Presentation av obligatoriska uppgift 5
trafiksimulering

Ett större program med flera klasser

Hur man designar ett system

Hur man gör simuleringar

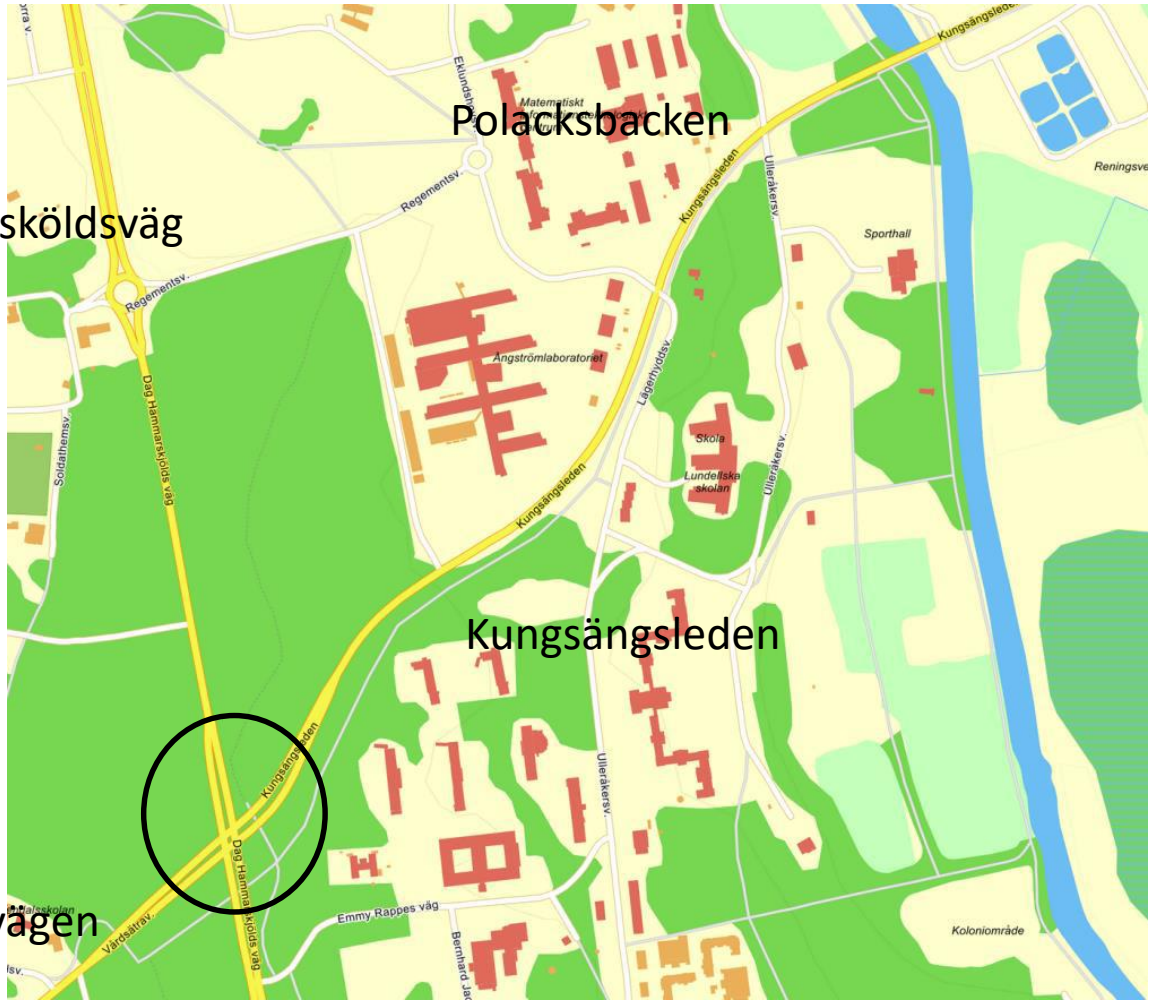
Korsningen Dag hammarsköldsväg och Vårdsätravägen/Kungsängsleden

