Zusatzblatt 2: Simulation am Computer – *Zweispurige* Verkehrssimulation

Jetzt sind eure Programmierkünste wieder gefragt, damit ihr eure Vermutungen aus vorigen Aufgaben wieder überprüfen könnt, ohne viele verschiedene Situationen per Hand auszuwerten –ein Hoch auf die Technik! Auch für diese Simulation existiert wieder eine Datei, in der ihr eure Regeln realisieren könnt.

Bevor ihr loslegen könnt, gibt es noch eine kleine Erweiterung an den Autos, denn was wären Autos ohne Blinker, wenn sie überholen wollen. Also gibt es noch ein drittes Attribut, der *Blinker* vom Typ Boolean. Auch hier gibt es wieder zwei Methoden, um auf das Attribut zugreifen zu können:

- **public boolean** getBlinker(): Liefert den aktuellen Zustand des Blinkers zurück. Ist der Blinker an, so liefert die Methode **true** zurück, sonst **false**.
- **public void** setBlinker(**boolean** blinker): Schaltet den Blinker eines Autos entweder an oder aus.

Die Klasse **ZweispurigeSimulation** hält wieder zwei Attribute bereit, die ihr schon aus der einspurigen Simulation kennt:

- **private** Auto[][] aktuelleGeneration = **new** Auto[20][2]: Ein zweidimensionales Array der Größe 20 vom Typ Auto. Es repräsentiert die zweispurige Straße und speichert die Autos an den entsprechenden Positionen ab.
- **private int** troedelfaktor = 0: Speichert die Wahrscheinlichkeit von 0 bis 100, mit der Autos später trödeln. In der Erweiterung der Simulation könnt ihr diesen Trödelfaktor später einbauen.

Realisiert euren Algorithmus in der Methode **public void** berechneNaechsteGeneration(). Welche Stellen haben sich im Vergleich zur einspurigen Simulation nicht verändert? Diese Teile könnt ihr aus der einspurigen Simulation übernehmen. Mit einem Doppelklick der Datei **ZweispurigeSimulation.java** könnt ihr auf der Überholspur durchstarten ☺.

Hinweis: Im Folgenden sind nur noch Hilfestellungen für den Spurwechsel gegeben. Den Rest schafft ihr schon selbst! Fühlt ihr euch fit genug in der Umsetzung, könnt ihr die Methode auch komplett selbst realisieren.

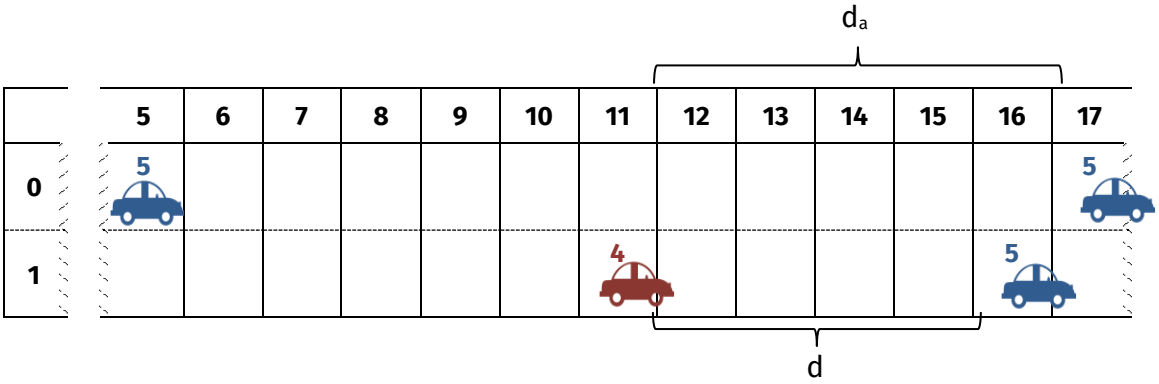
Der Spurwechsel

Um den Algorithmus für die Berechnung der nächsten Generation und damit auch den Spurwechsel etwas zu vereinfachen, könnt ihr euch Hilfsmethoden schreiben, die ihr dann zur Berechnung einfach verwenden könnt. Damit bleibt der Code lesbar und ihr unterteilt den ganzen Algorithmus in übersichtliche und leicht zu lösende Probleme.
Für den Spurwechsel eigenen sich zwei Hilfsmethoden, deren Methodenrümpfe bereits vorgeben sind:

