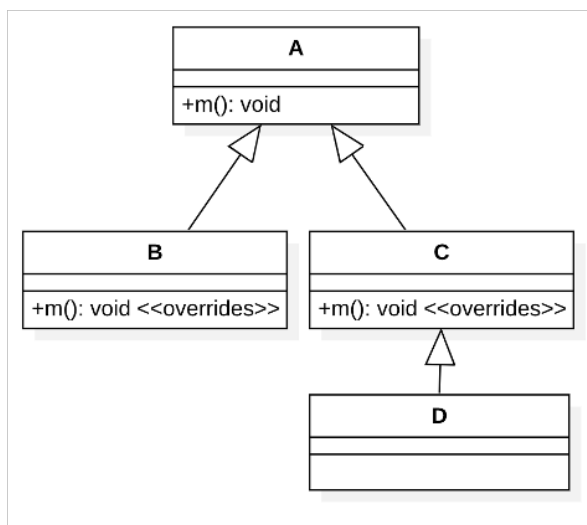


Beispielfragen und -aufgaben

Dies sind lediglich beispielhafte Fragen und Aufgaben, in der Klausur können natürlich andere Fragen und Fragetypen, sowie weitere Themen behandelt werden.

Theoriefragen

1. Was ist der Unterschied zwischen einem statischen Attribut und einem nicht-statischen Attribut?
2. Nennen Sie Beispiele aus der Java-Class-Library für generische Klassen.
3. Beschreiben Sie die Grundstruktur eines Javadoc-Kommentars.
4. Wozu dient Javadoc?
5. Was ist die Aufgabe von Unittests?
6. Beschreiben Sie die Grundidee bei der Verwendung einer der verschiedenen `assertEquals*`-Methode aus dem JUnit-Framework?
7. Was bewirkt der Aufruf von `super(arg1, arg2)` im Konstruktor?
8. Was bedeutet der Begriff Kapselung in der Objektorientierten Programmierung?
9. Was ist der Unterschied zwischen einer „Checked Exception“ und einer „Unchecked Exception.“
10. Wenn auf einem Objekt der Klasse „D“ die Methode „m()“ Aufgerufen wird, von welcher Klasse stammt dann die Implementierung der Methode „m()“.



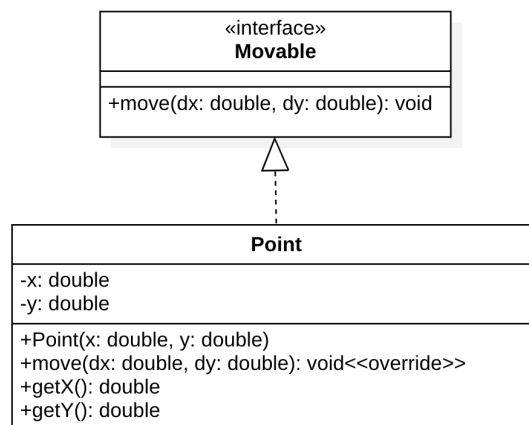
11. Erklären Sie das Grundprinzip des Beobachter-Musters.

12. a) Was ist der statische Typ in diesem Beispiel, was ist der dynamische Typ?
 b) Welche Methoden sind (ohne weiteren Type-Cast) auf der Variablen `device` aufrufbar (die öffentlichen Methoden von `ElectricalDevice` oder von `DvdPlayer`)? Begründen Sie Ihre Antwort kurz.

```
ElectricalDevice device = new DvdPlayer(20.0, 3.0);
```

Programmierung

1. Setzen Sie folgendes Interface (`Movable`) und die Klasse `Point` in Javacode um. Die Methode `move` soll die x-Koordinate um den Wert `dx` verschieben, die y-Koordinate um den Wert `dy`.