Dag Hammarsköldsväg

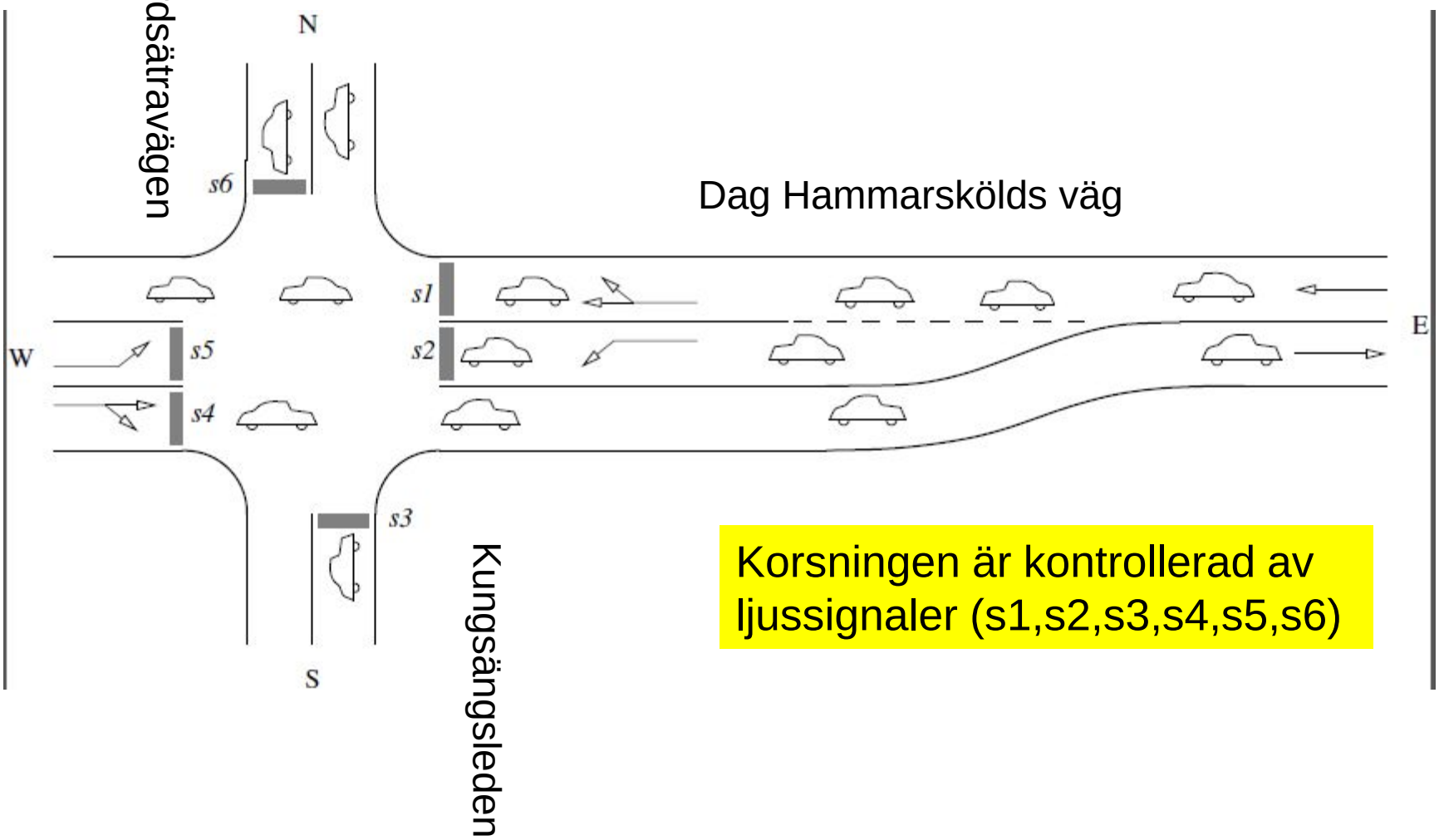
Polacksbacken

Kungsängsleden

Vårdsätravägen

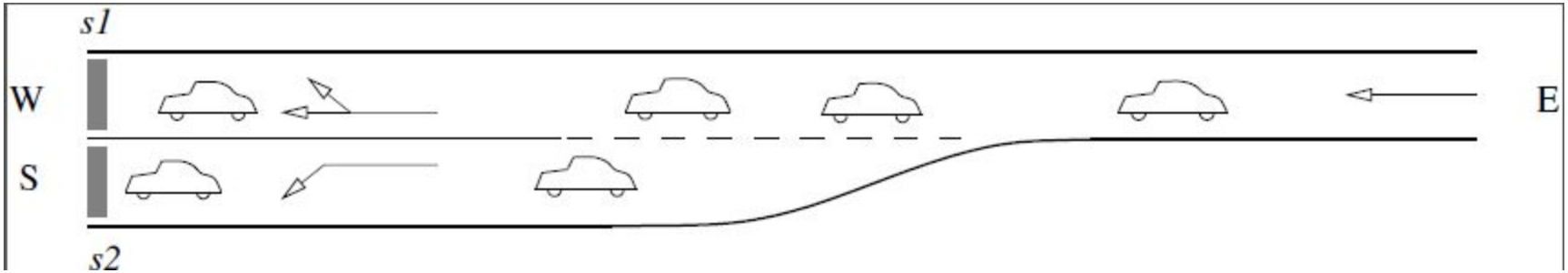


Korsningen Dag Hammarsköldsväg och Vårdsätravägen/Kungsängsleden



Korsningen är kontrollerad av ljussignaler (s1,s2,s3,s4,s5,s6)

Uppgiften består i att simulera en förenklad korsning!



Korsningen är kontrollerad av två ljussignaler (s1,s2)

Vi kommer att ta upp följande...

