Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

38

Programmieren 2

Test 1, Frühlingssemester 2012

R. Diehl, M. Klaper, R. Meier; Version 6.24

Name: Rohrer Kurs: 3

Vorname: Felix

Rahmenbedingungen:

1. **Prüfungszeit: 70 Minuten**
2. In diesem Test können 70 Punkte erreicht werden. Jeder Aufgabe ist eine maximal erreichbare Punktezahl zugeordnet.
3. Schreiben Sie Ihren Namen und Ihren Kurs auf dieses Blatt. Blätter ohne Namensangabe werden nicht bewertet.
4. Es handelt sich um einen schriftlichen Test ohne Einsatz des Computers oder elektronischer Hilfsmittel. Sie dürfen keine Unterlagen verwenden.
5. Sollte die Problemstellung Unklarheiten aufweisen, dürfen Sie eigene Annahmen treffen. Führen Sie diese in der Lösung auf.
6. Schreiben Sie möglichst verständlich und gut leserlich. Missverständliche Lösungen werden nicht berücksichtigt.
7. Benutzen Sie den Freiraum unter den Aufgaben für Ihre Lösung.

Für die Korrektur (nicht ausfüllen!)

1	2	3	4	5	6	7	8	Punkte
6	8	5	6	5	6	7	3	65
9	**10**	**11**	**12**	**13**				
4	5	5	4	1				

Aufgabe 1: Abstrakte Klassen und Methoden in Java (6 Punkte)

Nachfolgende Fragen beziehen sich auf das hier gegebene Interface `ShapeInterface` und die Klasse `ShapeBase`.

```
public interface ShapeInterface
{
    void setOffset(int newOffset);
    int getOffset();
    void drawAt(int lineNumber);
    void drawHere();
}
```

```
public abstract class ShapeBase implements ShapeInterface
{
    private int offset;

    public ShapeBase( )
    {
        offset = 0;
    }

    public ShapeBase(int theOffset)
    {
        offset = theOffset;
    }

    public abstract void drawHere( );

    public void drawAt(int lineNumber)
    {
        for (int count = 0; count < lineNumber; count++)
            System.out.println( );
            drawHere( );
    }

    public void setOffset(int newOffset)
    {
        offset = newOffset;
    }

    public int getOffset( )
    {
        return offset;
    }
}
```

Hinweis: Kreuzen Sie alle Aussagen an, die richtig sind. Pro richtiges Kreuz erhalten Sie 1 Punkt. Für ein falsches Kreuz erhalten Sie -1 Punkt. Bei dieser Aufgabe können Sie nicht weniger als 0 Punkte erhalten. Es gibt insgesamt 6 verschiedene richtige Kreuze.

a) Das Interface ShapeInterface …

- [x] definiert einen Typ.
- [x] dient als Spezifikation (WAS).
- [ ] kann instantiiert werden.
- [x] könnte Klassenkonstanten beinhalten.
- [ ] muss neben abstrakten auch konkrete Methoden beinhalten.

b) Die Klasse ShapeBase …

- [ ] darf keine Attribute haben.
- [x] definiert einen Typ.
- [ ] kann instantiiert werden.
- [x] könnte auch mehr als ein Interface implementieren.
- [ ] muss neben abstrakten auch konkrete Methoden beinhalten.

c) Ist folgende Zeile korrekt? `public class Hexagon extends ShapeBase {…}`

- [x] Ja
- [ ] Nein

Aufgabe 2: Vererbung in Java (8 Punkte)

Nachfolgende Fragen beziehen sich auf die hier gegebenen Klassen `Person` und `Student`.

```
public class Student extends Person
{
    private int studentNumber;

    public Student( )
    {
        super( );
        studentNumber = 0;  //Indicating no number yet
    }

    public Student(String initialName, int initialStudentNumber)
    {
        super(initialName);
        studentNumber = initialStudentNumber;
    }

    public void reset(String newName, int newStudentNumber)
    {
        setName(newName);
        studentNumber = newStudentNumber;
    }

    public int getStudentNumber( )
    {
        return studentNumber;
    }

    public void setStudentNumber(int newStudentNumber)
    {
        studentNumber = newStudentNumber;
    }

    public void writeOutput( )
    {
        System.out.println("Name: " + getName( ));
        System.out.println("Student Number: " + studentNumber);
    }

    public String toString( )
    {
        return "Name: " + getName( ) + "\nStudent number: "  +
                                                    studentNumber;
    }
}
```

```
public class Person
{
    private String name;

    public Person( )
    {
        name = "No name yet";
    }
    public Person(String initialName)
    {
        name = initialName;
    }
    public void setName(String newName)
    {
        name = newName;
    }
    public String getName( )
    {
        return name;
    }
    public void writeOutput( )
    {
        System.out.println("Name: " + name);
    }
    public boolean hasSameName(Person otherPerson)
    {
        return name.equalsIgnoreCase(otherPerson.name);
    }
}
```

