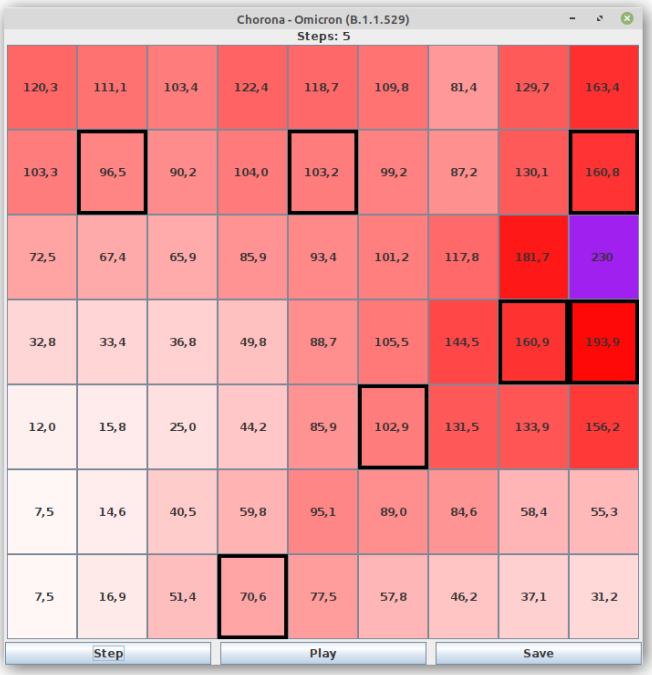


Kurs: TINF20B____ / Matr.-Nr.: _____

*Hinweis: Für die Bonuspunkt-Aufgabe am Ende der letzten Seite füllen Sie Kurs und Matrikelnummer aus und geben das Aufgabenblatt anders als bei der letzten Prüfung **ebenfalls ab**!*

*Andere Notizen/Markierungen auf dem Aufgabenblatt werden **nicht** bewertet!*

Hinweis zur Bewertung:

- 1/4 der Punkte (25%) wird nach Funktionstests Ihrer Lösung vergeben.
- 3/4 der Punkte (75%) werden entsprechend des in den Teilaufgaben angegebenen Schlüssels auf Basis des Quellcodes vergeben.

DHBW Karlsruhe, Vorlesung Programmieren I+II, Programmentwurf

14.01.2022

Bearbeitungszeit: 120 Minuten

Aufgabe

Die letzten beiden Jahre ist immer wieder von Aerosol-Verbreitung in Innenräumen berichtet worden. Um dies besser zu veranschaulichen, soll für verschiedene Varianten von SARS-CoV-2 eine Simulation umgesetzt werden. Ihre Aufgabe ist es, dies in Form der grafischen Java-Anwendung ***Chor*ona** zu entwickeln, welche ein (stark vereinfachtes!) Modell zur Aerosol-Verbreitung bei einer Chorprobe veranschaulicht.

Der Raum wird hierfür in einzelne Zellen unterteilt und es werden zufällig infizierte Chormitglieder platziert. Anschließend wird in Zeitscheiben die Verbreitung der Viruspartikel simuliert und grafisch dargestellt. Der Algorithmus für die Übertragung in die jeweiligen Nachbarzellen wird Ihnen bereitgestellt.

Teilaufgabe a) **[8%]**

Zur Unterscheidung der verschiedenen Varianten von SARS-CoV-2 in unserem Programm soll zunächst ein *komplexer Aufzählungstyp* **`Variant`** realisiert werden.

Der Konstanten-Name ist der griechische „Buchstabe“ der für die Variante vergeben wurde. Ein Label für die Anzeige (`label`) sowie die korrekte wissenschaftliche Bezeichnung (`designation`) sollen als Attribute erfasst werden:

Variant	Label	Wiss. Bez.	Variant	Label	Wiss. Bez.
`WILD_TYPE`	`Wildtyp`	`SARS-CoV-2`	**`DELTA`**	`Delta`	`B.1.617`
`ALPHA`	`Alpha`	`B.1.1.7`	**`FETA`**	`Feta`	`O.u.z.o`
`BETA`	`Beta`	`B.1.351`	**`LAMBDA`**	`Lambda`	`C.37`
`GAMMA`	`Gamma`	`P.1`	**`OMICRON`**	`Omikron`	`B.1.1.529`