Hur gör man en simulering?

Vilka förenklingar av verkligheten gör man i simuleringen?

Vilka klasser behövs?

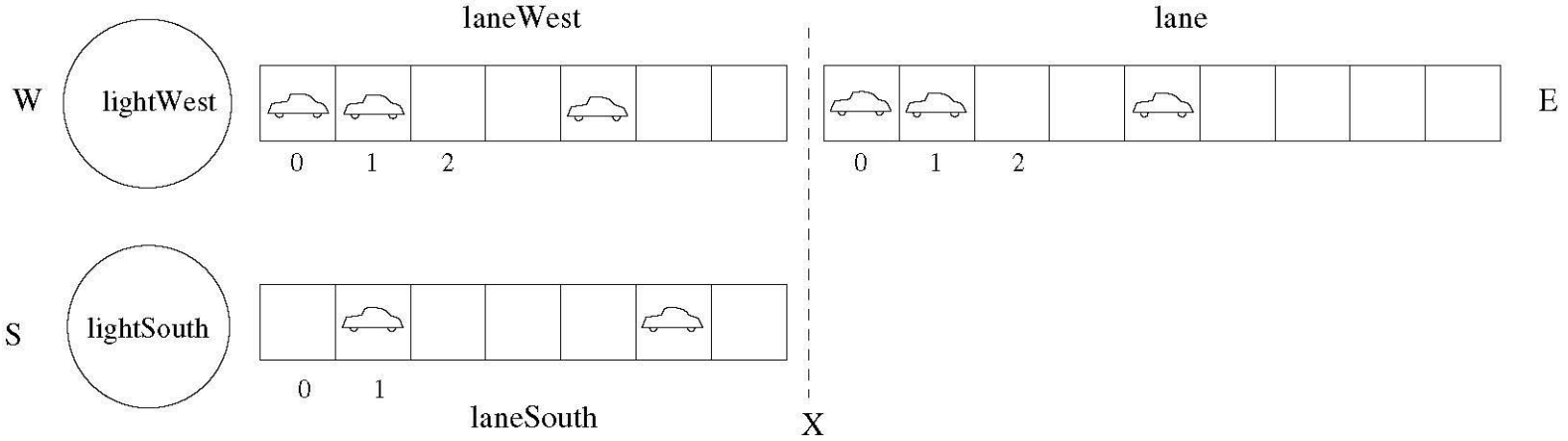
Hur skall (trafik)filerna kopplas ihop?
Hur kopplas signaler och (trafik)filer?

Vilka indata behövs? Vem hanterar indata?

Hur går tidsstegningen till? Vilka objekt bör känna till tiden?

Var samlas statistiken? Vad skall skrivas ut, när och av vem?

