Erzeugt mit chat GPT, noch nicht vollständig geprüft S. 204|1

```
    class Tempmessung {
    private double[] temperaturen = new double[365];

    public void setzeZufallstemperaturwerte() {
        for (int i = 0; i < temperaturen.length; i++) {
            temperaturen[i] = Math.random() * 40;
        }
    }

    public int gibTagMitHöchsterTemperatur() {
        int tag = 0;
        double maxTemp = 0;
        for (int i = 0; i < temperaturen.length; i++) {
            if (temperaturen[i] > maxTemp) {
                maxTemp = temperaturen[i];
                tag = i;
            }
        }
        return tag;
    }

    public double gibNiedrigsteTemperatur() {
        double minTemp = Double.MAX_VALUE;
        for (int i = 0; i < temperaturen.length; i++) {
            if (temperaturen[i] < minTemp) {
                minTemp = temperaturen[i];
            }
        }
        return minTemp;
    }

    public double gibDurchschnittstemperatur() {
        double summe = 0;
        for (double temp : temperaturen) {
            summe += temp;
        }
        return summe / temperaturen.length;
    }
}
```

2)

```
    class Potenz {
    int[] pot;
    int basis;

    Potenz(int b, int länge) {
        pot = new int[länge];
        basis = b;
        pot[0] = 1;
        for (int i = 1; i < pot.length; i++) {
            pot[i] = pot[i - 1] * basis;
        }
    }
```

```
    //Ggf. Edge cases prüfen hier!
    public int gibPotenz(int n) {
        return pot[n];
    }
}
```

3a,b)

```
class bubbleSort {
    public int[] erzeugeZufallsfeld(int länge) {
        int[] feld = new int[länge];
        for (int i = 0; i < länge; i++) {
            feld[i] = (int) (Math.random() * 100);
        }
        return feld;
    }
    public void sortiereFeld(int[] feld) {
        int temp;
        for (int i = 0; i < feld.length - 1; i++) {
            for (int j = 0; j < feld.length - 1 - i; j++) {
                if (feld[j] > feld[j + 1]) {
                    temp = feld[j];
                    feld[j] = feld[j + 1];
                    feld[j + 1] = temp;
                }
            }
        }
    }
}
```

4)

```
class Wuerfel {
    public int werfe() {
        return (int) (Math.random() * 6) + 1;
    }

    public static void main(String[] args) {
        Wuerfel wuerfel = new Wuerfel();
        int[] ergebnisse100 = new int[6];
        int[] ergebnisse1000 = new int[6];
        int[] ergebnisse10000 = new int[6];

        // Würfle 100 mal
        for (int i = 0; i < 100; i++) {
            int wurf = wuerfel.werfe();
            ergebnisse100[wurf - 1]++;
        }

        // Würfle 1000 mal
        for (int i = 0; i < 1000; i++) {
            int wurf = wuerfel.werfe();
            ergebnisse1000[wurf - 1]++;
        }

        // Würfle 10000 mal
```

```
        for (int i = 0; i < 10000; i++) {
            int wurf = wuerfel.werfe();
            ergebnisse10000[wurf - 1]++;
        }

        // Berechne die relative Häufigkeit und Abweichung
        for (int i = 0; i < 6; i++) {
            double relativeHäufigkeit100 = (double) ergebnisse100[i] / 100;
            double relativeHäufigkeit1000 = (double) ergebnisse1000[i] / 1000;
            double relativeHäufigkeit10000 = (double) ergebnisse10000[i] / 10000;

            double idealRelativeHäufigkeit = 1.0 / 6;
            double abweichung100 = Math.abs(relativeHäufigkeit100 - idealRelativeHäufigkeit);
            double abweichung1000 = Math.abs(relativeHäufigkeit1000 - idealRelativeHäufigkeit);
            double abweichung10000 = Math.abs(relativeHäufigkeit10000 - idealRelativeHäufigkeit);

            // Gib Ergebnisse auf der Konsole aus
            System.out.println("Ergebnis " + (i + 1) + ":");
            System.out.println("Relative Häufigkeit (100 Würfe): " + relativeHäufigkeit100);
            System.out.println("Abweichung (100 Würfe): " + abweichung100);
            System.out.println("Relative Häufigkeit (1000 Würfe): " + relativeHäufigkeit1000);
            System.out.println("Abweichung (1000 Würfe): " + abweichung1000);
            System.out.println("Relative Häufigkeit (10000 Würfe): " + relativeHäufigkeit10000);
            System.out.println("Abweichung (10000 Würfe): " + abweichung10000);
            }
        }
}
```

