void write(int carId, int col,String str) {
            String str2="   "+str;
            if (carId%col == col-1) { // basic visualization
                  System.out.println (str2); System.out.println (" ");
            }
            else  System.out.print (str2);
      }


      private void carWantsToEnterPark1() {
            write(carId, 8,"Car" +carId+" begins.");
            //System.out.println  ("  Auto" +carId+" gaat naar park1.");
            try {Thread.sleep((int)(Math.random()*500));}
            catch(InterruptedException e) {};
      }


      private void carEntersPark1(){

            write(carId, 2,"Car" +carId+" in park1.");
            //System.out.println ("  Auto" +carId+" ga de park1 binnen.");
            try {Thread.sleep((int)(Math.random()*2200));}
            catch(InterruptedException e) {};
      }


      private void carLeavesPark1Text(){

            //write(carId, 2,"Auto" +carId+" verlaat park1 => naar hall.");
            write(carId, 2,"  Car" +carId+" leaves park1 => goes to hall.");
      }


      private void carInHall(){     // not random, but sequential
       try {      Thread.sleep(300); }    catch(InterruptedException e) {};
      }


      private void carGoesToPark2Text(){

            write(carId, 3," Car" +carId+" goes into park2.");
            //System.out.println ("  Auto" +carId+" ga de park2 binnen.");
      }


      private void carInPark2(){
       try {Thread.sleep((int)(Math.random()*2200));} catch(InterruptedException
e) {};
      }


      private void carLeavesPark2Text(){
            //System.out.println ("  Auto" +carId+" verlaat park2.");
            write(carId, 2," Car" +carId+" leaves park2.");
      }


      // watch out text order: threads almost at same time could
      // inverse text order e.g. at release and acquire of semaphores

      public void run() {

            carWantsToEnterPark1();
            try {s1.acquire();}    catch(InterruptedException e) {};

            carEntersPark1();carLeavesPark1Text();
            try {s2.acquire();}    catch(InterruptedException e) {};
                  try {Thread.sleep(100);} catch(InterruptedException e) {};
            s1.release(); // sleep: car exits park1 before another comes in

            carInHall(); carGoesToPark2Text();

            try {s3.acquire();}    catch(InterruptedException e) {};
                  try {Thread.sleep(100);} catch(InterruptedException e) {};
            s2.release(); // sleep: car exits hall before another comes in

            carInPark2();carLeavesPark2Text();
            s3.release();
      }


      public static void main(String[] args) {


            int nSemaphors=3; Semaphore [] s =  new Semaphore[nSemaphors];

            s[0]=new Semaphore(3); // 3 places in park1
            s[1]=new Semaphore(2); // 2 places in hall
            s[2]=new Semaphore(3); // 3 places in park2

            int nCars=15; Car [] c =new Car[ nCars]; // for thread pool

            for(int i=0;i< nCars;i++) {
                  c[i]=new Car(i,s[0],s[1],s[2] );   c[i].start();

            }
                                                // main thread (this) sleeps 5.7
sec
            try {Thread.sleep(5700);} catch(InterruptedException e) {};
            System.out.println ("  joins: \r\n");

            for(int i=0;i< nCars;i++) {
                  try {c[i].join();}     catch(InterruptedException e) {};
                  write(i, 8," Car" +i+" Ends.");
            } // end for => Main proc waits that all cars exit park2.
      } // end main
} // end class
```