## Design Entscheidungen:

### Algorithmen:

Für dem Fleury-Algorithmus wurde unsere eigene Graphen Klasse um einen neuen Konstruktor erweitert der es erlaubt einen Graphen zu kopieren. Der Grund für diese Entscheidung war den Code im Fleury zu verringern da dort vorher der Graph in einen neuen Graphen kopiert wurde.

Dieser Graph wird genutzt um Brückenkanten zu bestimmen indem ihm die Kante genommen wird und dann mit einem Wegsuch-Algorithmus getestet wird ob es noch einen Weg gibt.

### Tools:

Der GraphGenerator wurde um eine Methode zum Erzeugen von Eulergraphen erweitert.

Durch eine äußerst allgemeine Konstruktion ist es uns möglich weiterhin die Freiheit über Knoten und Kantengrad zu entscheiden dem Benutzer zu überlassen, und trotzdem alle Eigenschaften eines Eulergraphen zu garantieren.

## 2. Textantworten:

### 2.1.(a) :

Eine gegebene Kantenfolge ist genau dann ein Eulerkreis, wenn sie alle Kanten des Ausgangsgraphen beinhält, und der Startknoten identisch zum Endknoten ist.

### 2.1.(b):

Bei der Erzeugung eines Eulergraphen gehen wir in 4 Schritten vor:

1. Schritt: Die Anzahl der gewünschten Knoten wird dem Graphen hinzugefügt.

Für den 2ten & 3ten Schritt gelten folgende Bedingungen:

- Sofern mehr als 2 Knoten einen ungeraden Grad besitzen so muss mindestens einer von ihnen im nächsten Schritt verbunden werden.

- Es werden solange Knoten verbunden bis nur noch 2 Knoten zu erzeugen sind.

1. Schritt: Solange nicht alle Knoten miteinander verbunden sind verbinden wir immer einen Knoten aus der Menge der verbundenen mit einem zufälligen Knoten.
2. Schritt: Sobald alle Knoten miteinander verbunden wurden können wir frei 2 Knoten aus der Menge wählen und mit einer neuen Kante verbinden.
3. Wir hören mit dem zufälligen Einfügen auf, wenn nur noch 2 Kanten zu vergeben sind und lösen die verbleibenden Knoten mit ungeradem Grad nach einem von uns entwickelten Verfahren auf.

In diesem Verfahren müssen wir alle 3 möglichen Fälle abdecken welche wären, es existieren:

0 Knoten mit ungeradem Grad: Wir fügen 2 Schleifen an einem zufälligen Knoten an

2 Knoten mit ungeradem Grad: Wir verbinden die 2 Knoten und fügen eine Schleife an einen beliebigen Knoten an

4 Knoten mit ungeradem Grad: Wir verbinden jeweils 2 der Knoten miteinander

### 3. Auswirkung des Test-First Verfahren:

Tests vor dem Implementieren zu schreiben bringt einige Vorteile mit sich. Beim Schreiben von Test muss man sich bereits klar machen was das Ergebnis eines Algorithmus bzw. eines Programms sein soll bzw. wie es aussehen soll.

Wenn gegen Tests Implementiert wird kann der Programmierer zielstrebiger vorgehen was allerdings bei ungenügenden Test eher von Nachteil sein kann.

Beim Implementieren der Eulerkreis-Algorithmen hat es geholfen gegen einen Test zu implementieren. Da vorher klar war was das Ergebnis sein sollte wurden einige Fehler schneller gefunden und andere schon vorher Ausgeschlossen.

Schwierigkeiten sind bei uns keine aufgetreten. Allerdings musste man sich vorher mit der Theorie mehr beschäftigen was nicht unbedingt eine Schwierigkeit ist.

# Beweis zu 2.1.(b)

Kungerade = Die Menge aller Knoten im Graphen die einen ungeraden Grad besitzen.

Kgerade = Die Menge aller Knoten im Graphen die einen geraden Grad besitzen.

G =Graph

Get(e) = Das Ziel der Kante e im Graphen G

Ges(e) = Die Quelle der Kante e im Graphen G

Trivialerweise unterschlagen wir in den Fällen die Erzeugung von Schleifen, da diese keine Auswirkung auf die Geradheit des Knotengrades haben.

## Fall 1: |Kungerade| = 0

Dann: Die neue Kante e hat Get(e) != Ges(e) mit Get(e) ∈ Kgerade ^ Ges(e) ∈ Kgerade

Dann: |Kungerade| = 2

## Fall 2: |Kungerade| = 2

Fall 1: Die neue Kante e hat Get(e) != Ges(e) mit Get(e) ∈ Kgerade ^ Ges(e) ∈ Kgerade

Dann: |Kungerade| = 4

Fall 2: Die neue Kante e hat Get(e) != Ges(e) mit Get(e) ∈ Kungerade ^ Ges(e) ∈ Kgerade

Dann: |Kungerade| = 2

Fall 3: Die neue Kante e hat Get(e) != Ges(e) mit Get(e) ∈ Kungerade ^ Ges(e) ∈ Kungerade