- eine Methode zur Berechnung des Abstands zum Vordermann für den Wechselanreiz
- eine Methode zur Berechnung des Sicherheitsabstandes zum Hintermann.

Hilfsmethode 1 – Den Abstand zum Vordermann berechnen

Die Methode **private int** abstandZumVordermann(**int** position, **int** spur) bekommt als Parameter die aktuelle Position des Autos und die entsprechende Spur ,auf welcher der Abstand zum Vordermann gemessen werden soll, übergeben. Anschließend soll die Anzahl an freien Feldern zurückgegeben werden.

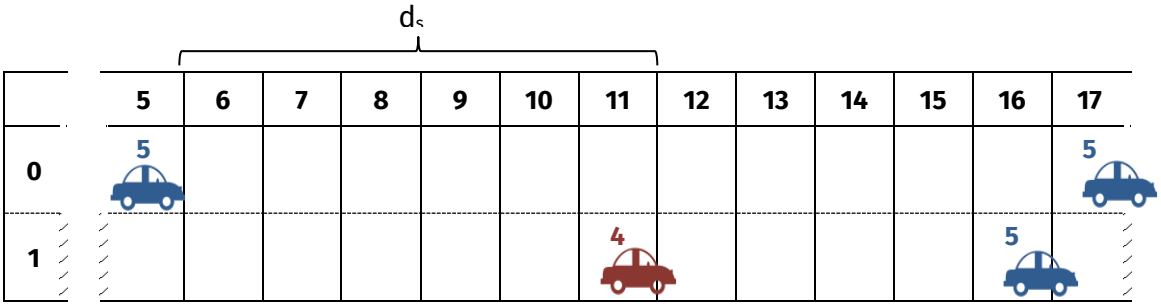


Der Aufruf abstandZumVordermann(11, 1) würde hier den Wert 4 zurückliefern.

Durch den Parameter spur könnt ihr mit nur einer Methode die Abstandberechnungen auf beiden Spuren berechnen. Mit der genauen Position können mit Hilfe einer Schleife solange die freien Felder vor dem Auto gezählt werden, bis ein anderes Auto gefunden wird.

Hilfsmethode 2 – Den Sicherheitsabstand berechnen

Die Methode **private boolean** sicherheitsabstand(**int** position, **int** spur) bekommt als Parameter die Position des Autos auf der Spur und die entsprechende Spur, auf welcher der Sicherheitsabstand zum Hintermann gemessen werden soll, übergeben. Wird der Sicherheitsabstand nach hinten eingehalten, so liefert die Methode **true** zurück, ansonsten **false**.