Hinweis: Kreuzen Sie alle Aussagen an, die richtig sind. Pro richtiges Kreuz erhalten Sie 1 Punkt. Für ein falsches Kreuz erhalten Sie -1 Punkt. Bei dieser Aufgabe können Sie nicht weniger als 0 Punkte erhalten. Es gibt insgesamt 7 verschiedene richtige Kreuze.

a) Die Superklasse ist (1 Pkt.)
- ☐ `equals`
- ☐ `name`
- ☒ `Person`
- ☐ `String`
- ☐ `Student`

b) Wie nennt man die Konstruktion der Methode `writeOutput` in der Klasse `Student` (2 Pkte.)
- ☐ Overloading
- ☒ Overriding
- ☐ Substitution
- ☐ Überladen
- ☒ Überschreiben
- ☐ Casting

c) Hat ein Objekt der Klasse `Student` eine Methode "`getName`"? (1 Pkt.)

- ☐ Ist in bestimmten Fällen möglich, aber nicht generell.
- ☒ Ja, immer
- ☐ Kann mit den vorliegenden Angaben noch nicht beurteilt werden.
- ☐ Nein, nie

d) Besitzt ein Objekt der Klasse `Student` eine Instanzvariable mit Namen "`name`"? (1 Pkt.)

- ☐ Nein, nie
- ☐ Kann mit den vorliegenden Angaben noch nicht beurteilt werden.
- ☒ Ja, immer
- ☐ Ist in bestimmten Fällen möglich, aber nicht generell.

e) Kann die Klasse `Student` direkt auf das Attribut `name` der Klasse `Person` zugreifen? (1 Pkt.)

- ☐ Ist in bestimmten Fällen möglich, aber nicht generell.
- ☐ Ja, immer
- ☐ Kann mit den vorliegenden Angaben noch nicht beurteilt werden.
- ☒ Nein, nie

f) Wie lautet die Ausgabe in folgendem Beispiel? (1 Pkt.)

```
Student s = new Student("Anna Muster", 678);
Person  p = new Person("Peter Muster");
s.writeOutput();
```

☐	☐	☒
Name: Anna Muster Student Number: 678 Name: Peter Muster Student Number: 339	Name: Anna Muster Student Number: 678 Name: Peter 678	Name: Anna Muster Student Number: 678

g) Wie lautet die Ausgabe in folgendem Beispiel ganz genau? (1 Pkt.)

```
Student st = new Student("Sepp Meier", 2011);
System.out.println(st.toString());
```

Name: Sepp Meier
Student number: 2011

8

Aufgabe 3: Polymorphie (5 Punkte)

Gegeben ist folgende Klasse:

```
public class Student
{
        private String name;
        private int ects;

        public Student(String name, int ects) {
                this.name = name;
                this.ects = ects;
        }

        public String toString() {
            return (name + " " + ects);
        }
        // other methods omitted
}
```

a) Ergänzen Sie in der folgenden Klasse den Konstruktor und die Methode `toString()`. Der Konstruktor soll *alle* Instanzvariablen korrekt initialisieren und die Methode `toString()` soll *alle* Instanzvariablen ausgeben. (3 Punkte)

```
public class MasterStudent extends Student
{
    private String advisor; // persönlicher Berater des Master Studenten

    public MasterStudent(String name, int ects, String advisor) {
        super(name, ects);
        this.advisor = advisor;


    }

    public String toString() {
        return (super.toString() + " " + advisor);
    }
}
```

1

2

b) Ergänzen Sie die folgenden Sätze:

Die Methode `toString()` in der Klasse `MasterStudent` ...überschreibt..... die Methode `toString()` der Klasse `Student`. (0.5 Punkte)