Teilaufgabe b) **[6%]**

Zur Nutzung der vorgegeben Klasse für die Viruslast-Verteilung muss die dort verwendete Java-*Schnittstelle (Interface)* **`IRoom`** realisiert werden. Folgende Methoden müssen definiert werden:

- `double getDose(int x, int y)`: Liefert Viruslast an den übergebenen Koordinaten des Rasters
- `void setDose(int x, int y, double dose)`: S*etzt* die Viruslast an den übergebenen Koordinaten
- `void addDose(int x, int y, double dose)`: *Erhöht* die Viruslast an den übergebenen Koordinaten
- `void step()`: Simuliert eine „Zeitscheibe“ mit Ausatmen und Verteilung der Aerosole

Hinweis: Sie müssen die Schnittstelle nur definieren, die Implementierung ist mit der Klasse `Room` bereits vorgegeben.

Teilaufgabe c) **[13%]**

Zur Definition einer Koordinate für infizierte Personen erweitern Sie die bereitgestellte Klasse **`Point`**, welche als Attribute zwei ganzzahlige Werte für die `x`- und `y`-Komponente einer Koordinate speichern kann. Definieren Sie einen Konstruktor der für beide Attribute Werte entgegennimmt und diese speichert.

Hinweis: Ein Klassenrumpf mit vordefinierter `hashCode` und `equals`-Methode ist bereits vorgegeben. Ergänzen Sie sinnvoll!

Um Parameter für den Verteilungsalgorithmus definieren zu können, erstellen Sie die Klasse **`RoomSetting`**, welche alle relevanten Informationen über einen Raum abspeichern kann. Das sind sowohl die Breite (`width`) und Höhe (`height`) des Rasters für den Raum (beide ganzzahlig) als auch eine Sammlung von Koordinaten mit den Positionen der infizierten Chormitglieder (`pollutants`).

Der Konstruktor soll die Breite und Höhe als Parameter entgegennehmen. Darüber hinaus ist die *Anzahl* der infizierten Personen zu übergeben (vgl. Verwendung in vorgegebener Klasse `Chorona`).

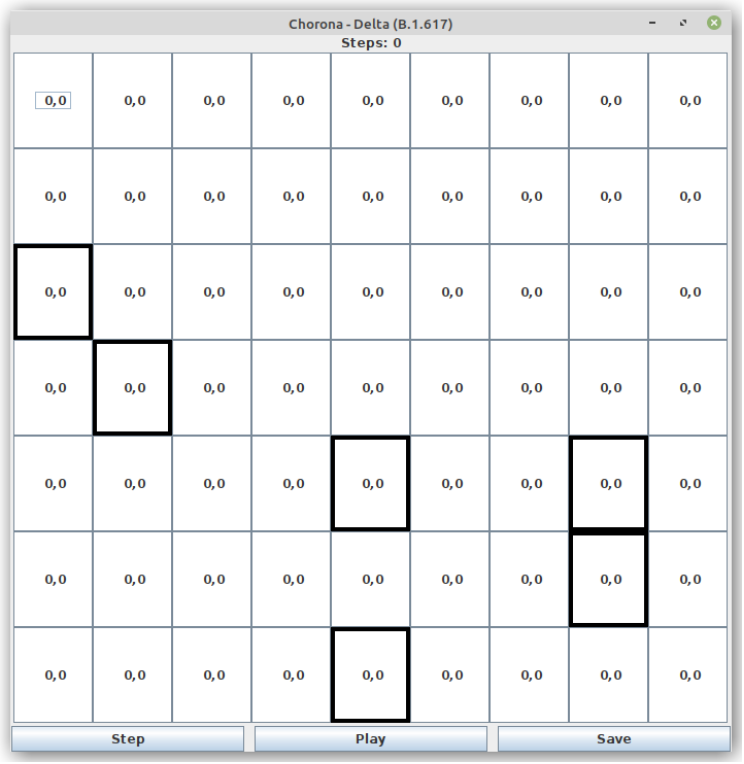
Sollte die Anzahl der sich aus Breite und Höhe ergebenen Felder kleiner sein als die Anzahl der übergebenen infizierten Personen ist eine *selbst zu schreibende* **`AHAException`** mit der *übergebenen* Nachricht „Insufficient singers“ zu werfen.

