Aufgabe 08 1 9/10

```
package uebung081;

public class Star {

	private String name;
	private String id;
	private double distance;
	private double apparentMagnitude;
	private String type;



	public Star() {

	}
	//Konstruktor mit Name und ID
	public Star(String name, String id) {
		setName(name);
		setId(id);
```

```
}

//Zugriff mit getter und setter
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
public String getId() {
        return id;
}
public void setId(String id) {
        this.id = id;
}
public double getDistance() {
        return distance;
}
public void setDistance(double distance) {
```

```
        this.distance = distance;
}
public double getApparentMagnitude() {
        return apparentMagnitude;
}
public void setApparentMagnitude(double apparentMagnitudes) {
        this.apparentMagnitude = apparentMagnitudes;
}
public String getType() {
        return type;
}
public void setType(String type) {
        this.type = type;
}

//Methoden

public boolean isCloserThan(Star other) {
        if(this.getDistance() < other.getDistance()) {
```

if/else ist heir nicht notwenig, return this.getDistance() < other.getDistance würde das gleich Ergbenis liefern (und wäre kürzer)

```
                return true;
        }

        else {
                return false;
        }
}

public double getDistanceInPc() {
        return this.getDistance() * 3.26;
}

public double getAbsoluteMagnitude() {
        double log = Math.log10(this.getDistanceInPc());
        double M = this.getApparentMagnitude() - 5 * log + 5;
        return M;



}
```

```
public boolean checkType() {
	if(this.getAbsoluteMagnitude() <= 20 && this.getAbsoluteMagnitude() >= 15) {
		if(this.getType().equals("Brown dwarf")) {
			return true;
		}
		else {
			return false;
		}
	}
	if(this.getAbsoluteMagnitude() <= 15 && this.getAbsoluteMagnitude() >= 10) {
		if(this.getType().equals("Red dwarf") || this.getType().equals("White dwraf") ) {
			return true;
		}
		else {
			return false;
		}
	}
```

```
            //es folgen noch die weiteren Bedingungen aus der Tablle....      das fehlt :/

            else {
                return false;
            }
        }
}
```

```java
package uebung081;

public class StarDatabase {

	private String name;
	private Star[] stars;
	private Star[] magnitudeRange;


	public String getName() {
		return name;
	}

	public StarDatabase() {

	}

	public Star getStars(int index) {
		return stars[index];
```

```
}


//Array mit erstem Stern erstellen
public StarDatabase(String name) {
        this.name = name;
        stars = new Star[0];
}

public void add(Star star) {
        // temporäres Array zum Kopieren anlegen und altes array reinkopieren
        Star[] tmparray = new Star[stars.length + 1];
        for(int j = 0; j < stars.length; j++) {
                tmparray[j] = stars[j];

        }
        tmparray[stars.length] = star;
        stars = tmparray;
```

```
        System.out.println(stars.length);

    }

    public void remove(int index) {
        Star[] tmparray = new Star[stars.length - 1];
        for(int i = 0; i < index; i++) {
            tmparray[i] = stars[i];
        }

        for(int k = index + 1; k < stars.length; k++) {
            tmparray[k - 1] = stars[k];
        }

        stars = tmparray;
        System.out.println(stars.length);

    }
```

```
public Star get(int index) {
        return stars[index];
}

public int size() {
        return stars.length;
}

public Star find(String id) {
        Star result = null;
        for(int i = 0; i < stars.length; i++) {
                String id1 = stars[i].getId();
                if( id.equals(id1)) {
                        result = this.getStars(i);
                }


        }
```

```
            return result;



        }

        public Star[] getMagnitudeRange(double low, double high) {
            for(int i = 0; i < stars.length; i++) {
                double mr = stars[i].getApparentMagnitude();
                if ( low <= mr && mr <= high) {
                    magnitudeRange[i] = this.getStars(i);
                }
            }
            return magnitudeRange;
        }
}
```

das magnitudeRange-Array wird nicht initialisiert

Aufgabe 2 4/4

```
package uebung082;

public class Account {

	private String besitzer;

	private double kontostand;

	public Account(String besitzer) {
		this.besitzer = besitzer;
	}
	public String getBesitzer() {
		return besitzer;
	}

	public void setBesitzer(String besitzer2) {
		besitzer = besitzer2;
	}
```

```
public double getKontostand() {
        return kontostand;
}

public void auszahlung(double betrag) {
        if(kontostand - betrag >= 0 ) {
        kontostand = kontostand - betrag;
        }
        else {
                System.out.println("Ihr Kontostand ist zu niedrig");
        }
}

public void einzahlung(double ez) {
        if (ez > 0) {
                kontostand = kontostand + ez;
        }
```

```
        else {
            System.out.println("Sie können nur Einzahlung > 0 vornehmen");
        }
    }

    public boolean transfer(double amout, Account other) {
        if((this.getKontostand() - amout) >= 0 ) {
            this.auszahlung(amout);
            other.einzahlung(amout);
            return true;
        }
        else {
            return false;
        }
    }


}
```

