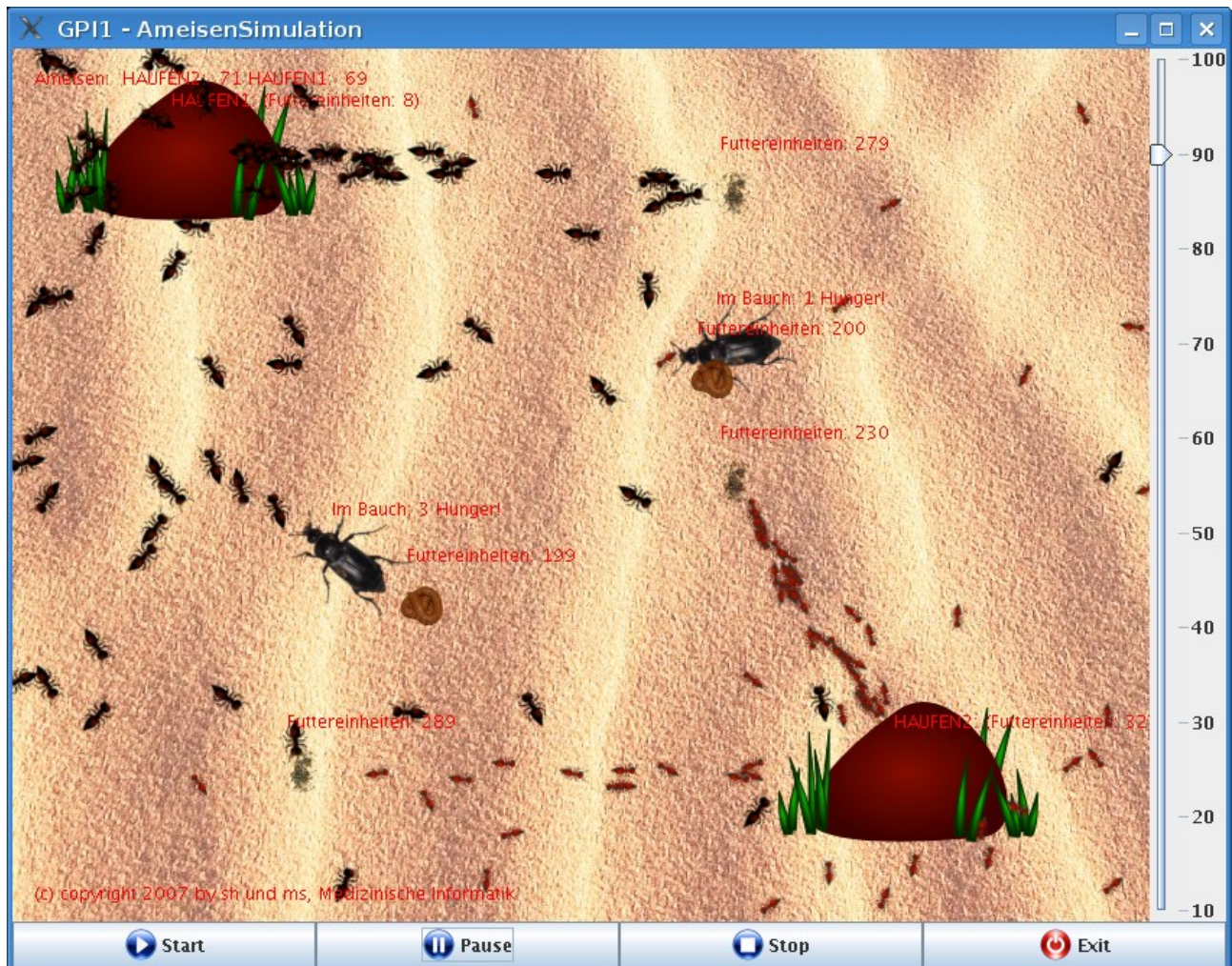


## Aufgabenbeschreibung GPI1

Ziel der Programmierübungen im ersten Semester ist die Implementierung einer Simulation virtueller Ameisen. In insgesamt drei Ausbaustufen soll das Programm schrittweise entwickelt werden. In der dritten Ausbaustufe kann das Programm in etwa so aussehen:



Die einzelnen Ausbaustufen im Detail:

1. Ameisen laufen zufällig auf dem Spielfeld herum. Das Spielfeld beinhaltet ausschließlich Ameisen. Die Ameisen laufen nicht über den Spielfeldrand hinaus. Stößt eine Ameise an den rechten Rand, stirbt sie. Dafür wird in Punkt (100, 100) für jede gestorbene Ameise eine neue Ameise geboren.
2. Dem Spielfeld wird ein Ameisenhaufen hinzugefügt, zu dem die Ameisen gehören. Weiterhin sind auf dem Spielfeld Futterhaufen, die jeweils eine gewisse Anzahl an Futtereinheiten beinhalten. Die Ameisen laufen so lange zufällig auf dem Spielfeld herum, bis sie einen Futterhaufen finden. Haben sie einen gefunden, nehmen sie eine Futtereinheit vom Futterhaufen, und tragen diese direkt zum Ameisenhaufen. Begegnet eine Ameise mit Futter unterwegs einer anderen Ameise, sagt sie dieser die Position des Futters weiter. Diese

geht dann ebenfalls zwischen Futter und Ameisenhaufen hin und her, bis alle Futtereinheiten verbraucht sind. Wichtig: Nur auf dem Rückweg vom Futter zum Ameisenhaufen darf die Position des Futterhaufens weiter gesagt werden. Geht die Ameise zu einem Futterhaufen und dieser existiert nicht mehr, macht sie sich wieder zufällig auf die Suche nach neuem Futter. Der Ameisenhaufen (dieser wird bereits vorgegeben), produziert für je 10 Futtereinheiten neue Ameisen, die in zufälliger Zahl jeweils im 10-Sekunden-Takt geboren werden. (Zusatzaufgabe: Fügen Sie einen zweiten Ameisenhaufen hinzu, der ebenfalls ein Ameisenvolk beinhaltet. Die Futterposition darf natürlich nur an Mitglieder des eigenen Volkes weiter gesagt werden!)

3. Ein weiteres interessantes Element tritt auf: Ein Fressfeind! In unserem Beispiel wurde ein Käfer implementiert, der die jeweils nächste Ameise verfolgt, sie verspeist und erst bei vollem Bauch aufhört zu fressen. Sie können jedoch auch einen Vogel, Ameisenbär oder ähnliches implementieren.

Insgesamt sind Sie in der Implementierung sehr frei. Die Ameisensimulation regt zum Experimentieren an und wir wollen Sie nicht daran hindern, weitere Ideen zu verwirklichen, auch wenn sie nicht in der Aufgabenbeschreibung stehen. Die Aufgabenstellungen legen lediglich die Mindestanforderungen fest. Wollen sie statt dieser Mindestanforderungen etwas anderes implementieren, fragen Sie uns vorher. Zusätzliche Möglichkeiten können Sie natürlich jederzeit implementieren. Beispielsweise könnten Sie die Markierung von Wegen und der Futterposition realistischer durch virtuelle Pheromone simulieren.

Die Ausbaustufen sind zu den im Terminplan vorgegebenen Zeiten zu erreichen. Zur Qualität der Software, auf die bei der Korrektur geachtet wird, gehört insbesondere eine gute, leicht verständliche und erweiterbare Architektur. Aufgrund dessen, dass Sie zu Beginn der Programmierübung noch nicht alle notwendigen Fähigkeiten besitzen, um eine wirklich gute Architektur zu verwirklichen, werden manchmal zunächst „Notlösungen“ implementiert. Diese müssen dann in folgenden Aufgaben umgebaut werden (dies ist jedoch mit begrenztem Aufwand möglich), um eine gute Architektur zu erhalten. Dieser kleine Zusatzaufwand lässt sich leider nicht vermeiden.

Ergänzen Sie bei jeder Klasse, die Sie modifizieren oder neu hinzufügen als Autor Ihren Namen.

## **Aufgabe 1**

Verwenden Sie das vorgegebene Framework, um Ameisen laufen, sterben und geboren werden zu lassen. Sie haben ein BlueJ-Projekt<sup>1</sup> vorgegeben, in dem sich unter anderen folgende für die erste Aufgabe relevante Klassen befinden:

- Anzeigefeld
  - Diese Klasse repräsentiert ein Fenster, um Ameisen und andere Objekte anzuzeigen. Hinweis: Diese Klasse dürfen sie NICHT modifizieren.
- Grafikobjekt
  - Ein Objekt dieser Klasse stellt ein einfaches Grafikobjekt dar, welches auf dem Anzeigefeld dargestellt werden kann.

---

<sup>1</sup> zu finden im ILIAS (<https://ilias.mi.hs-heilbronn.de>) bei den Vorlesungsunterlagen zu MIB\_GPI1

Hinweis: Jede Klasse, die auf dem Anzeigefeld angezeigt werden soll, muss auf jeden Fall alle Methoden und privaten Felder enthalten, die diese Klasse ebenfalls enthält. Sie dürfen diese Klasse jedoch NICHT modifizieren!