Es handelt sich dabei um ...Hacke..... Polymorphie. (0.5 Punkte)

1

c) In welchen Klassen kann die Methode `toString()` verwendet werden? (1 Punkt)

Bei allen, Object -> Student -> MasterStudent

1

Aufgabe 4: Interfaces (6 Punkte)

Gegeben ist die folgende Klasse:

```
public class Rectangle  implements Shape ✓
{
    private int height;
    private int width;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    public Rectangle()
    {
        height = 40;
        width = 30;
        xPosition = 45;
        yPosition = 25;
        color = "red";
        isVisible = false;
    }
    public void makeVisible()
    {
        isVisible = true;
        draw();
    }
    public void makeInvisible()
    {
        isVisible = false;
    }
    public void moveHorizontal(int distance)
    {
        xPosition += distance;
        draw();
    }
    public void moveVertical(int distance)
    {
        yPosition += distance;
        draw();
    }
    public void changeColor(String newColor)
    {
        color = newColor;
        draw();
    }
    private void draw()
    { …
    }
}
```

a) Schreiben Sie ein Interface `Shape`, das *alle* Methoden ausser der Methode `changeColor()` der Klasse `Rectangle` deklariert. (4 Punkte)

```
public interface Shape ✓
{
    void makeVisible();  ✓
    void makeInvisible();  ✓
    void moveHorizontal(int distance);  ✓
    void moveVertical(int distance);  ✓
}
```

5

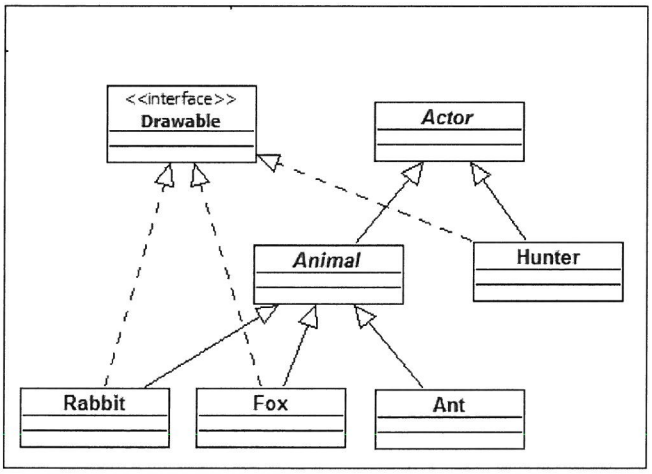
b) Ändern Sie die Klasse `Rectangle` direkt im Code so ab, dass diese das Interface `Shape` implementiert. (1 Punkt)

```
=> implements Shape ✓
```

1

Aufgabe 5: Vererbung in Java II (5 Punkte)

Gegeben ist folgendes Klassendiagramm:



Geben Sie für jede der oben stehenden Klassen und Schnittstellen (Interfaces) nur den Klassenkopf an, d.h. nur die Definition einer Klasse *ohne* Felder, Konstruktoren und Methoden. (5 Punkte)

Der Klassenkopf der Klasse `Actor` lautet: `public class Actor {...}`.

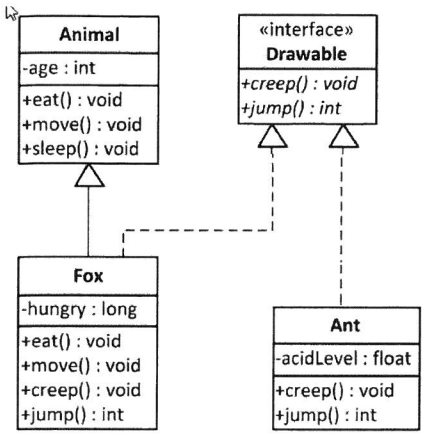
abstract

public interface Drawable {...}
public abstract class Actor {...}
public abstract class Animal extends Actor {...}
public class Hunter extends Actor implements Drawable {...}
public class Rabbit extends Animal implements Drawable {...}
public class Fox extends Animal implements Drawable {...}
public class Ant extends Animal {...}

5

Aufgabe 6: Typhierarchie (6 Punkte)

Gegeben ist folgendes Klassendiagramm:



Gegeben sind ausserdem die folgenden Deklarationen:

```
Ant  a;
Fox f;
Drawable d;
```