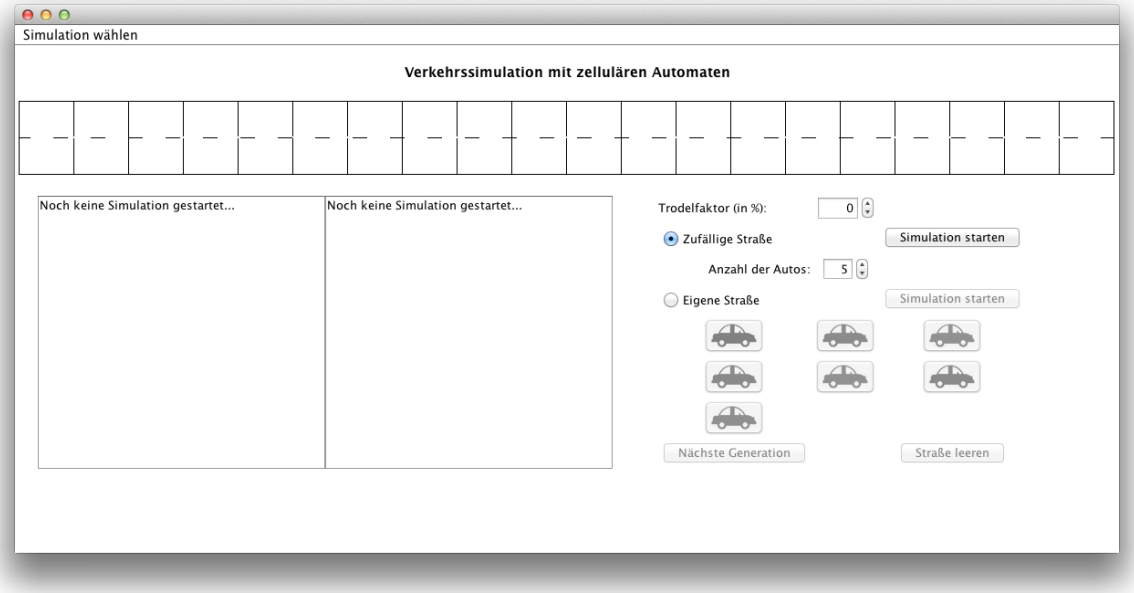
Der Aufruf sicherheitsabstand(11, 0) würde hier den Wahrheitswert **true** zurückliefern.
Auch hier kann wieder mit Hilfe einer Schleife die Anzahl der freien Plätze bis zum nächsten Auto festgestellt werden – allerdings diesmal rückwärts!

Umsetzung in der Methode berechneNaechsteGeneration()

Hier könnt ihr nun eure selbstgeschriebenen Hilfsmethoden aufrufen. Unterteilt den Code für den Spurwechsel ruhig in zwei Teile: eine Schleife zum *Bremsen und/oder Überholen auf der rechten Spur* und anschließend eine Schleife zum *Bremsen und/oder Wiedereinordnen auf der linken Spur*. Denkt daran, hier die Blinker beim Spurwechsel anzuschalten und später beim Bewegen der Autos auch wieder auszuschalten. Sonst gibt es nachher noch Unfälle!

Die Simulation testen

Startet nun die Simulation. Klick oben links in der Leiste auf *Simulation wählen* und dann *Zweispurige Straße*. Jetzt ist auch die Überholspur freigegeben.



Aufgabe 1:

Überlegt euch geeignete Testfälle, mit denen ihr eure Vermutungen aus den Aufgaben 2 und 3 von vorherigem Blatt überprüfen könnt. Stimmen eure Vermutungen?

Aufgabe 2:

Wie realistisch ist eure Simulation? Sind die Überholmanöver ähnlich zum realen Straßenverkehr oder gibt es Unterschiede? Mögliche Änderungen könnten sein:

- **Optimierung des Sicherheitsabstandes**: Autofahrer können durch den Rückspiegel in der Regel einschätzen, wie schnell sich das Auto hinter ihnen auf der anderen Spur bewegt, um daran auszumachen, ob sie überholen können oder nicht.
- **Umsetzung des Rechtsfahrgebots**: Auf deutschen Autobahnen ist es nicht gestattet, rechts zu überholen. Sobald das Auto überholt wurde, sollte sich der Überholer wieder rechts einordnen.
- **Überholverbot für langsame Autos**: Auf vielen Autobahnabschnitten gibt es Überholverbote für LKWs, Autos mit Anhängern etc., da langsame Fahrzeuge sonst die Autobahn verstopfen. Nur wer schnell genug fährt, darf also überholen.
- **Eigene sinnvolle Erweiterungen.**