- Ameise
  - Ein Objekt dieser Klasse stellt eine einzelne Ameise dar. Sie kann noch nicht sehr viel, außer auf dem Anzeigefeld angezeigt zu werden. Diese Klasse dürfen und sollen Sie modifizieren. Wie Sie sehen, hat Ameise alle Methoden und privaten Felder, die die Klasse Grafikobjekt besitzt, ergänzen Sie jedoch noch selbst weitere Felder und Methoden.
- Ameisensimulation
  - Diese Klasse führt die eigentliche Simulation durch. Sie können die Simulation starten, indem sie eine Instanz dieser Klasse erzeugen und dann die Methode simuliere() der erzeugten Instanz aufrufen. Wenn Sie dies tun, läuft eine Ameise von der Mitte des Anzeigefelds nach links.
- Hilfsklasse
  - Diese Klasse enthält einige hilfreiche Methoden, die Sie für die Implementierung verwenden dürfen.

Die weiteren Klassen des Projekts werden erst für die folgenden Aufgaben benötigt und können vorerst ignoriert werden.

Um Ihnen bei der Implementierung zu helfen, besitzt die Hilfsklasse bereits einige nützliche Methoden zur Winkelberechnung. Beachten Sie bitte, dass diese Methoden den Winkel im Bogenmaß verwenden.

- double sinus(double winkel)
- double cosinus(double winkel)
- double arcustangens(double x)
- double gradInRadiant(double grad)
- double radiantInGrad(double radiant)

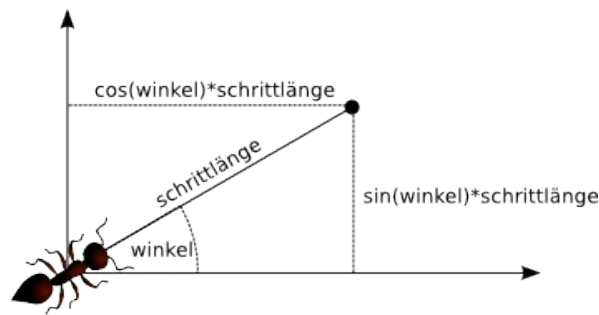
Weiterhin stehen Ihnen Methoden zur Generierung von Zufallszahlen zur Verfügung:

- int getZufallsGanzzahl(int grenze)
- double getZufallsDouble(double grenze)
- double getZufallsDoubleMitMinus(double grenze)

Die genaue Funktion entnehmen Sie bitte dem Kommentar oberhalb der Methode.

#### Aufgabenstellung:

1. Modifizieren Sie das Programm, so dass die Ameise in andere Richtungen läuft.
2. Finden Sie heraus, wie die Ameise gedreht werden kann.
3. Schreiben Sie eine Methode innerhalb der Klasse Ameise, die die Ameise um einen Schritt in die Richtung versetzt, in die sie schaut. Hinweise zum mathematischen Sachverhalt können Sie der folgenden Grafik entnehmen:



4. Schreiben Sie eine Methode, die den Blickwinkel der Ameise durch Zufall etwas verändert (beispielsweise um einen Winkel zwischen  $-30^\circ$  und  $30^\circ$ (exklusive)) und die Ameise dann einen Schritt in diese Richtung gehen lässt.
5. Verändern Sie die Methode `handle()` in der Klasse `Ameise`, so dass die Ameise nun zufällig durch die Gegend läuft.
6. Setzen Sie mit einer Schleife 50 Ameisen, die dann alle zufällig über die Anzeige laufen.
7. Verändern Sie die Methode aus Punkt 4 so, dass die Ameisen das Feld nicht mehr verlassen können, sondern am Rand umkehren. Dazu muss die Methode wissen, wie breit und hoch das Feld ist.
8. Befindet sich eine Ameise nach einem Schritt in der Nähe des rechten Randes, soll sie verschwinden.
9. Für jede verschwundene Ameise soll eine Ameise in Punkt (100,100) neu geboren werden.

