# Übungsklausur

## Aufgabe 1 - 6 Punkte

Erläutern Sie die folgenden Begriffe
(je 1 Punkt):

1. Klasse
2. Objekt
3. Instanzvariable
4. Getter-Methode
5. Setter-Methode
6. Parameter

# Übungsklausur

## Aufgabe 1 - 6 Punkte

Erläutern Sie die folgenden Begriffe (je 1 Punkt):

1. Klasse
2. Objekt
3. Instanzvariable
4. Getter-Methode
5. Setter-Methode
6. Parameter

Klasse = Vorlage für die Bildung von Objekten, alle Instanzvariablen und Methoden werden bei der Klassendefinition festgelegt.

Objekt = Exemplar oder Instanz einer Klasse. Jedes Objekt hat die gleichen Instanzvariablen und Methoden, aber die Werte der Variablen können sich individuell unterscheiden → Status eines Objektes.

Instanzvariable = Eine Variable, die bei der Definition einer Klasse festgelegt wird und für alle Objekte der Klasse gültig ist. Die Werte dieser Variablen können sich aber von Objekt zu Objekt unterscheiden.

# Übungsklausur

## Aufgabe 1 - 6 Punkte

Erläutern Sie die folgenden Begriffe (je 1 Punkt):

1. Klasse
2. Objekt
3. Instanzvariable
4. Getter-Methode
5. Setter-Methode
6. Parameter

Getter-Methode = Methode, mit der man den Wert einer Instanzvariable unverändert zurück erhält (direkte GM) oder mit der aus Instanzvariablen ein neuer Wert berechnet wird (berechnende GM).

Setter-Methode = Methode, mit der man den Wert einer Instanzvariable verändern kann.

Parameter = Eine Variable, die beim Aufruf einer Methode dieser Methode übergeben wird. Vor allem für Setter-Methoden wichtig, damit diese wissen, auf welchen Wert die Instanzvariable gesetzt werden soll.

# Übungsklausur

## Aufgabe 2 - 8 Punkte

```
if (gewicht < ideal)
   System.out.println("Sie sind zu leicht.");
else if (gewicht > ideal)
   System.out.println("Sie sind zu schwer.");
else if (gewicht == ideal)
   System.out.println
       ("Sie haben das Idealgewicht!");
```

Beide Variablen, `gewicht` und `ideal`, sind vom Typ `double`.

Analysieren Sie diesen Quelltext und erläutern Sie, was daran korrekt ist und was man besser machen könnte.

# Übungsklausur

## Aufgabe 2 - 8 Punkte

```
if (gewicht < ideal)
   System.out.println("Sie sind zu leicht.");
else if (gewicht > ideal)
   System.out.println("Sie sind zu schwer.");
else if (gewicht == ideal)
   System.out.println
       ("Sie haben das Idealgewicht!");
```

Beide Variablen, `gewicht` und `ideal`, sind vom Typ `double`.

Analysieren Sie diesen Quelltext und erläutern Sie, was daran korrekt ist und was man besser machen könnte.

Korrekt ist, dass drei mögliche Fälle des Gewichts behandelt werden und dass das Gewicht als double-Wert verarbeitet wird.

Verbesserungen:

1. Die letzte else-Anweisung kann entfallen, wenn das Gewicht weder größer noch kleiner als das Idealgewicht ist, handelt es sich zwingend um das Idealgewicht.
2. Dann hätten wir auch nicht das Problem mit dem Vergleich gewicht == ideal, der bei double-Variablen problematisch ist wegen der Rechenungenauigkeiten.