Entscheiden Sie für die folgenden Anweisungen, ob sie *compilieren* (je 0.5 Punkte):

`a = new Ant();`	☒ compiliert	☐ compiliert nicht
`d = new Drawable();`	☐ compiliert	☒ compiliert nicht
`f = new Animal();`	☐ compiliert	☒ compiliert nicht
`d = new Fox();`	☒ compiliert	☐ compiliert nicht

2 ✓

Gegeben sind folgende gültige Zuweisungen:

```
Animal a1 = new Animal();
Animal a2 = new Fox();
Drawable d1 = new Ant();
```

Ergänzen Sie die folgende Tabelle: Bestimmen Sie für die folgenden Methodenaufrufe jeweils den statischen und den dynamischen Typ der Variable und geben Sie an, welche Implementation (d.h. von welcher Klasse) ausgeführt wird (pro vollständig korrekter Zeile je 1 Punkt)

	statischer Typ	dynamischer Typ	Methodenimplementation der Klasse …
`d1.jump()`	Drawable	Ant	Ant ✓
`a2.sleep()`	Animal	Fox	Animal ✓
`a2.eat()`	Animal	Fox	Fox ✓
`a1.eat()`	Animal	Animal	Animal ✓

4 ✓

Aufgabe 7: Implemention und Verwendung einer Queue (8 Punkte)

Implementieren Sie in der gegebenen Klasse `MyQueue` die Methoden `enqueue()` und `dequeue()`. Beide Methoden sollen nicht prüfen, ob die Queue voll oder leer ist. Der Zugriff auf das Array muss als Ringliste implementiert werden. (8 Punkte)

```
public class MyQueue
{
    private int size;           // Anzahl Plaetze der Queue
    private int n = 0;          // Anzahl Elemente in der Queue
    private int in = 0;         // Zeiger auf leeren Platz
    private int out = 0;        // Zeiger auf belegten Platz
    private Object[] queue;     // Array als Ringliste verwenden

    public MyQueue(int s)
    {
        size = s;
        queue = new Object[size];
    }

    public void enqueue(Object o)
    {
       n++;
       if(in == size){          // in = in % size;
          in=0;
       }
       queue[in] = o;
       in++;
    }

    public Object dequeue()
    {
       Object o;
       n--;
       if(out == size){
          out=0;
       }
       o = queue[out];          // queue[out] = null;
       out++;                   // // Garbage Collector gibt speicher frei

       return o;
    }

    public boolean isEmpty() { return (n == 0); }

    public boolean isFull()  { return (n == size); }
}
```

3.5

3.5

Aufgabe 8: `equals()`-Methode (3 Punkte)

Kreuzen Sie die richtigen Aussagen an. Hinweis: Jedes richtige Kreuz ergibt 0.5 Punkte, jedes falsche -0.5 Punkte. Weniger als null Punkte gibt es aber nicht!

Ja	Nein	
☒	☐	die `equals()`-Methode muss reflexiv implementiert sein.
☐	☒	der Vergleich `x.equals(null)` provoziert eine Null Pointer Exception
☐	☒	jede Klasse muss die `equals()`-Methode implementieren
☐	☒	wenn die Methode `hashCode()` für zwei Objekte den identischen Wert liefert, muss `equals()` den Wert `true` liefern.
☒	☐	die `equals()`-Methode muss symmetrisch implementiert sein.
☐	☒	es macht Sinn die `equals()`-Methode zu überladen (overload)

3

Aufgabe 9: Korrekte Implementation von `equals()` (4 Punkte)

a) Bringen Sie die folgenden Aufgaben in die richtig Reihenfolge, damit sie einer korrekten Implementation einer `equals()`-Methode entsprechen (2 Punkte):

Reihenfolge (Zahl eintragen)	Aufgabe
2	Test auf `null`
4	Vergleich aller relevanter Felder
3	Test auf Typen-Vergleichbarkeit
1	Test auf Identität