## Aufgabe 2

Klassen dieser Aufgabe:

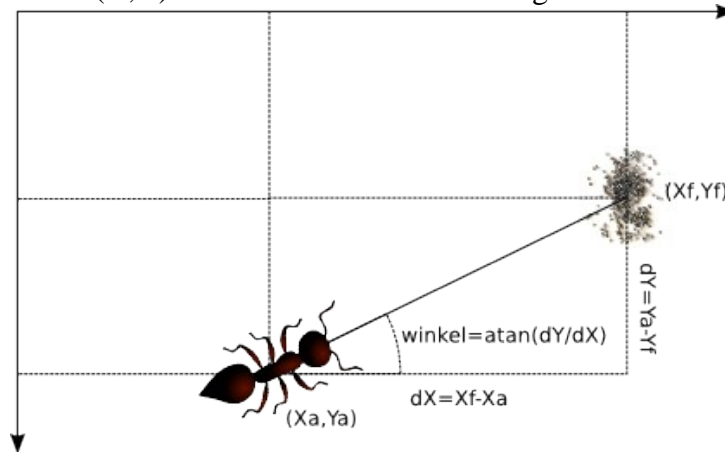
- Ameisenhaufen
  - Ein Objekt dieser Klasse stellt einen Ameisenhaufen dar.  
Hinweis: Diese Klasse sollen Sie nur benutzen, NICHT verändern. Sie haben JavaDoc zur Verfügung, um zu erfahren, was die Klasse macht.
- Futterhaufen
  - Ein Objekt dieser Klasse stellt einen Futterhaufen mit einer Anzahl Futtereinheiten dar, welche die Ameisen verspeisen können.  
Hinweis: Diese Klasse müssen Sie ergänzen, damit sie das macht, was in der Aufgabenstellung vorgesehen ist.
- Hilfsklasse
  - Diese Klasse bitte ergänzen, damit sie das macht, was in der Aufgabenstellung vorgesehen ist.

Aufgabenstellung:

1. Entfernen Sie das Sterben und Entstehen der Ameisen aus Aufgabe 1 aus dem Programm. Die Ameisen sollen am rechten Rand nun ebenfalls umkehren (vgl. Aufgabe 1.7)
2. Sie haben inzwischen das Konzept der statischen Methoden kennen gelernt. Die mathematischen Methoden sind bisher nicht statisch. Diese Methoden sollten statisch werden. Führen Sie diese Änderung in Ihrem Programm durch.
3. Fügen Sie den vorgegebenen Ameisenhaufen in Ihre Simulation ein.
4. Implementieren Sie einen Futterhaufen mit einer bestimmten Anzahl von Futtereinheiten.  
Anmerkung: Alle Methoden und Felder von `Grafikobjekt` müssen auch in dieser Klasse

vorhanden sein!

5. Fügen Sie eine Liste von Futterhaufen in die Simulation ein.
6. Schreiben Sie eine Methode, um eine Ameise in die Richtung eines bestimmten Koordinatenpunktes (X,Y) schauen zu lassen. Dazu folgende Skizze mit Hinweisen:



Hinweis zum gezielten Laufen:

In der Klasse Math existiert eine große Anzahl statischer mathematischer Methoden. Zu beachten ist, dass der von atan zurückgelieferte Winkel nur zwischen  $-\pi/2$  und  $\pi/2$  ist! Die Ameise muss zusätzlich um  $\pi$  gedreht werden, wenn das Futter sich weiter links als die Ameise befindet. Wenn das Futter genau unter oder über der Ameise ist ( $dX = X_f - X_a = 0$ ), würde es bei der Berechnung  $dY/dX$  eine Division durch Null geben! Dieser Spezialfall muss manuell behandelt werden.

7. Schreiben Sie eine Methode, um zu überprüfen, ob sich eine Ameise „in der Nähe“ eines bestimmten Punktes befindet. In welcher Klasse sollte diese Methode definiert werden?
8. Schreiben Sie nun das Programm so um, dass die Ameisen zunächst wie in Aufgabe 1 zufällig durch die Gegend laufen. Wenn sie nahe eines Futterhaufens sind, nehmen sie von dort eine Futtereinheit mit, merken sich die Position und laufen zum Ameisenhaufen. Treffen sie auf dem Weg vom Futterhaufen zum Ameisenhaufen eine andere Ameise, die kein Futter hat, sagen sie dieser die Position des Futterhaufens weiter. Die Ameisen laufen dann so lange zwischen Futterhaufen und Ameisenhaufen hin und her, bis im Futterhaufen keine Futtereinheiten mehr vorhanden sind. Danach laufen sie wieder zufällig über das Feld.
9. Fügen Sie die vom Ameisenhaufen „geborenen“ Ameisen ebenfalls in das Feld ein.  
Hinweis: Klasse Ameisenhaufen, Methode gibGeboreneAmeise().  
Der Ameisenhaufen erzeugt nur maximal alle 10 Sekunden neue Ameisen, die meiste Zeit über wird hier also keine neue Ameise entstehen.  
Besonders wichtig: Die Methode, die die Ameise nun steuert, muss überschaubar und an geeigneter Stelle mit geeigneten Parametern (gut wartbar) untergebracht sein.
10. Zusatzaufgabe: Schaffen Sie die Möglichkeit, beliebig viele Ameisenhaufen mit jeweils eigenem Ameisenvolk auf dem Anzeigefeld zu simulieren.  
Hinweis: Sie benötigen eine Sammlung von Ameisenvölkern bzw. Ameisenhaufen.