5)

```
class Lotto {
    public int[] zieheKugeln(int k, int n) {
        int[] urne = new int[n];
        int[] gezogeneKugeln = new int[k];

        // fülle die Urne mit den Kugelnummern
        for (int i = 0; i < n; i++) {
            urne[i] = i + 1;
        }

        for (int i = 0; i < k; i++) {
            int zufallsindex = (int) (Math.random() * (n - i));
            gezogeneKugeln[i] = urne[zufallsindex];
            urne[zufallsindex] = urne[n - i - 1];
        }

        // Sortiere die gezogenen Kugeln mithilfe des Bubblesorts
        int temp;
        for (int i = 0; i < k - 1; i++) {
            for (int j = 0; j < k - 1 - i; j++) {
                if (gezogeneKugeln[j] > gezogeneKugeln[j + 1]) {
                    temp = gezogeneKugeln[j];
                    gezogeneKugeln[j] = gezogeneKugeln[j + 1];
                    gezogeneKugeln[j + 1] = temp;
                }
```

```
            }
        }
    return gezogeneKugeln;
    }
}
```

6)

```
    class Primzahlen {
    public int[] gibPrimzahlen(int n) {
        boolean[] istKeinePrimzahl = new boolean[n + 1];
        int[] primzahlen = new int[n];
        int anzahlPrimzahlen = 0;

        for (int i = 2; i <= n; i++) {
            if (!istKeinePrimzahl[i]) {
                primzahlen[anzahlPrimzahlen++] = i;
                for (int j = i * i; j <= n; j += i) {
                    istKeinePrimzahl[j] = true;
                }
            }
        }

        return primzahlen;
    }
}
```

7)

```
class Feldarbeiten {
    private int[] feld;
    private int n;

    public Feldarbeiten(int n) {
        this.n = n;
        this.feld = new int[n];
        Random rand = new Random();
        for(int i = 0; i < n; i++) {
            int zufallszahl = rand.nextInt();
            while(contains(zufallszahl)) {
                zufallszahl = rand.nextInt();
            }
            feld[i] = zufallszahl;
        }
    }

    private boolean contains(int zahl) {
        for(int i = 0; i < n; i++) {
            if(feld[i] == zahl) {
                return true;
            }
        }
        return false;
    }

    public int minimum() {
```

```
        int min = feld[0];
        for(int i = 1; i < n; i++) {
            if(feld[i] < min) {
                min = feld[i];
            }
        }
        return min;
    }

    public int summe() {
        int sum = 0;
        for(int i = 0; i < n; i++) {
            sum += feld[i];
        }
        return sum;
    }

    public double durchschnitt() {
        return (double) summe() / n;
    }

    public int anzahlKleinerAls(int k) {
        int count = 0;
        for(int i = 0; i < n; i++) {
            if(feld[i] < k) {
                count++;
            }
        }
        return count;
    }

    public String positionMinimum() {
        int min = minimum();
        for(int i = 0; i < n; i++) {
            if(feld[i] == min) {
                return "Minimum: " + min + " an Position: " + i;
            }
        }
        return "";
    }

    public int[] gemischtesFeld() {
        int[] gemischt = new int[n];
        ArrayList<Integer> list = new ArrayList<>();
        for(int i = 0; i < n; i++) {
            list.add(feld[i]);
        }
        Collections.shuffle(list);
        for(int i = 0; i < n; i++) {
            gemischt[i] = list.get(i);
        }
        return gemischt;
    }

    public void ausgeben() {
```

```
        System.out.println("Feld: " + Arrays.toString(feld));
        System.out.println("Minimum: " + minimum());
        System.out.println("Summe: " + summe());
        System.out.println("Durchschnitt: " + durchschnitt());
        System.out.println("Anzahl kleiner als 5: " + anzahlKleinerAls(5));
        System.out.println(positionMinimum());
        System.out.println("Gemischt: " + Arrays.toString(gemischtesFeld()));
    }
}
```

8)

```
class Noten {
int[] note;
int anzahl;

public Noten() {
    this.note = new int[10];
    this.anzahl = 0;
}

public void eintragen(int neueNote) {
    if(anzahl < 10) {
        if (neueNote >= 1 && neueNote <= 6) {
            note[anzahl] = neueNote;
            anzahl++;
        } else {
            System.out.println("Ungültige Note. Bitte zwischen 1 und 6 wählen.");
        }
    } else {
        System.out.println("Fehler: Es können nur maximal 10 Noten eingetragen werden.");
    }
}

public double durchschnitt() {
    int summe = 0;
    for (int i = 0; i < anzahl; i++) {
        summe += note[i];
    }
    return (double) summe / anzahl;
}

public int gibNote(int i) {
    if (i >= 0 && i < anzahl) {
        return note[i];
    } else {
        System.out.println("Fehler: Die angegebene Position ist ungültig.");
        return -1;
    }
}

public void setzeNote(int i, int neueNote) {
    if (i >= 0 && i < anzahl) {
        if (neueNote >= 1 && neueNote <= 6) {
            note[i] = neueNote;
```

```
            } else {
                System.out.println("Ungültige Note. Bitte zwischen 1 und 6 wählen.");
            }
        } else {
            System.out.println("Fehler: Die angegebene Position ist ungültig.");
        }
    }
}
```