# Übungsklausur

## Aufgabe 3 - 14 Punkte

A. Korrigieren Sie diesen fehlerhaften Quelltext (6 Punkte)

B. Erläutern Sie den Unterschied zwischen lexikalischen, syntaktischen und logischen Fehlern anhand dieses Beispiels (8 Punkte).

```
public void rateMal (integer rate)
{
   if (rate < 3)
      System.out.printeln("viel zu klein");
   else if (rate < 8)
      System.out.println("zu klein");
   else if (rate < 13)
      System.out.println("zu klein, aber fast richtig");
   else if (rate = 13)
      System.out.println("genau richtig");
   else if (rate > 13);
      System.out.println
           ("etwas zu groß, aber fast richtig");
   else if (rate > 18)
      System.out.println(zu groß);
   else if (rate > 23)
      System.out.println("viel zu groß");
}
```

# Übungsklausur

## Aufgabe 3 - 14 Punkte

```
public void rateMal (integer rate)
{
1.  if (rate < 3)
        System.out.printeln("viel zu klein");
2.  else if (rate < 8)
        System.out.println("zu klein");
3.  else if (rate < 13)
        System.out.println("zu klein, aber fast richtig");
7.  else if (rate = 13)
        System.out.println("genau richtig");
6.  else if (rate > 13);
        System.out.println
              ("etwas zu groß, aber fast richtig");
5.  else if (rate > 18)
        System.out.println(zu groß);
4.  else if (rate > 23)
        System.out.println("viel zu groß");
}
```

# Übungsklausur

## Aufgabe 4 - 20 Punkte

Es soll eine Methode geschrieben werden, die ermittelt, durch welchen Bruch Z/N (Z = Zähler, N = Nenner) die Zahl `Pi =3,14160` möglichst genau dargestellt wird.

Der Bruch soll mithilfe einer while-Schleife so gewählt werden, dass sein Wert von π = 3,14160 höchstens um 0,001 abweicht.

Die Methode beginnt mit Z = 10 und N = 3. In der Schleife werden dann je nach Bedarf Z oder N inkrementiert, solange wie die Differenz zwischen Z/N und Pi noch größer ist als 0,001.

Hier eine mögliche Konsolenausgabe der Methode

10/3 = 3,333 => zu groß
10/4 = 2,500 => zu klein
11/4 = 2,750 => zu klein
12/4 = 3,000 => zu klein
13/4 = 3,250 => zu groß
13/5 = 2,600 => zu klein
14/5 = 2,800 => zu klein
15/5 = 3,000 => zu klein
16/5 = 3,200 => zu groß
16/6 = 2,667 => zu klein

und so weiter.

# Übungsklausur

## Aufgabe 4 - 20 Punkte

Es soll eine Methode geschrieben werden, die ermittelt, durch welchen Bruch Z/N (Z = Zähler, N = Nenner) die Zahl `Pi =3 ,14160` möglichst genau dargestellt wird.

Der Bruch soll mithilfe einer while-Schleife so gewählt werden, dass sein Wert von π = 3,14160 höchstens um 0,001 abweicht.

Die Methode beginnt mit Z = 10 und N = 3. In der Schleife werden dann je nach Bedarf Z oder N inkrementiert, solange wie die Differenz zwischen Z/N und Pi noch größer ist als 0,001.

# Übungsklausur

## Aufgabe 4 - 20 Punkte

Es soll eine Methode geschrieben werden, die ermittelt, durch welchen Bruch Z/N (Z = Zähler, N = Nenner) die Zahl `Pi =3 ,14160` möglichst genau dargestellt wird.

Der Bruch soll mithilfe einer while-Schleife so gewählt werden, dass sein Wert von π = 3,14160 höchstens um 0,001 abweicht.