### Aufgabe 3

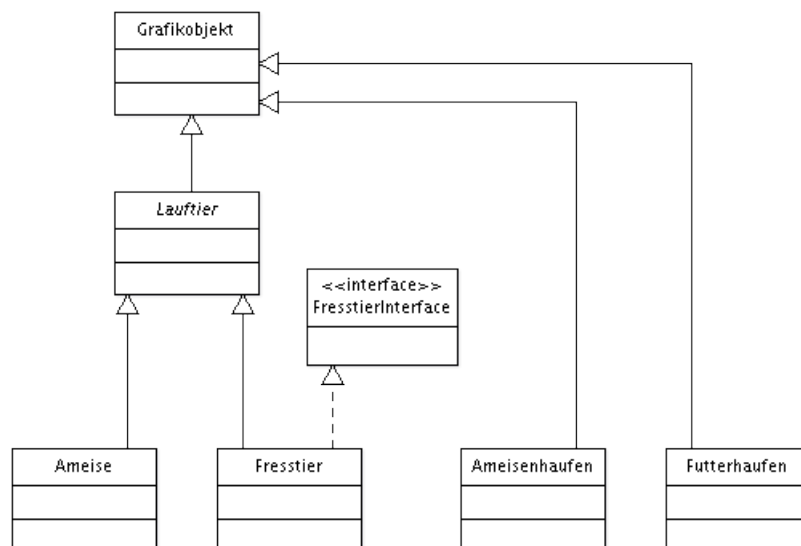
Wechseln Sie für die Bearbeitung dieser Aufgabe von BlueJ nach Eclipse. Wie die Migration von BlueJ zu Eclipse funktioniert, können Sie in einem eigenen Dokument nachlesen. Sie werden die erweiterten Möglichkeiten, die Eclipse für das Refactoring bietet, sehr zu schätzen wissen.

Neue Klassen und Interfaces:

- **FresstierInterface**
  - Ist ein Interface für ein Fresstier (beispielsweise ein Käfer, Vogel oder ähnliches).  
Hinweis: Dieses Interface dürfen sie NICHT verändern.

Aufgabenstellung:

- Inzwischen kennen Sie das Konzept der Vererbung. Sie haben sicherlich bemerkt, dass es sehr umständlich ist, jedes Mal die Methoden und Felder des Grafikobjekts in die selbst erstellten Klassen zu kopieren. Und denken Sie erst an den Aufwand, wenn sich dort etwas ändern würde! Die Änderung müsste an vielen Stellen durchgeführt werden und dürfte an keiner Stelle vergessen werden. Die Lösung des Problems: Lassen Sie alle Klassen, die auf dem Anzeigefeld dargestellt werden sollen, von der Klasse Grafikobjekt erben und entfernen Sie den doppelten Code aus den Klassen! Lassen Sie jedoch nicht alle Ihrer Klassen direkt von Grafikobjekt erben, sondern erstellen Sie selbst eine u.U. abstrakte Klasse, die von Grafikobjekt erbt, von der ein Teil Ihrer darzustellenden Klassen erben. In diese Klasse kommen alle Methoden, die gemeinsam sind. Erst von dieser selbst erstellten Klasse lassen Sie dann Ihre darzustellenden Klassen erben. Eine unverbindliche Beispielhierarchie sehen Sie in der folgenden Grafik:  
Durchgezogene Pfeile bedeuten hier „erbt von“, der gestrichelte Pfeil bedeutet „implementiert das Interface“.



- Implementieren Sie das Interface FresstierInterface in einer eigenen Klasse! Lassen Sie das Fresstier sinnvoll über das Anzeigefeld laufen und Ameisen fressen.

- Erweitern Sie freiwillig das Programm um beliebige tolle Fähigkeiten. Das beste Programm wird mit dem „mi.ants.getAward()“ versehen, der mit einem Preisgeld in Höhe von 1 Cent dotiert ist. Die Jury setzt sich aus hochrangigen Mitarbeitern des MI-Assistenten-Stabs zusammen. Teilnahmeberechtigt sind ausschließlich die an der Veranstaltung GPI1 teilnehmenden Studierenden. Der Rechtsweg ist ausgeschlossen. Als gute Quelle für Inspiration kann z.B. <http://de.wikipedia.org/wiki/Ameisen> dienen.