Implementation av den förenklade korsningen (*trafiksystemet*)



Det behövs tre st filer och två st signaler för denna implementation

- Fordon kommer in i korsningen i en fil `lane` vid E.
- Fordon med destination W kör in i fil `laneWest`
- Fordon med destination S kör in i fil `laneSouth`

Notera att filerna består av diskreta positioner, i varje sådan position finns ett fordon eller är tomt.

Klasser som behövs i trafiksystemet

Vilka olika typer av objekt kan utkristalleras?

Jo, följande delar behövs i trafiksystemet:

- Vehicle: Representerar ett fordon
- Light: Representerar en trafiksignal
- Lane: Representerar en fil

Dessutom behövs en klass som utgör själva trafiksystemet som består av ovan tre delar (klasser).

- TrafficSystem:
  - Tre Lane-listor med element av typen Vehicle
  - Två st Light-objekt

    + en kö av inkommande fordon vid E (se fig s 6)
    + variabler som samlar på sig data som man kan beräkna statistik från

Hur gör man simuleringen?

Jo, allt måste styras av en global ”klocka”. Denna tickar ett tidssteg i taget. Programmet har en variabel för detta.

Vid starten för simuleringen (t=0) har hela trafiksystemet initiala värden: Inga fordon finns i systemet, trafikljusen visar grönt.

I nästa steg, t=1 skall trafiksystemet ”stegas”, dvs trafikljusen skall också stegas i deras repetitiva beteende (och kanske slå om ljuset). Dessutom skall fordon i filerna röra sig framåt ett steg framåt (om möjligt) och det skall kontrolleras om ett nytt fordon kommer in (föds) vid punkten E. När fordonet föds måste vi se till att fordonet kommer ihåg tiden och dess destination.

Om det i ett givet tidssteg var grönt ljus och ett fordon lämnat systemet så skall det bokföras (dvs tidsvärdet), man kan därmed beräkna hur lång tid det tog för fordonet att passera korsningen.

Tidstegningen kan göras hur många ggr som helst

Det finns ett program TrafficSystem0 som sköter om (driver) simuleringen och tidsstegningen av trafiksystemet.
Finns att ladda ned från kursens hemsida: trafficSystem0.py
(www.it.uu.se/edu/course/homepage/prog1/python/ht20_p12/lessons_p12/le10/code/traficSystem0.py)
Programmet innehåller:

- Ett skal till en klass TraficSystem.
- En funktion (`main`) som sköter simuleringen

Vi kikar på programmet och kör det.

Vi ser att det i main-funktionen skapas ett **`TrafficSystem`**-objekt **`ts`**, med satsen:

```
ts = TrafficSystem()
```

Att **`TrafficSystem`** har följande metoder:

- **`step()`**
- **`snapshot()`**
- **`printStatistics()`**

ty mainmetoden anropar dessa metoder för **`ts`**.

I klassen TrafficSystem ser vi bl.a följande:

```
def __init__(self):
        self.time = 0  # Klockan är noll

def snapshot(self):
        print(f'Time step {self.time}')

def step(self):
        self.time += 1
```

- Klockan styrs av instansvariabeln `time`. Den är noll när TrafficSystem-objektet skapas.
- Klockan ökar med 1 av metoden `step`.
- Metoden `step` skriver ut vad klockan är.

Metoderna skall göra mer än det ovan visar…

Hur gör man själva simuleringen när det gäller fordon som kommer in i korsningen vid E?
Dvs hur genereras/produceras fordonen?

Fordon måste ju komma in i korsningen med en viss frekvens (intensitet).
I en simulering kan man låta detta styras av en modell som definierar hur fordon skapas, dvs kommer in i korsningen.

Genereringen av fordon sköts av klassen Destinations.
Den finns att ladda ned från kursens hemsida,
destinations.py
(www.it.uu.se/edu/course/homepage/prog1/python/ht20_p12/lessons_p12/le10/code/destinations.py)
Vi kikar på klassen och testkör.
Klassen sköter också om vilken destination fordonen har.

Klasserna `Vehicle`, `Light` och `Lane` är givna

De finns i filen trafficComponentsNew.py på kurshemsidan att ladda
(`www.it.uu.se/edu/course/homepage/prog1/python/ht20_p12/lessons_p12/le10/code/trafficComponentsNew.py`)

Vi kikar på denna fil.