Die Methode beginnt mit Z = 10 und N = 3. In der Schleife werden dann je nach Bedarf Z oder N inkrementiert, solange wie die Differenz zwischen Z/N und Pi noch größer ist als 0,001.

```
public class PiBerechnung
{
    final double PI = 3.14160;
    final double EPSILON = 0.001;

    public double wert(int zaehler, int nenner)
    {
       return (double) zaehler/nenner;
    }

    public double differenz(int zaehler, int nenner)
    {
        return Math.abs(wert(zaehler, nenner) - PI);
    }

    public void findePiBruch()
    {
        int zaehler = 10;
        int nenner  = 3;

        while (differenz(zaehler, nenner) > EPSILON)
        {
            if (wert(zaehler, nenner) < PI) zaehler++;
            else nenner++;
        }

        System.out.println("Gefundener Bruch: "
            + zaehler + " / " + nenner
            + " = " + wert(zaehler, nenner));
    }
}
```

# Übungsklausur

## Aufgabe 5 - 18 Punkte

A. Schreiben Sie eine Methode, die aus einem Array von zufälligen int-Zahlen alle geraden Zahlen entfernt.
Die rechts stehenden Elemente sollen jeweils um eine Position nach links rücken. Frei gewordene Positionen am Ende des Arrays werden mit 0 überschrieben.
Sie dürfen die delete()-Methode aus ArrayTools verwenden (15 Punkte)

B. Begründen Sie außerdem, warum der veränderte Array nicht als Rückgabewert zurückgegeben werden muss. (3 Punkte)

# Übungsklausur

## Aufgabe 5 - 18 Punkte

A. Schreiben Sie eine Methode, die aus einem Array von zufälligen int-Zahlen alle geraden Zahlen entfernt.
Die rechts stehenden Elemente sollen jeweils um eine Position nach links rücken. Frei gewordene Positionen am Ende des Arrays werden mit 0 überschrieben.
Sie dürfen die delete()-Methode aus ArrayTools verwenden (15 Punkte)

B. Begründen Sie außerdem, warum der veränderte Array nicht als Rückgabewert zurückgegeben werden muss. (3 Punkte)

```
for (int i = numbers.length - 1; i >= 0; i--)
{
    if (numbers[i] % 2 == 0)
    {
        ArrayTools.delete(numbers, i);
    }
}
```

Arrays sind Referenzvariablen, speichern also die Adresse der eigentlichen Daten.

Wenn innerhalb der Methode etwas an dem Array geändert wird, werden auch die eigentlichen Daten verändert.

Eine spezielle Rückgabe des veränderten Arrays ist damit überflüssig.

# Übungsklausur

## Aufgabe 6 - 19 Punkte

A. Schreiben Sie den Quelltext für einen Bubblesort und einen Selectionsort - einschließlich Hilfsmethoden wie tausche() oder findeMiniPos(). - 10 Punkte

B. Sortieren Sie die Zahlen 12 - 8 - 15 - 4 - 6 schrittweise einmal mit Bubblesort und einmal mit Selectionsort. Beim Bubblesort reicht es, wenn Sie nur die beiden ersten Durchgänge aufschreiben (7 Punkte).

C. Begründen Sie, wieso der Selectionsort im *average-case* schneller arbeitet als der Bubblesort - 2 Punkte.

# Übungsklausur

## Aufgabe 6 - 19 Punkte

A. Schreiben Sie den Quelltext für einen Bubblesort und einen Selectionsort - einschließlich Hilfsmethoden wie tausche() oder findeMiniPos(). - 10 Punkte

B. Sortieren Sie die Zahlen 12 - 8 - 15 - 4 - 6 schrittweise einmal mit Bubblesort und einmal mit Selectionsort. Beim Bubblesort reicht es, wenn Sie nur die beiden ersten Durchgänge aufschreiben (7 Punkte).

C. Begründen Sie, wieso der Selectionsort im *average-case* schneller arbeitet als der Bubblesort - 2 Punkte.

# Übungsklausur

## Aufgabe 7 - 15 Punkte

Ein **Stack** oder **Stapelspeicher** ist eine Datenstruktur, die durch folgende Operationen beschrieben werden kann:

`public void init()`
→ erzeugt einen leeren Stack (2 Punkte).

`public boolean isEmpty()`
→ true, wenn noch kein Element enthalten ist
→ false, wenn mindestens ein Element vorhanden ist (2 Punkte).