So wird 2x geprüft, ob der Betrag ausgezahlt werden kann (einmal hier und nochmal in der auszahlungsmethode. Diese doppelte Überprüfung könnte man z.B. vermeiden, indem man bei der Auszahlung einen boolean zurückgibt

```java
package uebung082;

public class AccountTest {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Account accountDonald = new Account("Donald Duck");

		accountDonald.einzahlung(100);
		accountDonald.einzahlung(-100);

		accountDonald.auszahlung(5000);

		Account accountScrooge = new Account("Scrooge McDuck");

		accountDonald.transfer(800, accountScrooge);
		accountDonald.transfer(50, accountScrooge);
		accountDonald.transfer(-50, accountScrooge);
```

```
        System.out.println(accountDonald.getKontostand()); // 50
        System.out.println(accountScrooge.getKontostand()); // 50

    }

}
```

Aufgabe 3

```
package uebung083;   6/6

public class Cube {

	private double edgeLength = 0;
	private double surfaceArea = 0;
	private double volume = 0;

	public Cube(double edgeLength) {
		this.edgeLength = edgeLength;
	}

	public double getEdgeLength() {
		return edgeLength;
	}

	public double getSurfaceArea() {
		surfaceArea = 6 * edgeLength * edgeLength;
```

```
            return surfaceArea;
        }

        public double getVolume() {
            volume = edgeLength * edgeLength * edgeLength;
            return volume;
        }

        public void change(double increment) {
            edgeLength = edgeLength + increment;
        }

}
package uebung083;

public class Sphere {

        private double radius;
        private double surfaceArea;
```

```
private double volume;

public Sphere (double radius) {
        this.radius = radius;
}

public double getSurfaceArea() {
        surfaceArea = 4 * Math.PI * Math.pow(this.radius, 2);
        return surfaceArea;
}

public double getVolume() {
        volume = (4 * Math.PI * Math.pow(this.radius, 3)) / 3;
        return volume;
}

public void change(double increment) {
        this.radius = this.radius + increment;
}
```

```
}
```

```java
package uebung083;
import java.util.Scanner;

public class VolumenvonBehältern {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner scanner = new Scanner(System.in);
		System.out.print("Edge length: ");
		double kantenlänge = scanner.nextDouble();
		System.out.print("Radius: ");
		double radius = scanner.nextDouble();

		//Prüfen ob werte Korrekt
		if(kantenlänge > 0 && radius > 0 ) {
			Cube cube1 = new Cube(kantenlänge);
			Sphere sphere1 = new Sphere(radius);

			System.out.println("Cube volume: " + cube1.getVolume());
```

```
System.out.println("Sphere volume: " + sphere1.getVolume());

//Prüfen auf Wert Korrekt
System.out.println("Increment: ");
double inkrement = scanner.nextDouble();

if(inkrement > 0) {

if(cube1.getVolume() > sphere1.getVolume()) {
while (cube1.getVolume() > sphere1.getVolume()) {
        cube1.change(inkrement * (-1));
        sphere1.change(inkrement);

}

        System.out.println("Cube volume: " + cube1.getVolume());
        System.out.println("Sphere volume: " + sphere1.getVolume());
        System.out.println("Cube surface: " + cube1.getSurfaceArea());
        System.out.println("Sphere surface: " + sphere1.getSurfaceArea());
```

Das hätte man auch ans Ende packen können, dann gibt es keinen doppelten Code

```
}
else if(cube1.getVolume() < sphere1.getVolume()) {
        while (cube1.getVolume() < sphere1.getVolume()) {
                cube1.change(inkrement);
                sphere1.change(inkrement * (-1));
        }

        System.out.println("Cube volume: " + cube1.getVolume());
        System.out.println("Sphere volume: " + sphere1.getVolume());
        System.out.println("Cube surface: " + cube1.getSurfaceArea());
        System.out.println("Sphere surface: " + sphere1.getSurfaceArea());
}

else {
        System.out.println("Die Volumina sind gleich groß");
}

}
```

```
                else {
                        System.out.println("Der Wert muss grösser als 0 sein");
                }

        }

        else {
                System.out.println("Die Berechnung kann nicht gestartet werden. Die Werte müssen größer als 0 sein.");
        }

        scanner.close();
    }

}
```