Testkör filen.

Givna saker på kurswebben

Hela uppgiften är beskriven i lektion 10 på kurshemsidan.

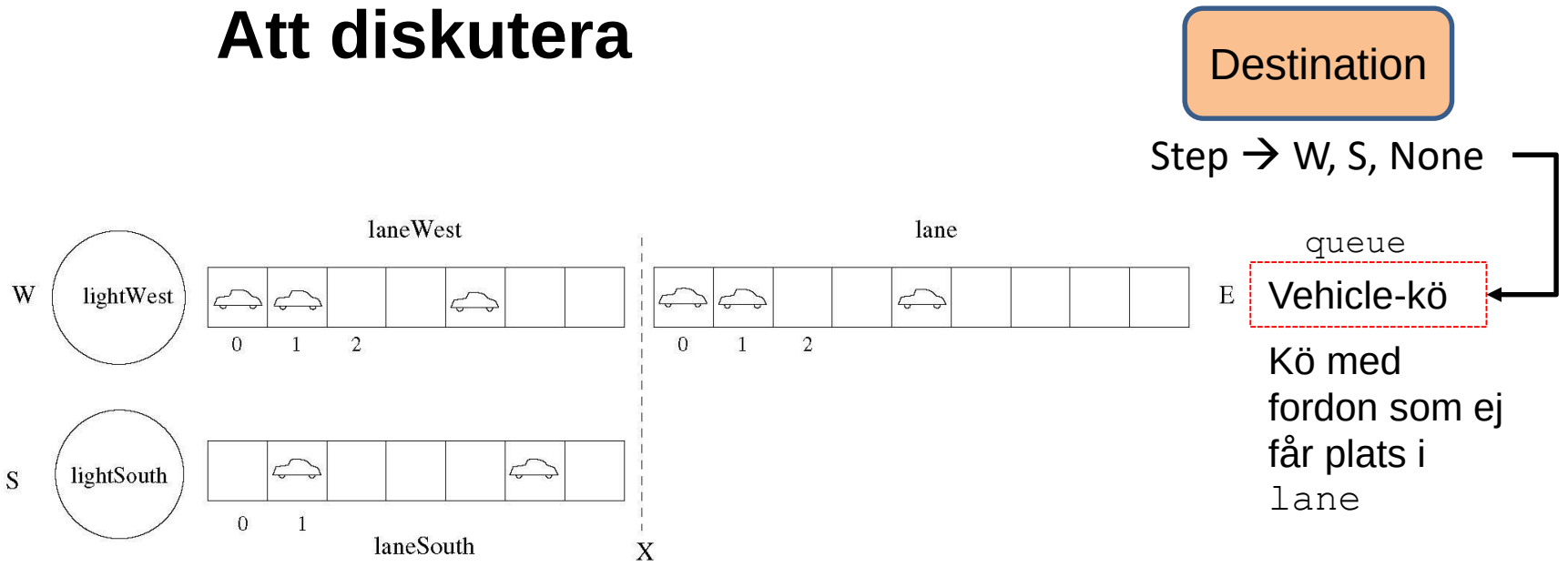
De färdiga klasserna

- `Destinations` i filen `destinations.py`
- `Vehicle`, `Light`, `Lane` i filen `trafficComponentsNew.py`

Kodskal för klassen:

- `TrafficSystem` i filen `trafficSystem0.py`

Att diskutera



Vet ett fordon om var den befinner sig? Nej.
Ett Lane-objekt har koll på sina fordon. TrafikSystem-objektet har koll på Lane-objekten.

Kan ett fordon titta framför sig och ”se” vad som är framför i filen? Nej. TrafikSystem-objektet har koll på detta

Kan ett fordon ”se” trafikljuset? Nej. TrafikSystem-objektet har koll

Arbetsgång när du börjar arbetet

Du testkör programmet `trafficComponentsNew`.
Läs igenom de färdiga klasserna Vehicle, Light och Lane som finns där samt den kod som testar dem.