Ansonsten ist die übergebene Anzahl an Infizierten auf zufällige Koordinaten des Rasters zu verteilen. Fügen Sie dazu für jeden Infizierten einen Punkt mit den entsprechenden zufälligen Koordinaten in `pollutants` ein. Duplikate müssen hierbei vermieden werden!

Teilaufgabe d) **[18%]**

Zur Visualisierung der Verbreitung setzen Sie nun die grafische Oberfläche in Form der Klasse **`ChoronaTerminal`** um. Der Konstruktor nimmt die Virus-Variante für das Fenster sowie die passende Room-Instanz entgegen (vgl. Verwendung in bereitgestellter Klasse `Chorona`) und speichert diese in den Attributen `variant` und `room` mit den entsprechenden Datentypen.

Der Titel des Fensters ist nach dem Muster „`Chorona - $variantLabel ($variantDesignation)`“ zu setzen (vgl. Screenshot). Im oberen Bereich ist ein Label zur Anzeige der aktuellen Zeitschreibe („Step“) vorzusehen, darunter ein Bereich zur Anzeige des Rasters. Am unten Rand sind 3 Buttons *Step*, *Play* und *Save* vorzusehen.



Die übergebene Room-Instanz stellt Ihnen einige benötigte Informationen zur Verfügung:

- Breite und Höhe des Rasters sowie die Positionen der infizierten Chormitglieder über die **`RoomSetting`**-Instanz welche die `getSettings` -Methode liefert.
- Die aktuelle Zeitscheibe (`getStep`-Methode)
- Die aktuelle Belastung einer Zelle abhängig von x und y-Werten mit der `getDose(x,y)`-Methode.

Zur Repräsentation einer Zelle im Raster sollen Sie eine weitere Klasse **`CellButton`** realisieren, welche von `JButton` erbt und zusätzlich die aktuelle Belastung (**`double`** `dose`) und einen Wahrheitswert zur Bestimmung ob die Zelle eine infizierte Person enthält oder nicht (`polluter`).

Sollte die Zelle eine infizierte Person beinhalten ist ein deutlich sichtbarer der Rahmen (vgl. Hinweis unten) zu setzen!

Wenn die Belastung des Buttons gesetzt wird, soll die vorgegebene Methode `Chorona.updateButtonForDose` aufgerufen werden, die sowohl den Text als auch die Hintergrundfarbe bzw. Icon des Buttons aktualisiert (vgl. Screenshot Deckblatt). Initial ist eine Belastung von 0 zu setzen (o.g. Methode aufrufen!).

Beim Drücken des *Step*-Buttons soll nun die Verbreitung der Viruslast simuliert werden.

Hierfür ist:

- in der Room-Instanz durch Aufruf der `step`-Methode die Verbreitung der Viruslast für eine Zeitscheibe auszulösen (Simuliert ausatmen + Verteilen)
- danach die Nutzeroberfläche zu aktualisieren, d.h.
  - die Anzeige der aktuellen Zeitscheibe („Steps“) aktualisieren
  - sämtlichen `CellButton`-Instanzen die neue Belastung zu setzen (neu aus `Room` auslesen!).

Hinweis 1: Um den Rahmen zur Markierung von Zellen mit infizierten Personen zu setzen, können Sie folgenden Aufruf in `CellButton` verwenden: `this.setBorder(BorderFactory.createLineBorder(Color.BLACK,4));`

Hinweis 2: Die Klasse `ChoronaTerminal` wird in den Teilaufgaben g) und h) noch erweitert.