2

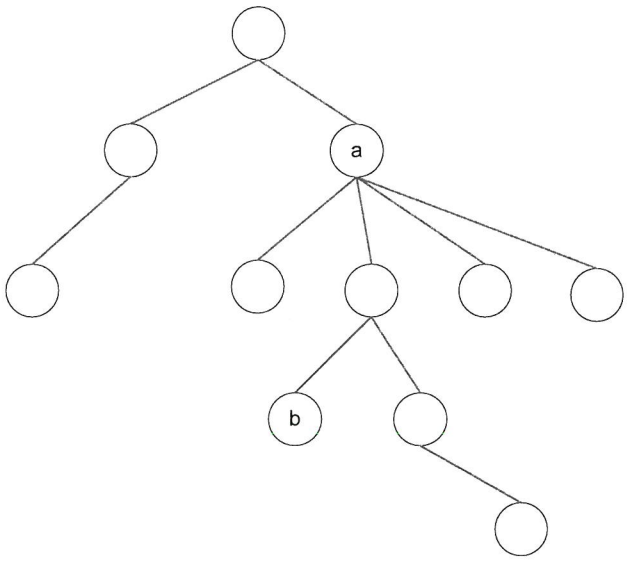
b) Zeigen Sie anhand eines (einzeiligen) Codebeispieles mit dem Aufruf der `equals()`-Methode auf ein Objekt `demo`, was der Begriff der **Reflexivität** in Bezug auf den `equals`-Contract bedeutet (2 Punkte):

demo.equals(demo) → true

2

Aufgabe 10: Bäume - Definitionen (5 Punkte)

Gegeben ist folgender Baum:



a) Bestimmen Sie den Grad des Knoten a. (1 Punkt)

4 ✓

b) Welche Tiefe hat der Knoten mit dem Wert b? (1 Punkt)

4

c) Bestimmen Sie die Ordnung des Baumes. (1 Punkt)

>= 4 ✓

d) Welche Höhe hat der Baum? (1 Punkt)

5 ✓

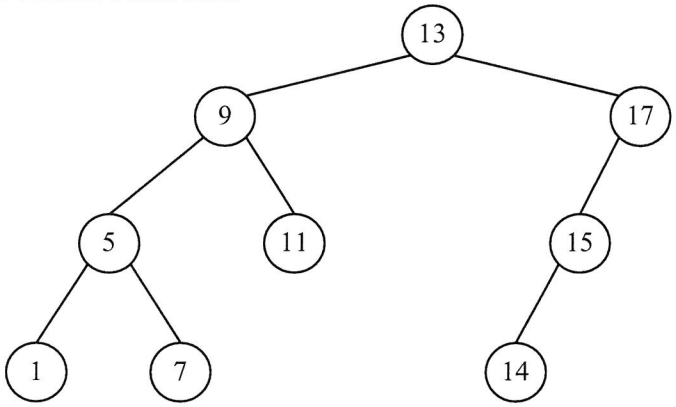
e) Ist dieser Baum ausgefüllt? (1 Punkt)

☐ Ja
☒ Nein ✓

5

Aufgabe 11: Binäre Suchbäume (5 Punkte)

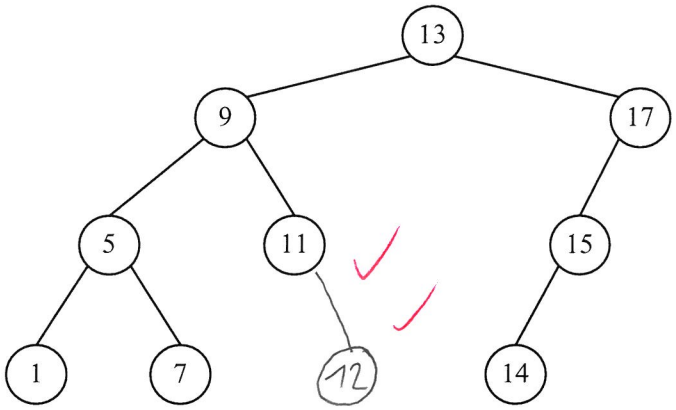
Gegeben ist folgender binärer Suchbaum:



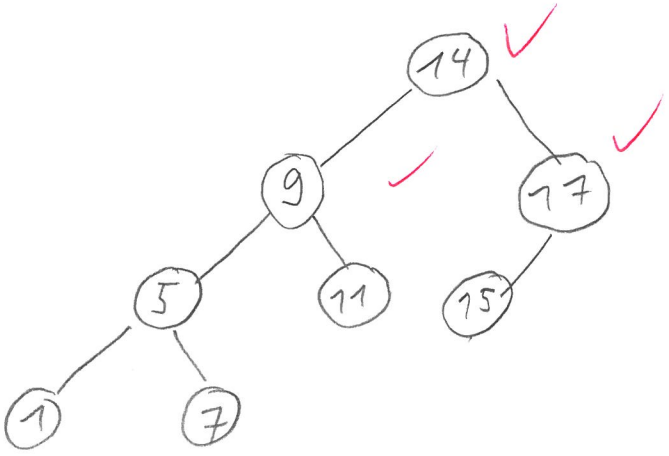
a) Geben Sie die Ausgabe des Baumes in der ***Postorder*** Durchlaufordnung an. (1 Punkt)