`public void push(int neu)`
→ legt die Zahl neu oben auf den Stack (4 Punkte).

`public void pop()`
→ entfernt die zuletzt eingefügte Zahl (2 Punkte).

`public int top()`
→ liefert den Wert der zuletzt eingefügten Zahl zurück ((2 Punkte).

`Instanzvariablen (3 Punkte).`

Schreiben Sie eine Java-Klasse **Stack**, die diese Operationen mithilfe eines int-Arrays (maximal 20 Elemente) implementiert.

# Übungsklausur

## Aufgabe 7 - 15 Punkte

```
public void init()
```

```
public boolean isEmpty()
```

```
public void push(int neu)
```
→ legt die Zahl neu oben auf den Stack

```
public void pop()
```
→ entfernt die zuletzt eingefügte Zahl

```
public int top()
```
→ liefert den Wert der zuletzt eingefügten Zahl zurück.

```
public void init()
{



}

public boolean isEmpty()
{



}
```

# Übungsklausur

## Aufgabe 7 - 15 Punkte

```
public void init()
```

```
public boolean isEmpty()
```

```
public void push(int neu)
```
→ legt die Zahl neu oben auf den Stack

```
public void pop()
```
→ entfernt die zuletzt eingefügte Zahl

```
public int top()
```
→ liefert den Wert der zuletzt eingefügten Zahl zurück.

```
public void push()
{



}

public void pop()
{



}

public int top()
{



}
```

# Übungsklausur

## Aufgabe 7 - 15 Punkte

```
public void init()

public boolean isEmpty()
```

```
public void push(int neu)
```
→ legt die Zahl neu oben auf den Stack

```
public void pop()
```
→ entfernt die zuletzt eingefügte Zahl

```
public int top()
```
→ liefert den Wert der zuletzt eingefügten Zahl zurück.

```
public void init()
{
    anzahl = 0;


}

public boolean isEmpty()
{

    return anzahl == 0;

}
```

# Übungsklausur

## Aufgabe 7 - 15 Punkte

```
public void init()

public boolean isEmpty()

public void push(int neu)
```
→ legt die Zahl neu oben auf den Stack

```
public void pop()
```
→ entfernt die zuletzt eingefügte Zahl

```
public int top()
```
→ liefert den Wert der zuletzt eingefügten Zahl zurück.

```
public void push()
{

    if (anzahl < 20)
    {
        zahlen[anzahl] = neu;
        anzahl++;
    }
}

public void pop()
{
    if (!empty())
        anzahl--;
}

public int top()
{
    if (!empty())
        return zahlen[anzahl];
}
```

# Übungsklausur

## Aufgabe 8 - 15 Punkte

```
public class Gegenstand
{
   private String name;
   private int angriffswert;
   private int goldwert;
}

// Getter- und Setter-Methoden ...

public int getGoldwert()
{
   return goldwert;
}
```

```
public class Inventar
{
   Gegenstand[] rucksack = new Gegenstand[12];

   // Getter- und Setter-Methoden ...

   // Methode zum Füllen des Inventars mit
   // 12 Gegenständen

   public void sortiereNachGoldwert()
   {
      // Eure Aufgabe 15!
      // Das Inventar mit Bubblesort
      // nach Goldwert sortieren.
   }
}
```

# Übungsklausur

**Aufgabe 8**

```
public void sortiereNachGoldwert()
{









}
```

# Übungsklausur

## Aufgabe 8

```
public void sortiereNachGoldwert()
{
   for(int i=0; i<12; i++)
      for (int j=0; j<12-1-i; j++)
      {
         int gold1 = rucksack[j].getGoldwert();
         int gold2 = rucksack[j+1].getGoldwert();

         if (gold1 > gold2)
         {
            Gegenstand temp = rucksack[j];
            rucksack[j] = rucksack[j+1];
            rucksack[j+1] = temp;
         }
      }
}
```