Dann: |Kungerade| = 0

## Fall 3: |Kungerade| = 4

Fall 1: Die neue Kante e hat Get(e) != Ges(e) mit Get(e) ∈ Kungerade ^ Ges(e) ∈ Kgerade

Dann: |Kungerade| = 4

Fall 2: Die neue Kante e hat Get(e) != Ges(e) mit Get(e) ∈ Kungerade ^ Ges(e) ∈ Kungerade

Dann: |Kungerade| = 2

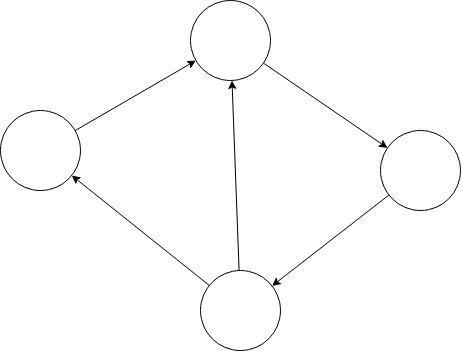
Fall 3: Die neue Kante e hat Get(e) != Ges(e) mit Get(e) ∈ Kgerade ^ Ges(e) ∈ Kgerade kann nicht auftreten, da wir verlangen das bei |Kungerade| > 2 mindestens ein Knoten k ∈ Kungerade verbunden wird.

## 3. Theorieteil:

### Aufgabe X:

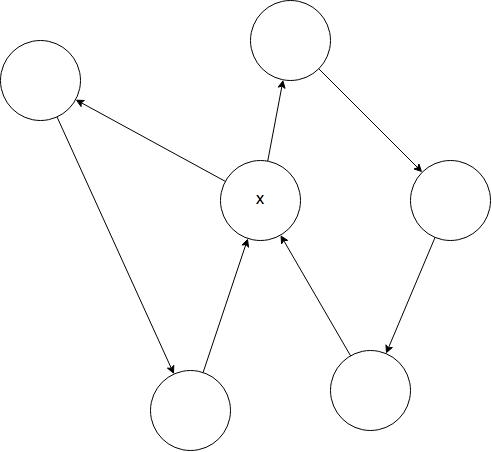
1. Wir können über den äußeren Kreis alle Knoten besuchen -> Hamiltonkreis

Da wir aber 2 Knoten mit ungeradem Grad besitzen, kann der Graph nur einen Eulerpfad beinhalten.



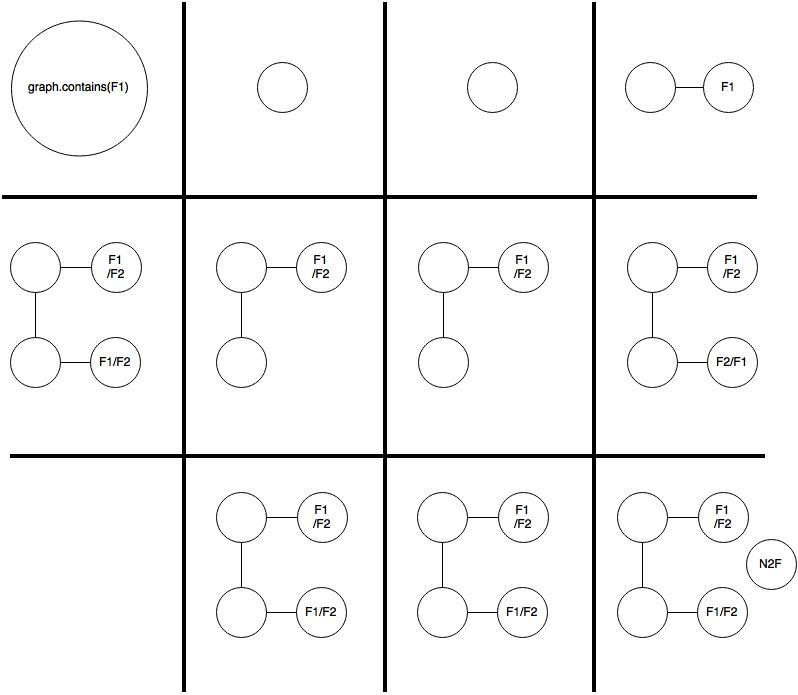
1. Jeder Knoten besitzt einen gerade Grad -> Eulerkreis

Aber der mittlere Knoten muss 2mal durchlaufen werden -> Kein Hamiltonkreis



### Aufgabe XI:

Sofern gegeben werden kann, dass die erste Regel nur einmal ausgeführt wird, kann mit den folgenden Regeln entweder eine 2 Färbung bestimmt, oder aber die nicht 2-Färbbarkeit bewiesen werden.

(Bei doppelten Bezeichnern in Knoten ist die Regel als 2 Regeln zu verstehen, wobei jeweils die ersten bzw. zweiten Bezeichner zueinander gehören)

### Aufgabe XII:

### Z:\git\GraphFramework\doc\Praktikum4\gkap_4_XII.jpg