Enumerations

- **Datentyp** für Variablen mit **endlicher Wertemenge**.
  - Alle zulässigen Werte werden bei der Deklaration des Datentyps mit konstanten Namen definiert.

```
public enum Tribool {WAHR, FALSCH, VIELLEICHT}
```

- Zugriff auf Werte über den Punkt-Operator

```
Tribool myVar = Tribool.VIELLEICHT;
```

- Jeder Enum-Typ erbt von Java folgende statische Methoden:
  - `public static Tribool[] values()`
    - Gibt alle Werte als Array zurück
  - `public Tribool valueOf(String s)`
    - Gibt den Wert zurück, der s entspricht

Probleme ohne Enum

- In den USA gibt es für gewisse Cent-Beträge eigene Namen (z.B. Penny, Nickle, Dime, Quarter). Wir wollen diese Beträge modellieren:

```
public class CurrencyDenom {
   public static final int PENNY = 1;
   public static final int NICKLE = 5;
   public static final int DIME = 10;
   public static final int QUARTER = 25;
}

public class Currency {
   private int currency; //CurrencyDenom.PENNY,CurrencyDenom.NICKLE,
                         // CurrencyDenom.DIME,CurrencyDenom.QUARTER
}
```

- Nicht Typ-Sicher: Man kann jedwedgen Wert zur Variable currency hinzufügen (z.B. 78), obwohl es dafür keine Münze gibt
- Keine sinnvolle Ausgabe: Wenn man den Namen von z.B. Nickle ausgeben will (es kommt 5 raus, statt „Nickle“)
- Kein eigener Namensraum (man muss immer CurrencyDenom vor der eigentlichen Konstante schreiben

Besser mit Enums

- Typsicherheit durch festgelegte Werte

```
public enum Currency {PENNY, NICKLE, DIME, QUARTER};
Currency coin = Currency.PENNY;
coin = 1; //compilation error
```

Typsicher

- Ein einzelner, eigener Namensraum

```
for(Currency coin: Currency.values()){
        System.out.println("coin: " + coin);
}
```

Ausgabe:
coin: PENNY
coin: NICKLE
coin: DIME
coin: QUARTER

Aufgabe Enum

- **Erstellen sie eine Aufzählung, die die Planeten des Sonnensystems beinhaltet.**
  - Gespeichert werden sollen die **Masse** und der **Radius** des Planeten
  - MERCURY (3.303e+23, 2.4397e6), VENUS (4.869e+24, 6.0518e6), EARTH (5.976e+24, 6.37814e6), MARS (6.421e+23, 3.3972e6), JUPITER (1.9e+27, 7.1492e7), SATURN (5.688e+26, 6.0268e7), URANUS (8.686e+25, 2.5559e7), NEPTUNE (1.024e+26, 2.4746e7);
- **Zusätzlich soll die Aufzählung folgende Methoden besitzen:**
  - `public double getMass()`, `public double getRadius()`: Gibt vom jeweiligen Planeten die Masse und den Radius aus.
  - `public double surfaceGravity()`: Gibt die Gravitation des jeweiligen Planeten aus.
  - `public double surfaceWeight(double otherMass)`: Gibt das Gewicht auf den jeweiligen Planeten aus
- **Hinweise:**
  - Universelle Gravitationskonstante: double G = 6.67300E-11;
  - Formel Gravitation: G * MassePlanet / radiusPlanet^2
  - Gewicht auf Planet: Masse * Gravitation

Lösung Enum

```
public enum Planet {
    MERCURY (3.303e+23, 2.4397e6),
    VENUS   (4.869e+24, 6.0518e6),
    EARTH   (5.976e+24, 6.37814e6),
    MARS    (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27,   7.1492e7),
    SATURN  (5.688e+26, 6.0268e7),
    URANUS  (8.686e+25, 2.5559e7),
    NEPTUNE (1.024e+26, 2.4746e7);

    private final double mass;   // in kilograms
    private final double radius; // in meters
    Planet(double mass, double radius) {
        this.mass = mass;
        this.radius = radius;
    }
    private double mass() { return mass; }
    private double radius() { return radius; }

    // universal gravitational constant  (m3 kg-1 s-2)
    public static final double G = 6.67300E-11;
```

Alle gültigen Werte allerdings mit Konstruktoraufruf

Speichern der Masse und des Radius als Member. Konstruktor zur Initialisierung eines Planeten.

Gravitationskonstante

Lösung Enum

```
    double surfaceGravity() {
        return G * mass / (radius * radius);
    }
    double surfaceWeight(double otherMass) {
        return otherMass * surfaceGravity();
    }
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage: java Planet <earth_weight>");
            System.exit(-1);
        }
        double earthWeight = Double.parseDouble(args[0]);
        double mass = earthWeight/EARTH.surfaceGravity();
        for (Planet p : Planet.values())
           System.out.printf("Your weight on %s is %f%n",
                             p, p.surfaceWeight(mass));
    }
}
```

Methoden zur Berechnung der Gravitation und des Gewichts

Testklasse: Alle Planeten durchgehen mit foreach-Schleife

Aufgabe Enum II

- **Erstellen sie eine Aufzählung, die verschiedene Währungen enthält**
  - Gespeichert werden soll der der **Wechselkurs** zum Euro.

    EURO(1.00) , DOLLAR(1.07), GBP(0.84), AUD(1.44), CNY(7.38);
- **Zusätzlich soll die Aufzählung folgende Methode besitzen:**

  Die Möglichkeit, einen Wert in Euro per Switch-Case in die jeweilige Währung umzurechnen.
- **Hinweise:**
  - Universelle Gravitationskonstante: double G = 6.67300E-11;
  - Formel Gravitation: G * MassePlanet / radiusPlanet^2
  - Gewicht auf Planet: Masse * Gravitation