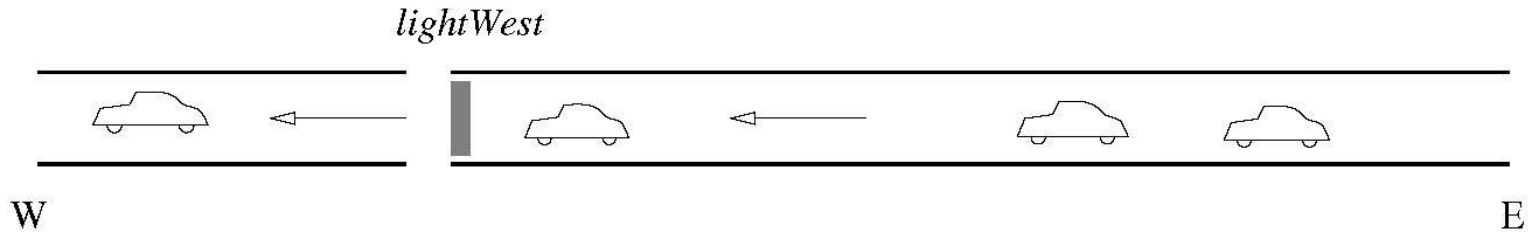
Testkör programmet `destinations`. Läs igenom klassen Destination som finns där.

Du har nu fyra pusselbitar (klasser) att använda för att skriva **ett första testsystem**, se nästa sida. I och med att de fyra klasserna är färdiga så behöver du inte ”bekymra” dig om dessa, utan ”bara” använda dem.

Tips: Börja med göra en kopia av den givna python-filen `trafficSystem0.py` till `trafficSystem1.py`, i vilken du skall skriva det första testsystemet.

Ett första testsystem, `trafficSystem1.py`

Börja med att få detta förenklade system att fungera först



I denna implementation räcker det med två Lane-objekt och ett Light-objekt

Klassen TrafficSystem som har koll på detta system består alltså av:

- Två Lane-objekt (dvs före och efter trafikljuset)
- Ett Light-objekt
- Ett Destinations-objekt (W,S ger fordon, None inget fordon)
- En lämplig variabel som tar hand (köar) bilar (vid E) som ej får plats i filen före trafikljuset.

Så här kan klassen TrafficSystem för det första testsystemet där det hela tiden är grönt ljus

```
class TrafficSystem:

    # Intitieringsmetod (konstruktor)
    def __init__(self):
        self.time = 0
        # Destination-objekt
        self.generator = destinations.Destinations()
        self.laneEast = Lane(5)  # Lane east, 5 platser
        self.laneWest = Lane(5)  # Lane west, 5 platser
        # Trafikljus, alltid grönt
        self.light = Light(5, 5) # period, green time
        # Här saknas en variabel som tar hand
        # köar) bilar som ej får plats i filen före
        # trafikljuset.
```

Exempel anrop:

```
ts = TrafficSystem()
```

Fortsättning →

.

Metoden `step` i klassen `TrafficSystem`

```
# Denna metod utför ett tidssteg i systemet
  def step(self):
```

- Öka time med 1
- Ta ut (bort) första fordonet från filen `laneWest`
- Stega filen `LaneWest` med dess `step`-metod, dvs flytta fordonen ett steg
- Om trafikljuset är grönt så flytta första fordonet på filen `laneEast` till sist i filen `laneWest`
- Stega trafikljuset med dess `step`-metod.
- Stega filen `laneEast`, dvs flytta fordonen ett steg.
- Anropa step-metoden i Destinations-objektet. Om denna returnerar en destination (d.v.s. något annat än `None`) så skapa ett fordon och lägg det sist på filen `laneEast`
- Istället för att lösa alla ovan delproblem på en gång, kan det vara bra att lösa ett delproblem i taget och testa detta

När det fungerar för alltid grönt trafikljus ändra det till omväxlande grönt och rött ljus

Måste då ta hand om följande scenario:
Om trafikljuset visar rött och det redan ligger ett fordon på sista platsen på filen `laneEast` får vi problem om det samtidigt kommer ett nytt fordon. För att undvika det ska du organisera en kö för fordon som väntar på att komma in i filen. För detta behövs en instansvariabel av listtyp.
Koden blir enklast om man alltid placerar tillkommande fordon i kön och sedan hämtar det första fordonet från kön.