Teilaufgabe e) **[4%]**

Bisher sind in der bereitgestellten Klasse `Chorona` feste Zahlen für Breite, Höhe und Anzahl der infizierten Personen für die `RoomSetting`-Instanz vorgegeben. Überarbeiten Sie die Initialisierung nun so, dass *zufällige* Werte verwendet werden. Dabei soll

- die Raumbreite zwischen 7 und 9 (jeweils inklusive!) sein,
- die Raumhöhe ebenfalls zwischen 7 und 9 (jeweils inklusive!) sein
- und die Anzahl der infizierten Personen zwischen 5 und 7 (jeweils inklusive!) betragen.

Teilaufgabe f) **[6%]**

Um die unterschiedliche Ansteckungswahrscheinlichkeit für die verschiedenen Varianten bei neuen wissenschaftlichen Erkenntnissen flexibel anpassen zu können, sollen diese aus einer Datei eingelesen werden.

Ändern Sie die Implementierung der `getPollutantFactors`-Methode in der bereitgestellten Klasse `Chorona` (s. USB-Stick) nun so, dass die ebenfalls bereitgestellte Datei „`pollutantfactors.txt`“ (s. USB-Stick) zeilenweise eingelesen wird. Fehler müssen zwar abgefangen, aber nicht weiter behandelt werden.

Dabei ist jede Zeile der Datei analog zu den vorgegebenen Beispiel-Daten mit Hilfe der bereitgestellten Methode `parsePolluteFactor` in die vorgegebene `Map` einzufügen.

*Hinweis: Damit die Einlese-Methode funktioniert, müssen die Konstanten des Aufzählungstyps `Variant` (vgl. Teilaufgabe a)) **exakt** wie vorgegeben lauten. Dies betrifft auch Groß- und Kleinschreibung!*

Teilaufgabe g) **[13%]**

Erweitern Sie die Klasse `ChoronaTerminal` (vgl. *Teilaufgabe d)*) nun so, dass beim Drücken des *Save*-Buttons der aktuelle Zustand des Fensters in eine Datei geschrieben wird. Der Dateiname soll aus Virusvariante, Raum-Breite/Höhe, Anzahl infizierter Personen und aktuellem Step zusammengesetzt werden (vgl. Screenshots).

Falls die Datei bereits existiert, soll sie ***überschrieben*** werden. In jedem Fall soll ein Dialog am Ende angezeigt werden, der darüber informiert ob die Datei erfolgreich gespeichert wurde oder nicht:

Hinweis: Beispiel für Breite: 9, Höhe: 7, Infizierte: 6, Step: 2. Der Dateiname muss im Dialogtext erwähnt werden. Dialog-Titel & -Icon sind egal, sofern aus dem Text eindeutig Erfolgs- und Fehlerfall ersichtlich sind.

Für jede Zelle soll nun eine Zeile mit ihren Koordinaten und dem aktuellen Wert für die Belastung geschrieben werden. Für ein 9x7-Raster hätte die Datei somit 63 Zeilen. Eine Zeile könnte bspw. so aussehen:

`3;6;254.3` (für x=3, y=6 und dose=254.3)

Teilaufgabe h) **[7%]**

Erweitern Sie die Klasse `ChoronaTerminal` (vgl. *Teilaufgabe d)*) nun so, dass beim Drücken des *Play*-Buttons automatisch 20 Schritte im Abstand von 0,5s ausgelöst werden. Diese Aktion soll *nebenläufig* durchgeführt werden.

Die nebenläufige Ausführung endet, nachdem die 20 Schritte durchgeführt wurden. Zu Beginn sind der *Step*- und *Play*-Button zu deaktivieren. Am Ende der Nebenläufigkeit werden beide wieder aktiviert.

Allgemeine Hinweise

Starten
Starten Sie die Anwendung mit der gegebenen Klasse `Chorona` (siehe USB-Stick).

Schließen eines Fensters
Beim Schließen eines Fensters soll die komplette Anwendung beendet werden.

Sichtbarkeit von Instanz-Attributen
Sämtliche Instanz-Attribute sind als privat zu definieren und von außerhalb der Klasse ggf. mittels Getter- und/oder Setter-Methoden zu verwenden.

Bonuspunkt: Zeichnen Sie (grob) das Icon, welches bei einem Zellenwert größer 500 erscheint: