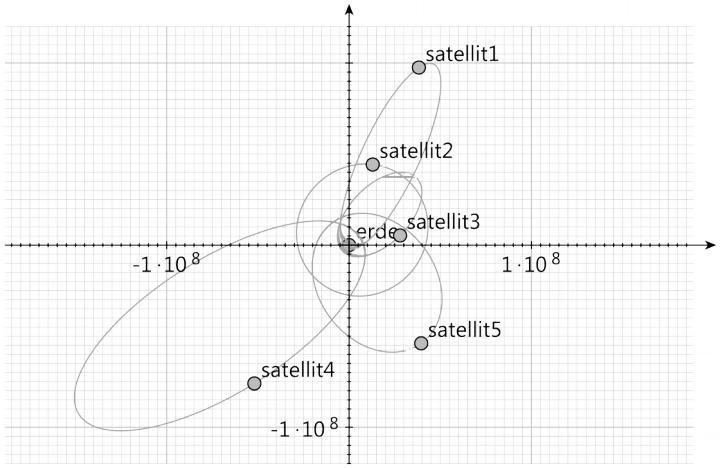
Aufgabenblatt 9

Satelliten, objektorientiert programmiert



1. Teilaufgabe

In Aufgabenblatt 8 wurde ein ein physikalisches System programmiert, bei dem ein Satellit um die Erde kreist. Dabei waren alle physikalischen Variablen skalare Größen, also einzelne double-Werte. In dieser Teilaufgabe soll die gleichen Aufgabenstellung noch einmal bearbeitet werden. Dabei soll jedoch mit Vektoren gearbeitet werden. Die Position des Satelliten, seine Geschwindigkeit und seine Beschleunigung sollen durch Vektoren repräsentiert werden. Die physikalischen Formeln sollen mit Hilfe von Vektoroperationen (Vektoraddition, Skalarprodukt etc.) ausgedrückt werden.

Die Position des Satelliten wurde bisher durch die beiden skalaren Variablen *x* und *y* ausgedrückt, seine Geschwindigkeit durch *vx* und *vy* und seine Beschleunigung durch *ax* und *ax*. Entfernen Sie diese Variablen aus Ihrem Programmcode und fügen Sie gemäß dem nachfolgenden Programmcode die folgenden Objektattribut-Deklarationen für *r*, *v* und *a* hinzu. Die Variable *r* soll für die Position des Satelliten stehen, *v* für seine Geschwindigkeit und *a* für die Beschleunigung, die der Satellit erfährt. Alle drei Variablen haben den Typ *Vector2D*. Damit handelt es sich bei den drei Variablen um Pärchen von Zahlen. Jede der drei Variablen hat je einen *x*- und einen *y*-Anteil vom Typ *double*. Der Java-Ausdruck *r.x* steht für den *x*-Anteil der Position *r* und *r.y* steht für den *y*-Anteil von *r.* Entsprechend stehen *v.x*, *v.y*, *a.x* und *a.y* für die *x*- und *y*-Anteile von *v* und *a.*

```
import de.physolator.usr.components.*;
import static de.physolator.usr.components.VectorMath.*;
...

public class Satelliten extends PhysicalSystem {
    ...

    @V(unit = "m", derivative = "v")
    public Vector2D r = new Vector2D(4e7, 0);

    @V(unit = "m/s", derivative = "a")
    public Vector2D v = new Vector2D(0, 4000);

    @V(unit = "m/s^2")
    public Vector2D a = new Vector2D(0, 0);

    ...
}
```

Hinweis: Die Klasse *Vector2D* befindet sich im Paket *de.physolator.usr.components*. Deshalb benötigt man im Anfangsbereich des Programmcodes das entsprechende *import*-Statement.

Bei der Initialisierung müssen den Vektor-Variablen Vektoren zugewiesen werden, die mit *new Vector(x,y)* erzeugt werden. Man bezeichnet dies in Java als Konstruktoraufrufe. Hier werden die Vektoren erzeugt und ihnen werden als Anfangswerte jeweils zwei Werte *x* und *y* zugewiesen. Bei *r* sind dies die Werte 4e7 und 0, bei *v* sind es 0 und 4000 und bei *a* sind es 0 und 0.

Auch zwischen Vektoren können Ableitungsbeziehungen festgelegt werden. Durch die Annotations wird festgelegt, dass *v* die erste Ableitung von *r* ist und das *a* die erste Ableitung von *v* ist. Dies geschieht in gleicher Weise wie bei skalaren Größen. Auch Vektoren können physikalische Einheiten zugeordnet werden. Wenn etwa für die physikalische Variable *r* festgelegt wird, dass die physikalische Einheit *m* sein soll, dann bedeutet dies nicht nur, dass *r* die Einheit *m* erhält, sondern es bedeutet gleichzeitig auch, dass die Teile von *r*, nämlich *r.x* und *r.y* auch diese Einheit erhalten.

Aus Java kann man auf die *x*- und *y*-Anteile zugreifen und mit diesen *double*-Werten gewöhnliche arithmetische *double*-Operationen durchführen (Addition, Multiplikation, Sinus,...). Alternativ kann man aber auch auf der Ebene der Vektoren rechnen. Schreiben sie die Formeln in der Methode *f* so um, dass Sie mit Vektorrechnung arbeiten. Die Klasse *VectorMath* stellt dazu verschieden Vektoroperationen bereit. Die Vektoroperationen sind in der nachfolgenden Tabelle aufgelistet. Verwenden Sie bei der Implementierung der Methode *f* diese Operationen.

add(a,b)	Vektoraddition. Addiert zwei Vektoren (*Vector2D*-Objekte). Das Ergebnis ist wieder ein Vektor (ein *Vector2D*-Objekt).
add(a,b,c) add(a,b,c,d) add(a,b,c,d,e) ...	Vektoraddition mit mehreren Vektoren
sub(a,b)	Vektorsubtraktion.
mult(p,a)	Skalarprodukt. Multipliziert die (skalare) double-Zahl *p* mit dem Vektor *a*. Das Ergebnis ist wieder ein Vektor.
abs(a)	Berechnet den Betrag eines Vektors. Der Betrag ist eine skalare Größe vom Typ *double*.
normalize(a)	Normalisierung. Berechnet zu einem Vektor einen Vektor mit der Länge 1 und der gleichen Richtung.
dist(a,b)	Der Abstand zwischen zwei Punkten *a* und *b*. *a* und *b* sind vom Typ Vector2D, das Ergebnis ist vom Typ *double*.

Bei der Programmierung von physikalischen Systemen für den Phyosolator gibt es eine Einschränkung: Man darf einem Vektor *v* nicht durch eine Zuweisung

```
v = p;
```

einen Wert *p* zuweisen, sondern man muss dem Vektor den Wert durch den Aufruf der Methode

```
v.set(p);
```

zuweisen. Beachten Sie bei der Implementierung der Methode *f* diese Regel. Diese Regel gilt nur für Vektoren, nicht für die skalaren Größen vom Typ double. Wenn irgend einer Variable *z* vom Typ *double* ein Wert zugewiesen werden soll, so geschieht das weiterhin mit einer Zuweisung der Form *z=...;*

Bei der bisherigen Implementierung wurde der Satellit mit Hilfe der Klasse *MechanicsTVG* graphisch dargestellt. Mit dem Befehl

```
t.addPointMass("x", "y", "vx", "vy", "ax", "ay");
```

wurde festgelegt, dass eine punktförmige Masse dargestellt werden soll, die über diese sechs skalaren Variablen definiert wird. Dieser Befehl kann ersatzlos gestrichen werden. Bei physikalischen Systemen, die in der oben dargestellten Form mit Vektoren mit den Namen *r*, *v* und *a* arbeiten, erkennt die Klasse *Mechanics* automatisch, dass es sich um eine punktförmige Masse handelt und stellt diese ohne weiteres Zutun dar.

2. Teilaufgabe

Übernehmen Sie die folgende Klasse *Himmelskoerper*.

```
import de.physolator.usr.V;
import de.physolator.usr.components.Vector2D;

public class Himmelskoerper {

    @V(unit = "kg")
    public double m = 5.974E24;

    @V(unit = "m", derivative = "v")
    public Vector2D r = new Vector2D(0, 0);

    @V(unit = "m/s", derivative = "a")
    public Vector2D v = new Vector2D(0, 0);

    @V(unit = "m/s^2")
    public Vector2D a = new Vector2D(0, 0);

    public Himmelskoerper(double x, double y, double vx, double vy, double ax, double ay,
                          double m) {
        r.set(x,y);
        v.set(vx,vy);
        a.set(ax,ay);
        this.m = m;
    }
}
```

Die Klasse *Himmelskoerper* repräsentiert Objekte, die eine Masse *m* haben, eine Position *r*, eine Geschwindigkeit *v*, und eine Beschleunigung *a*. Dabei ist *m* eine skalare Größe und *r*, *v* und *a* sind Vektoren. Den physikalischen Variablen dieser Klasse sind bereits die passenden physikalischen Variablen zugeordnet und auch die Ableitungsbeziehungen sind angegeben.

Sowohl die Erde als auch der Satellit sollen jetzt jeweils als ein Instanz der Klasse Himmelskoerper beschrieben werden. Übernehmen Sie dazu den folgenden Programmcode:

```
import static de.physolator.usr.components.VectorMath.*;
import static java.lang.Math.pow;
import mechanics.tvg.MechanicsTVG;
import de.physolator.usr.*;

public class Satelliten extends PhysicalSystem {

    @V(unit = "m^3/kg s^2")
    double G = 6.67428e-11;

    public Himmelskoerper erde = new Himmelskoerper(0, 0, 0, 0, 0, 0, 5.974E24);
    public Himmelskoerper satellit = new Himmelskoerper(4e7, 0, 0, 4000, 0, 0, 100);

    public void f(double t, double h) {
        // Platz für die Formeln zur Berechnung der abhängigen Variablen
    }

    public void initGraphicsComponents(GraphicsComponents g, Structure s, Recorder r,
            SimulationParameters sp) {
        MechanicsTVG t = new MechanicsTVG(this, s, r);
        double p = 1.2e8;
        t.geometry.setUserArea(-p, p, -p, p);
        t.showPaths = true;
        t.showVelocity = false;
        t.showAcceleration = false;
        t.showLabels = true;
        g.addTVG(t);
    }

    public void initSimulationParameters(SimulationParameters s) {
        s.fastMotionFactor = 20000;
    }
}
```

In diesem Programmcode sind bereits zwei Instanzen der Klasse Himmelskoerper erzeugt worden. Eine der beiden Instanzen wird in der Variablen *erde* gespeichert, die andere in der Variablen *satellit*. Die Konstruktoraufrufe *new Himmelskoerper(...)* erzeugen die Himmelskoerper. Die Parameterwerte der Konstruktoraufrufe legen die Anfangswerte der Objekte fest: deren Position *r*, deren Geschwindigkeit *v*, deren Beschleunigung *a* und deren Masse *m*.

Sowohl die Erde als auch der Satellit haben jeweils einen *r*-Anteil, einen *v*-Anteil, einen *a*-Anteil und einen *m*-Anteil. Über die Variablen *erde* und *satellit* kann auf diese Anteile zugegriffen werden. So steht *satellit.r* für die Position des Satelliten, *satellit.v* für dessen Geschwindigkeit, *satellit.a* für dessen Beschleunigung und *satellit.m* für dessen Massen. Analog dazu stehen *erde.r, erde.v, erde.a* und *erde.m* für die Position, die Geschwindigkeit, die Beschleunigung und die Masse der Erde.

Berechnen Sie damit in der Methode *f* die Beschleunigung des Satelliten!

Anmerkungen: Die Klasse *MechanicsTVG* zeichnet automatisch alle punktförmige Massen des physikalischen Systems, also alle physikalischen Teilkomponenten, die einen *r*-Anteil, einen *v*-Anteil und einen *a*-Anteil haben. Somit werden Erde und Satellit automatisch eingezeichnet. Die Erde musste bisher gesondert gezeichnet werden. Das ist jetzt nicht mehr notwendig. Werden auf dem Bildschirm, wie in diesem Fall, mehrere punktförmige Massen eingezeichnet, so könnte es zu Verwechslungen kommen. Die Zuweisung *t.showLabels = true;* sorgt dafür, dass die punktförmigen Massen mit ihren Variablennamen beschriftet werden.

3. Teilaufgabe

Ergänzen Sie in der Klasse *Himmelskoerper* die folgende Methode *gravitationsbeschleunigung*.

```
public Vector2D gravitationsbeschleunigung(Vector2D p) {
    return ...; // Programmcode fehlt noch
}
```

Die Methode *gravitationsbeschleunigung* ist Teil eines *Himmelskoerper*-Objekts. Die Methode soll bestimmen, welche Gravitationsbeschleunigung eine punktförmige Masse durch das *Himmelskoerper*-Objekt erfährt, wenn sich die punktförmige Masse an der Position *p* befindet. Dazu greift die Methode auf die Objektattribute des *Himmelskoerper*-Objekts *r*, *v, a* und *m* zu und auf den Parameter *p* und berechnet die Beschleunigung in Form eines Vektors (Typ *Vector2D*). Die berechnete Beschleunigung wird als Rückgabewert zurückgegeben.

Empfehlung: Ergänzen Sie in der Klasse *Himmelskoerper* eine physikalische Variable *G* mit der Gravitationskonstante.

Diese neue Objektmethode in der Klasse *Himmelskoerper* kann nun in allen *Himmelskoerper*-Objekten verwendet werden. In der Klasse *Satelliten* können mit *erde.gravitationsbeschleunigung(p)* zu einem beliebigen Punkt *p* die von der Erde ausgehende Gravitationsbeschleunigung bestimmt werden und analog mit *satellit.gravitationsbeschleunigung(p)* die Gravitationsbeschleunigung, die vom Satelliten ausgeht. Verwenden Sie diese neue Methode bei der Programmierung von *f* !

4. Teilaufgabe

Bisher umkreist nur ein Satellit die Erde. Jetzt sollen fünf Satelliten die Erde umkreisen.

Erzeugen Sie dazu in der Klasse Satelliten fünf Variablen satellit1, satellit2, ... satellit5 und weisen Sie diesen Satelliten unterschiedliche Anfangspositionen und Anfangsgeschwindigkeiten zu! Sorgen Sie dafür, dass deren Beschleunigung in der Methode *f* berechnet wird!