1, 7, 5, 11, 9, 14, 15, 17, 13

b) Fügen Sie in den binären Suchbaum einen neuen Knoten mit dem Schlüsselwert 12 ein. (2 Punkte)

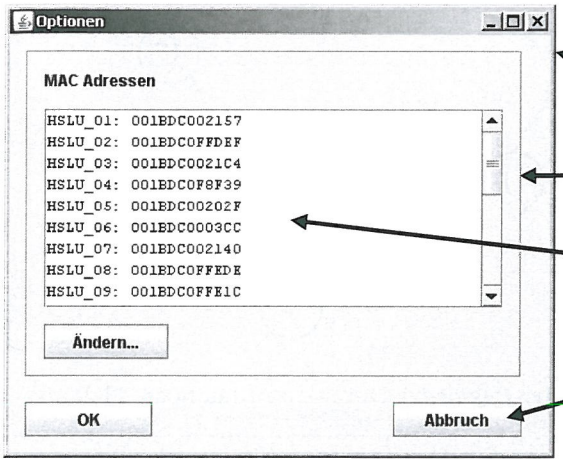


c) Zeichnen Sie den binären Suchbaum **aus Teilaufgabe a)**, nachdem der Knoten mit dem Schlüsselwert 13 entfernt worden ist. (2 Punkte)



Aufgabe 12: GUI Komponenten (4 Punkte)

Benennen Sie die im Bild mit Pfeilen markierten GUI Komponenten (AWT) und geben Sie zu jeder Komponente an, ob es sich dabei um eine elementare Komponente oder einen Container handelt:



Bezeichnung	
Frame	☐ Elementar ☒ Container
Panel	☐ Elementar ☒ Container
List oder TextArea	☒ Elementar ☐ Container
Button	☒ Elementar ☐ Container

✓

(4)

Aufgabe 13: Ereignissteuerung (5 Punkte)

a) Wie heissen die zwei grundlegenden Arten von Objekten beim Event-Handling in Java? (1 Punkt)

sequentiell, Ereignis gesteuert

b) Beschreiben Sie die zwei grundlegenden Hauptkonzepte die das Event-Handling in Java ermöglichen.
(3 Punkte)

sequentiell: fest Ablauf, input nur bei Aufforderung möglich

Ereignis gesteuert: "dynamischen Ablauf" input jederzeit möglich
z.B. drücken eines Button

c) Kann ein spezifischer Event in Java an mehrere Abonnenten gesendet werden? (1 Punkt)

☒ Ja
☐ Nein

a) Event Quelle / Source Objekte
Event Verarbeiter / Listener Objekte

b) • Listener registriert sich mit der Source um Events zu empfangen

• Source sendet Events zum registrierte Listener und ruft die ActionPerformed() auf

Inhaltsverzeichnis:

Aufgabe 1: Abstrakte Klassen und Methoden in Java (6 Punkte) 2
Aufgabe 2: Vererbung in Java (8 Punkte) 4
Aufgabe 3: Polymorphie (5 Punkte) 7
Aufgabe 4: Interfaces (6 Punkte) 8
Aufgabe 5: Vererbung in Java II (5 Punkte) 10
Aufgabe 6: Typhierarchie (6 Punkte) 11
Aufgabe 7: Implemention und Verwendung einer Queue (8 Punkte) 12
Aufgabe 8: `equals()`-Methode (3 Punkte) 13
Aufgabe 9: Korrekte Implementation von `equals()` (4 Punkte) 13
Aufgabe 10: Bäume - Definitionen (5 Punkte) 14
Aufgabe 11: Binäre Suchbäume (5 Punkte) 15
Aufgabe 12: GUI Komponenten (4 Punkte) 16
Aufgabe 13: Ereignissteuerung (5 Punkte) 17

---- Ende Test 1 ----