- a) Schreiben Sie eine Klasse `Point` (`public`) mit einer `x` und `y`-Koordinate als Attribute (beide `private` und Typ `double`). Erstellen Sie einen parametrisierten Konstruktor, der beide Attribute initialisiert und einfache `get`- und `set`-Methoden.
 b) Schreiben Sie Java-Code in dem drei Objekte der Klasse `Point` neu erstellt werden und den Variablen `p1`, `p2` und `p3` zugewiesen werden. Die Werte für die Argumente beim Konstruktor-Aufruf können sie selbst festlegen.
2. Implementieren Sie eine Methode zur Approximation von π , welche die folgende Formel umsetzt, um eine Annäherung von π errechnen:

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots = \frac{\pi}{4}.$$

Quelle: <https://de.wikipedia.org/wiki/Leibniz-Reihe>

Die Formel kann als Schleife umgesetzt werden. Im Computer können wir keine unendlichen Summen berechnen. Die Anzahl der berücksichtigten Brüche, bzw. wie weit die Summe läuft ($k=0$ bis $kMax$), soll durch einen Parameter (`kMax`) der Methode festgelegt werden. Die Methode soll folgende Eigenschaften haben:

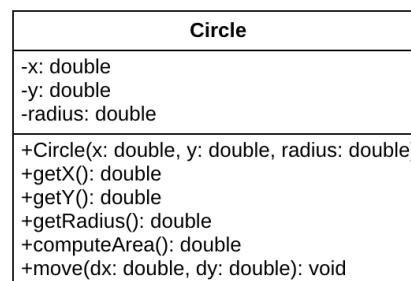
- Methodenname: `approximatePi` (`public`, `static`)

- Parameter kMax (long): legt fest, wie viele Brüche für die Gesamtsumme berechnet werden sollen
- Rückgabetyp der Methode: double, Rückgabewert: die Approximation von π

```
public class ApproxApplication {
    // your code

    public static void main(String[] args) {
        // Aufruf der Methode, 1000000 Brüche sollen berücksichtigt werden
        double result = approximatePi( 1000000L );
        System.out.println( result );
    }
}
```

3. Implementieren Sie folgende die folgende Klasse Circle in Java:



- Die Klasse soll einen parametrisierten Konstruktor haben (siehe UML-Klassendiagramm). Implementieren Sie geeignete Prüfungen im Konstruktor.
 - Die Klasse soll folgende Methoden haben (siehe UML-Klassendiagramm):
 - getX(): liefert die x-Koordinate zurück
 - getY(): liefert die y-Koordinate zurück
 - getRadius(): liefert den Radius zurück
 - computeArea(): liefert die Fläche des Kreises zurück (errechnet aus dem Radius)
 - move(dx, dy): Verschiebt den Mittelpunkt um die Werte dx und dy
4. Schreiben Sie einen Enum Color welches eine private-Attribut colorCode (String) besitzt. Das Attribut colorCode soll über die Methode getColorCode zurückgegeben werden. Der Enum soll folgende Werte mit folgenden ColorCodes (Hexadezimalcodes für RGB-Farben) beinhalten:
- RED: #FF0000
 - GREEN: #00FF00
 - BLUE: #0000FF
 - CYAN: #00FFFF
 - MAGENTA: #FF00FF
 - YELLOW: #FFFF00
5. Erstellen Sie Javacode, welcher folgendes Interface als anonyme Klasse umsetzt.

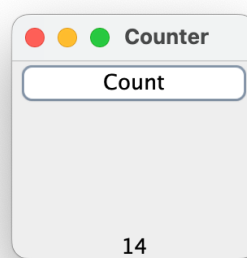
```
public interface Quadrable {  
    double square(double value);  
}
```

Die Methode square soll das Quadrat (value^2) des Parameters zurückgeben.

// Bitte Code einfügen

Quadrable q1 =

6. Erweitern Sie den Code in der Klasse ButtonFrame (siehe unten). Benutzen Sie für die Anordnung der GUI-Elemente ein BorderLayout als LayoutManager. Fügen Sie ggf. weitere benötigte GUI-Elemente hinzu. Erzeugen Sie eine JButton und fügen Sie ihn an der Stelle BorderLayout.NORTH hinzu. Erzeugen Sie ein JLabel (Element zur Anzeige von Text) mit dem Namen counterLabel und fügen es der Benutzeroberfläche an der Stelle BorderLayout.SOUTH hinzu. Jedes Mal, wenn der Button gedrückt wird, soll der counter um +1 hochgezählt werden und der aktuelle Wert im counterLabel angezeigt werden.