- Ändra delar av konstruktorn i `TrafficSystem` enligt:
  `Ljusperioden 5, varav 3 gröna (och 2 röda)`
  Testa att systemet att det fungerar
- Utmana systemet genom att ändra till
  `Ljusperioden 5, varav 2 gröna (och 3 röda)`
  Testa systemet
- Utmana systemet ytterligare genom att ändra till
  `Ljusperioden 5, varav 1 grön (och 4 röda)`
  Testa systemet

Utskrifter för `trafficSystem1.py`

Metoden `snapshot` i klassen `TrafficSystem` skall skriva ut alla de värden som gör att vi kan se vad som händer i systemet. Metoden skall skriva ut trafikljusets värden och filens värden i varje tidssteg

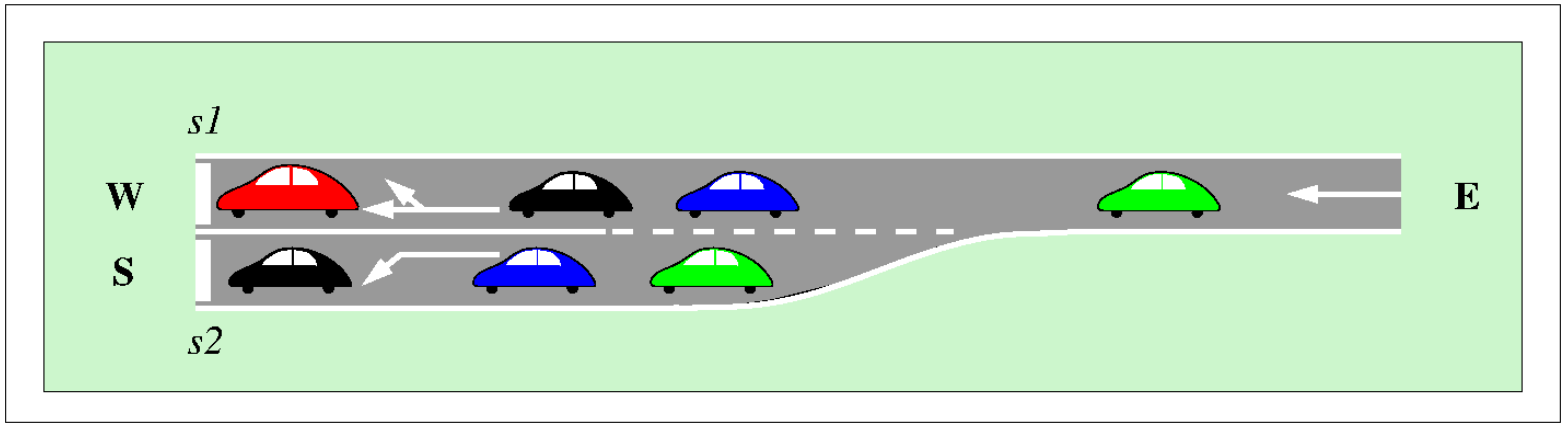
Exempel på en programkörning med utskrifterna:

```
  0: [.....](G)[.....]   []
  1: [.....](G)[....S]   []
  2: [.....](G)[...SS]   []
  3: [.....](G)[..SSS]   []
  4: [.....](G)[.SSSS]   []
  5: [.....](G)[SSSSS]   []
  6: [....S](G)[SSSSW]   []
  7: [...SS](G)[SSSWS]   []
  8: [..SSS](R)[SSWSS]   []
  9: [.SSS.](R)[SSWSS]   ['S']
 10: [SSS..](G)[SSWSS]   ['S', 'W']
 11: [SS..S](G)[SWSSS]   ['W', 'W']
 12: [S..SS](G)[WSSSW]   ['W', 'S']
 tid   West  L  East        Queue
```

