Aufgabenblatt 1

Allgemeine Anmerkungen

Ihre Lösung für dieses Aufgabenblatt ist bis Freitag, 13.3. 13h durch `git commit` und `push` abzugeben. Mit der Angabe werden folgende Dateien mitgeliefert `Space.java` und `Body.java`. Vorgegebene Programmteile dürfen nur an den Stellen verändert werden, die mit TODO markiert sind. Zusätzliche Klassen, Interfaces, Methoden und Variablen dürfen aber eingefügt werden. Wenn Sie zusätzlich zu den gefragten Klassen, weitere Klassen definieren, achten Sie darauf, dass die Klassennamen mit `My` beginnen, z.B. `MyVector3D`, um Konflikte mit späteren Aufgabenblättern zu vermeiden.

Thema

Das allgemeine Thema dieses und der kommenden Aufgabenblätter ist der Weltraum und die Simulation der physikalischen Gesetze, die für Himmelskörper gelten. Obwohl ein möglichst exaktes physikalisches Modell wünschenswert ist, ist bei der Implementierung die Genauigkeit der physikalischen Modelle sekundär. Konzeptuelle Fehler bei den physikalischen Berechnungen spielen keine Rolle bei der Bewertung. Schwerpunkt sind die Konzepte der Programmiersprache.

Ziel

Ziel der Aufgabe ist der Entwurf, die Implementierung und das Verwenden eines Abstrakten Datentyps (siehe Skriptum Seiten 31-39).

Aufgaben

Gegeben ist die Datei `Space.java`. Diese beinhaltet die unvollständige Definition einer ausführbaren Klasse, die mit Objekten anderer Klassen arbeitet. Zumindest eine weitere Klasse müssen Sie für Ihre Lösung in der Datei `Body.java` selbst definieren. Sie können auch weitere Klassen definieren.

Bei Ausführung soll die vervollständigte ausführbare Klasse verlangte Berechnungen durchführen und in `Space.java` angegebene Sollausgaben erzeugen. Sie sollen auch weitere eigene Testfälle gemäß der untenstehenden Aufgabenstellung überlegen.

Die Klasse `Body` (50%)

Definieren Sie die Klasse `Body`, die Körper im Weltraum (oder ähnlichen Umgebungen) repräsentiert. Führen Sie dabei folgende Arbeitschritte aus:

1. Beschreiben Sie alle Methoden der Klasse, insbesondere die Methoden, die Sie implementieren wollen. Diese Beschreibung soll später als Kommentar im Programmcode stehen.

2. Implementieren Sie die Klasse nach Ihrer Beschreibung.
3. Testen Sie die Klasse. Überlegen Sie sich dazu entsprechende Testfälle (gemäß Punkt 2, siehe unten).
4. Überlegen Sie sich Antworten auf die Zusatzfragen (unter Abschnitt Fragen).

Beschreibung der Klasse `Body`: Jedes Objekt dieser Klasse hat eine Masse in Kilogramm und eine Position mit 3 `double`-Koordinaten (x,y,z) im Raum (Meter). Körper bewegen sich außerdem gemäß eines 3D-Bewegungsvektors (vx,vy,vz), der jedem Körper zugeordnet ist, z.B. (1.0,1.2,0.0) in Metern pro Sekunde. Der Vektor gibt an, wo relativ zur momentanen Position der Körper als nächstes positioniert wird. Eine entsprechende Nachricht `move()` soll diese neue Positionierung durchführen. Diese ergibt sich aus (x+vx,y+vy,z+vz) . Wirken keine weiteren Kräfte auf den Körper, bleibt der Bewegungsvektor (vx,vy,vz) dabei auf Grund der Trägheit des Körpers unverändert (für die nächste Nachricht `move()`). Eine Nachricht `move(double fx, double fy, double fz)` positioniert den Körper so wie `move()`, mit dem Unterschied dass (fx,fy,fz) einem 3D-Kräftevektor (Einheit: Newton) entspricht und sich der neue Bewegungsvektor aus dem momentanen Bewegungsvektor plus (fx,fy,fz) dividiert durch Masse des Körpers ergibt. Mit diesem Vektor wird die neue Position bestimmt. Damit ändert sich auch der momentane Bewegungsvektor für die nächste Nachricht `move()` oder `move(fx,fy,fz)` entsprechend.

Instanzen verwenden (50%)

Erzeugen Sie mehrere Instanzen der Klasse und experimentieren Sie damit:

1. Ein Himmelskörper fliegt mit konstanter Geschwindigkeit in eine angegebene Richtung. Annahme: Es wirken keine Kräfte auf ihn. Geben Sie seine Position nach einer angegebenen Zeit an. Die Anzahl von Nachrichten `move()` entspricht den vergangenen Sekunden.
2. Eine Rakete im leeren Raum soll mit einem Bewegungsvektor (0,0,0) initialisiert werden, auf sie wirkt eine konstante Kraft in z-Richtung z.B. (0,0,5e6). Geben Sie die Position der Rakete nach einer spezifizierten Anzahl (Variable) von Bewegungsschritten (entspricht Sekunden) aus. Freiwillige Erweiterung: Die Rakete verliert durch fuel-burn pro Sekunde an Masse.
3. Ein Ball fällt im luftleeren Raum aus einer spezifizierbaren Höhe zu Boden und wird dabei von der Erdanziehungskraft beschleunigt. Nach wievielen Sekunden (entspricht Anzahl der Nachrichten `move(fx,fy,fz)`) trifft der Ball auf dem Boden auf?
4. Eine Feder wird vom Wind getragen. Die einwirkende Kraft ändert sich von Sekunde zu Sekunde zufällig.

Fragen

Wie könnte man `move(seconds, fx,fy,fz)` implementieren? Diese Nachricht soll bewirken, dass mehrere Bewegungsschritte durchgeführt werden.

Bedingungen für die Implementierung

Die gesamte Implementierung soll ohne Schleifen auskommen. Definieren Sie stattdessen rekursive Methoden.