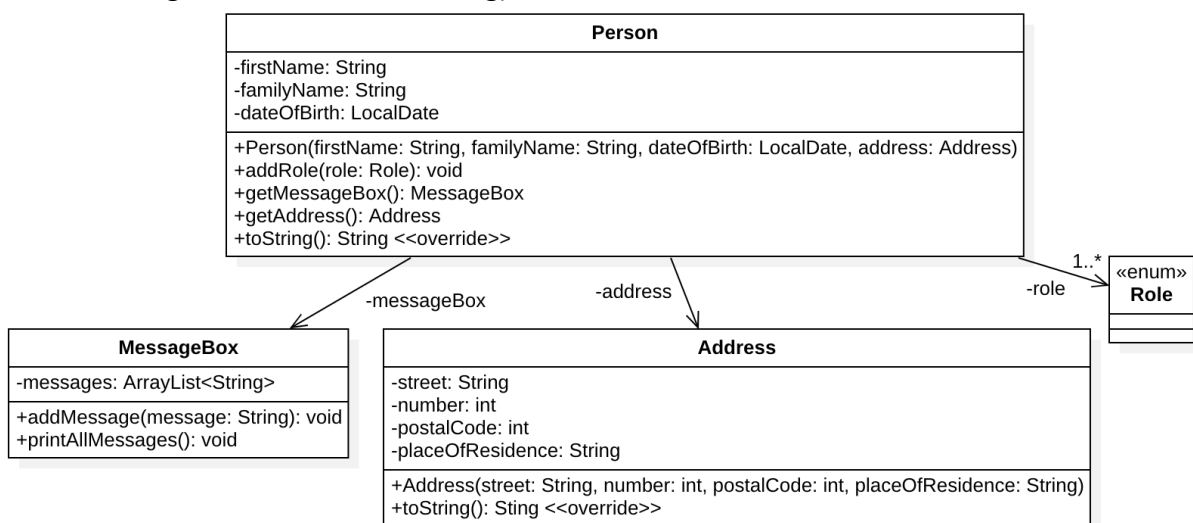
```
package de.htw.button;  
  
import javax.swing.*;  
  
public class ButtonFrame extends JFrame {  
  
    public ButtonFrame() {  
        setTitle("Counter");  
        setSize(150, 150);  
        setLocationRelativeTo(null);  
  
        setVisible(true);  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }  
  
    public static void main(String[] args) {  
        SwingUtilities.invokeLater(new Runnable() {  
            @Override  
            public void run() {  
                JFrame frame = new ButtonFrame();  
            }  
        });  
    }  
}
```

```

        frame.setVisible(true);
    }
    });
}
}

```

7. Schreiben Sie eine Methode `computeVolumeSphere` (static, Rückgabety: `double`, Sichtbarkeit: `public`), mit einem Parameter `radius` (`double`). Die Methode soll das Ergebnis nach folgender Formel berechnen und zurückgeben: $\frac{4}{3} * \pi * \text{radius}^3$
8. In dieser Aufgabe geht es darum einen kleinen Ausschnitt aus einer Hochschule zu modellieren. Es sollen folgende Klassen und Enum-Typen umgesetzt werden (siehe UML-Diagramm und Beschreibung):



Es sollen folgende Referenztypen umgesetzt werden:

- Enum: `Role`
 - `STUDENT`, `STUDENT_ASSISTANT`, `LAB_ENGINEER`, `PROFESSOR`, `LECTURER`
- Klasse: `Person`
 - Hochschulmitglied, kann verschiedene Rollen haben (min 1 Rolle)
 - hat genau eine Adresse
 - hat eine `MessageBox`, die Nachrichten an diese Person speichert
 - Überschreibt die `toString`-Methode, um eine Zusammenfassung der Personendaten als `String` zurückzugeben, Beispiele:

```

Lennart Jones, Geburtsdatum: 08.01.1990, (STUDENT)
Peter Lustig, Geburtsdatum: 12.10.1970, (LECTURER)

```

- Klasse: `Address`
 - Speichert Adressdaten
 - Überschreibt die `toString`-Methode, um eine Zusammenfassung der Adress-Daten zurückzugeben, Beispiele:

```

Pappelpromenade 205, 12345 Berlin
Schöne Allee 20, 10408 Berlin

```

- Klasse: `MessageBox` (Speichert Nachrichten für eine Person)

- MessageBox speichert eine Sammlung (z.B. vom Typ ArrayList) von Nachrichten (Strings)
- Eine neue Nachricht kann mit der addMessage-Methode hinzugefügt werden
- Alle Nachrichten einer MessageBox können mit der printAllMessages-Methode auf der Konsole ausgegeben werden, Beispiel:

Schreiben Sie eine Klasse UniApplicaton mit einer main-Methode. Legen Sie in dieser main-Methode 3 Personen-Objekte mit Beispieldaten (inkl. selbstgewählter Adressdaten und Rolle) an. Fügen Sie alle 3 Personen-Objekte einer ArrayList zu. Nutzen Sie diese Sammlung von Personenobjekten, um in einer Schleife allen Personen eine neue Nachricht in die Nachrichtenbox zu hinzuzufügen.