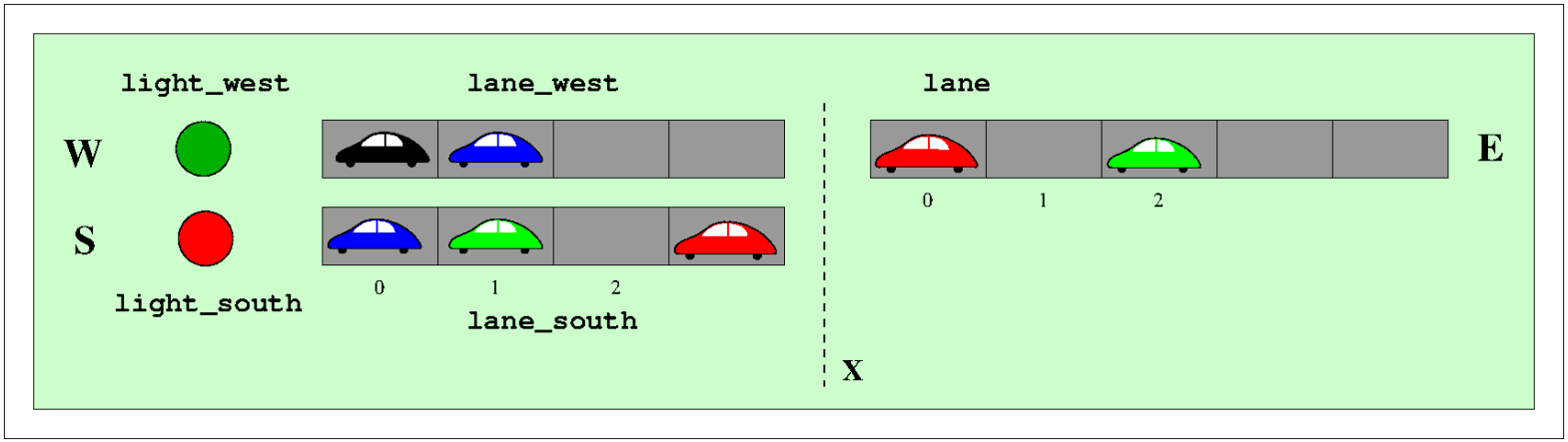
Trafikljuset har här perioden 10 och gröntid 8

Det fullständiga programmet, `trafficSystem2.py`

Tips: Börja med göra en kopia av din python-fil `trafficSystem1.py` (första systemet). Döp kopian till `trafficSystem2.py` eftersom detta program skall simulera det andra systemet. Programmet skall simulera följande scenario.



Trafiksystemet ska således bestå av tre filer (`lane`, `laneWest` och `laneSouth`), trafikljusen `lightWest` och `lightSouth` samt en kö (`Queue`) vid `E`.



Alla fordon skall först in i filen `lane`, men om den är full skall fordonen placeras i kön vid E (ej utritad ovan).

Från filen `lane`, skall fordonen placeras i filen `laneSouth` eller `laneWest` beroende på fordonets destinationsvärde S eller W.

Om exvis trafikljuset `lightSouth` visar rött och filen `laneSouth` är full innebär detta att filen `lane` blockeras.

Tips: Börja med en förenkling där båda ljusen är gröna hela tiden, så att du ser att det blir ett flöde av fordon genom hela systemet i filerna och utan några som helst hinder.

Statistik som skall beräknas i det andra testsystemet
Data för statistiken fås genom att varje gång ett fordon lämnar korsningen beräknas hur länge fordonet varit i korsning. Bl.a. genomsnittlig tid för fordon och maximal tid för ett fordon att passera korsningen.

Samla tiderna för fordon som lämnar systemet i två lämpliga variabler ett för vardera utgång, för att ta fram statistiken Det innebär att trafiksystemet skall innehålla en instansvariabel som samlar på sig lämpliga data.

Statistik skall beräknas och skrivas ut av metoden `print_statistics` i klassen TrafficSystem.

- genomsnittliga och maximala tider (antal tidssteg) för fordon att passera ljussignalen `lightW` respektive `lightS`
- andel tidssteg som kön framför vardera signalen varit längre än längden på `laneW` och `laneS` dvs den tid som fildelningen vid X varit blockerad av kö

  (för att beräkna detta behöver man räkna antal tidssteg som det ej gick att lägga ett fordon i `laneW` resp `laneS` därför att sista positionen var upptagen)
- andel tidssteg som det funnits fordon i kön vid entrypunkten (E).

Notera – uppgiften finns beskriven till fullo på

www.it.uu.se/edu/course/homepage/prog1/python/ht20_p12/lessons_p12/le10/

Redovisas senast 14 dec,vid L10:7