**Software-Architektur und Testen**

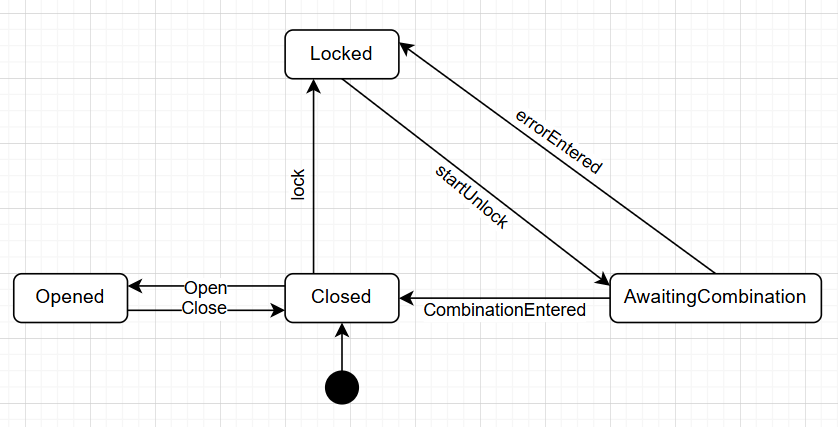
**90min**

**90 Punkte**

**80p 🡪 1.0**

**Aufgabe 1 (15p)**

Zustandsautomaten Garage



Entwerfen Sie Testfälle für die Zustandsautomaten indem Sie:

1. Übergangsbaum daraus ableiten
2. Aus dem Übergangsbaum Testfälle nach der N+-Strategie („all round trip path“) herleiten.

**Aufgabe 2 (15p)**

Neues Anmeldungssystem für Tagungen

Methoden:

Tagung(String titel, int kapazitaet)

void anmelden(Teilnehemer t)

// Teilnehmer meldet sich zu Tagung an

void abmelden(Teilnehmer t)

boolean istAngemeldet(Teilnehmer t)

Spezifikation:

* Begrenzung der Tagung durch die „kapazitaet“
* Mehrfache Anmeldung nicht möglich
* Tagungsname ist eindeutig

Implementieren Sie die Klasse Tagung gemäß Design-by-Contract. Führen Sie notwendige Intsanzen und Klassenvariablen ein. Geben Sie zusätzliche Methoden, precondition und postcondition an.

**Aufgabe 3 (25p)**

Mehrstufige Kundensupportsystem. Anfragen durchlaufen mehrere Supportebenen. Jede Ebene prüft, ob sie die Anfrage bearbeiten kann, falls nicht an die nächste Instanz weitergeben. Implementieren Sie das ChainOfResponsibility-Pattern anhand folgender Anforderungen:

* SupportRequest beinhaltet ein *SupportType* („technisch“, „kaufmännisch“ oder „allgemein“) und ein *Inhalt*.
* 3 Supportklassen:
  + Technicksupport 🡪 „technisch“
  + KaufmännischSupport 🡪 „kaufmännisch“
  + AllgemeinerSupport 🡪 „Alles andere“

1. UML-Diagramm für obige Klassen gemäß ChainOfResponsibility
2. Pseudo-Code für SupportRequest Methode für TechnickSupport
3. Erklären Sie an diesem Beispiel Vor-/Nachteile des ChainOfResponsibility

**Aufgabe 4 (15p)**

public class Rectangle {

    protected int width, height;

    // getter und setter

    public int area() {

        return width \* height;

    }

}

public class Square extends Rectangle {

    public void setWidth(int w) {

        width = w;

        height = w;

    }

    public void setHeight(int h) {

        width = h;

        height = h;

    }

}

@Test

public void areaTest(Rectangle r) {

    int width = 4;

    int height = 5;

    r.setWidth(width);

    r.setHeight(height);

    int area = r.area();

    assertEquals(width \* height, area);

}

1. Welches Prinzip ist hier verletzt? Begründen Sie Ihre Antwort und erklären Sie das Prinzip an diesem Beispiel.
2. Ändern Sie das Beispiel so um, dass das Prinzip eingehalten ist.

**Aufgabe 5 (20p)**

public enum ReportingType {

    SCV, XML,

}

public class ReportingService {

    public void generateReportBaseOnType(ReportingType){

        if("CSV".equalsIgnoreCase(ReportingType)){

            generateCSVReport();

        }else if("XML".equalsIgnoreCase(ReportingType)){

            generateXMLReport();

        }

    }

    public void generateCSVReport(){...}

    public void generateXMLReport(){...}

}

1. Erklären Sie wieso das Beispiel das Open-Closed-Prinzip verletzt.
2. Führen sie alle Codeänderungen durch, sodass Open-Closed-Prinzip eingehalten wird.