9. Implementieren Sie einen Algorithmus, welcher aus einer ArrayList von Strings mit Städtenamen alle paarweisen Kombinationen der Städtenamen bildet und auf der Konsole ausgibt. Ein Städtenamen soll dabei nicht mit sich selbst kombiniert werden, doppelte Kombinationen sollen nicht gebildet werden (z.B. die Kombination *Hamburg* <-> *Berlin* und *Berlin* <-> *Hamburg* soll nur einmal ausgegeben werden).

```
public static void main(String[] args) {

    ArrayList<String> cities = new ArrayList<>();
    cities.add("Hamburg");
    cities.add("Berlin");
    cities.add("Leipzig");
    cities.add("Stuttgart");

    // Bitte Code hinzufügen

}
```

Ausgabe:

```
Hamburg <-> Berlin
Hamburg <-> Leipzig
Hamburg <-> Stuttgart
Berlin <-> Leipzig
Berlin <-> Stuttgart
Leipzig <-> Stuttgart
```

10. Schreiben Sie eine Methode:

```
public boolean checkTicTacToe(char[][] field, char symbol)
```

welche überprüft, ob in einem zweidimensionalen Array von Charactern (char) in einer Zeile, Spalte oder Diagonalen die gleichen Zeichen vorkommen. Die Methode soll folgende Eigenschaften haben:

- Parameter field (char[][]): Spielfeld mit Zeichen
- Parameter symbol (char): Zeichen, dass überprüft werden soll

- Rückgabebetyp der Methode: boolean, Rückgabewert: true, wenn in mindestens einer Zeile, Spalte oder Diagonalen das Zeichen in jeder Array-Zelle vorkommt, ansonsten false

```
public class TicTacToeApplication {

    // your code

    public static void main(String[] args) {

        char[][] field1 = {{'x', 'x', 'x'},
                           {'x', 'o', 'o'},
                           {'o', 'o', 'x'}};

        char[][] field2 = {{'o', 'x', 'x'},
                           {'x', 'o', 'o'},
                           {'x', 'x', 'o'}};

        System.out.println( checkTicTacToe(field1, 'x') );
        System.out.println( checkTicTacToe(field1, 'o') );
        System.out.println( checkTicTacToe(field2, 'x') );
        System.out.println( checkTicTacToe(field2, 'o') );
    }
}
```

Ausgabe:

```
true
false
false
true
```


Welche Prüfungen wären sinnvoll in die checkTicTacToe-Methode einzubauen?

Code-Verständnis

1. Welchen Wert liefert folgender Ausdruck?

```
boolean a = true;
boolean b = true;
boolean c = false;
```

```
(a && b) || false
```



2. Warum kompiliert folgender Code nicht?

```
int x = 1;

int x = 20;
```

3. Warum kompiliert folgender Code nicht?

```
{
    int x = 5;
```

```
}  
x = 10;
```

4. Was wird folgender Code auf der Kommandozeile ausgeben? Begründen Sie Ihre Aussage.

```
Person p1 = new Person("Lennart", "Smith");  
Person p2 = p1;  
  
p2.setFirstname("Sheldon");  
System.out.println( p1.getFirstname() );  
System.out.println( p2.getFirstname() );
```

5. Was wird folgender Code auf der Kommandozeile ausgeben? Begründen Sie Ihre Aussage:

```
int[] a = {1, 5, 3};  
int[] b = a;  
  
a[1] = 20;  
  
System.out.println( a[1] );  
System.out.println( b[1] );
```

6. Was ist der Unterschied zwischen beiden Ausdrücken? Was geben die beiden Ausdrücke jeweils zurück?

```
4 / 3  
4.0 / 3.0
```

11. Was ist der Unterschied zwischen beiden Ausdrücken?

```
i == 20  
  
i = 20
```