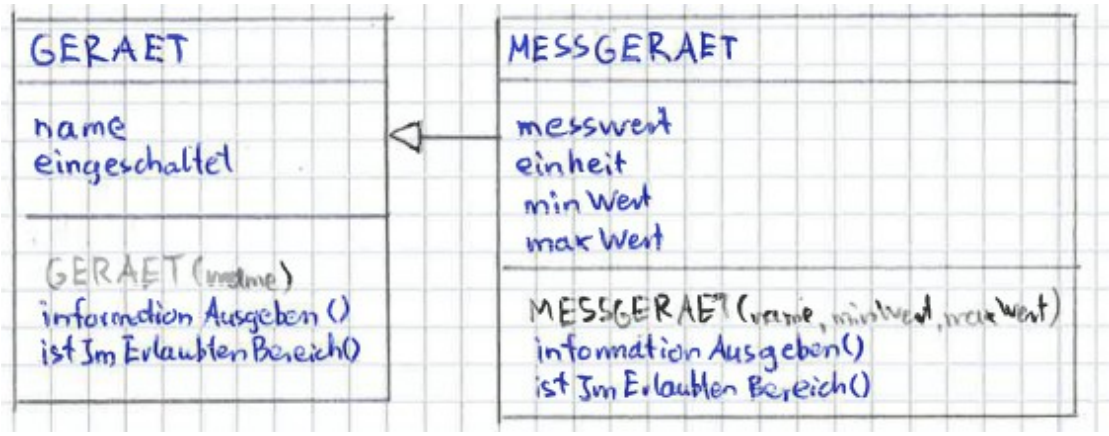
Informatik Abitur Bayern 2019 / II - Lösung

Autor:
Dippon (1, 2)
Neppl (3)
Rieger (4)
Reinold (5)

1 Erstellen des Pflichtenhefts → Analyse
Einsatz von Softwaremustern → Entwurf
Aufwandsabschätzung → Analyse (oder auch: Entwurf)
Modellieren mithilfe von Diagrammen → Entwurf (oder auch: Analyse)
weitere Phasen: Implementierung, Test und Wartung

4

2a



5

2b

13

```
public class GERAET{
	private String name;
	protected boolean eingeschaltet;
	public GERAET(String name){
		this.name = name;
		eingeschaltet = false;
	}
	public void informationAusgeben(){
		String text;
		if (eingeschaltet){
			text = "an";
		} else {
			text = "aus";
		}
		System.out.println(name + ": " + text);
	}
	public boolean istImErlaubtenBereich(){
		return true;
	}
}


public class MESSGERAET extends GERAET{
	private int messwert, minWert, maxWert;
	private String einheit;
	public MESSGERAET(String name, int min, int max, String einheit){
		super(name);
		minWert = min;
		maxWert = max;
		messwert = min;
		this.einheit = einheit;
	}
	public void informationAusgeben(){
		String text;
		if (super.eingeschaltet){
			text = messwert+" "+einheit;
		} else {
			text = "aus";
```

```
            }
            System.out.println(super.nameGeben() + ": " + text);
        }
        public boolean istImErlaubtenBereich(){
            if(messwert >= minWert && messwert <= maxWert)
            return true;
            else return false;
        }
    }
```

2c

```
public class LISTE{
...
        public void handlungsbedarfAnzeigen(){
            erster.handlungsbedarfAnzeigen();
        }
    }

    public abstract class LISTENELEMENT{
    ...
        public abstract void handlungsbedarfAnzeigen();
    }

    public class ABSCHLUSS...{
    ...
        public void handlungsbedarfAnzeigen(){}
    }

    public class KNOTEN...{
    ...
        public void handlungsbedarfAnzeigen(){
            if (!inhalt.istImErlaubtenBereich()){
                inhalt.informationAusgeben();
            }
            nachfolger.handlungsbedarfAnzeigen();
        }
    }
```

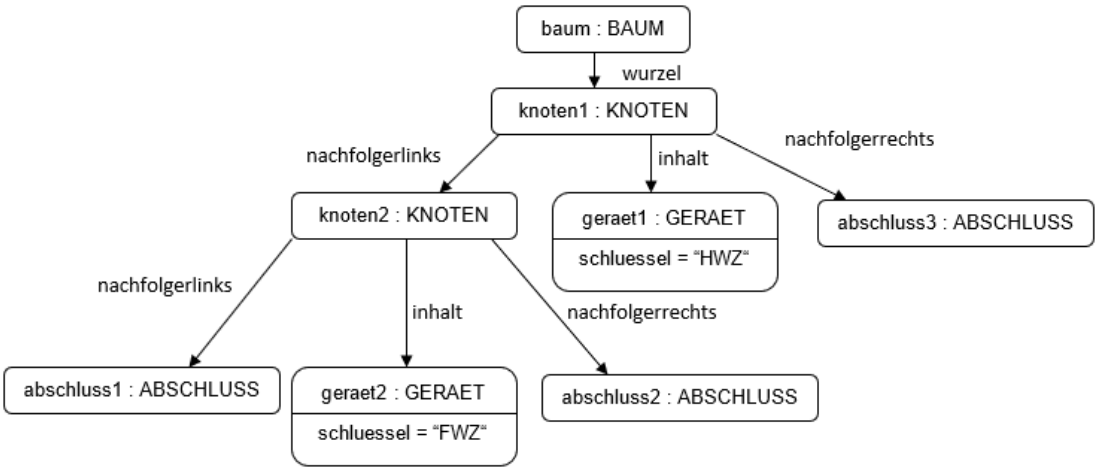
6

2d

```
public class KNOTEN{
...
        public KNOTEN vorruecken(GERAET geraet){
            if(naechster.inhaltGeben().equals(geraet)){
                KNOTEN t = naechster;
                naechsterSetzen(t.naechsterGeben());
                t.naechsterSetzen(this);
                return t;
            }
            else{
                naechster = naechster.vorruecken(geraet);
                return this;
            }
        }
    }
```

7

3a 6



3b $2^n - 1 > 534, woraus\ folgt: n > log_2(534+1) \approx 9{,}1$ 4

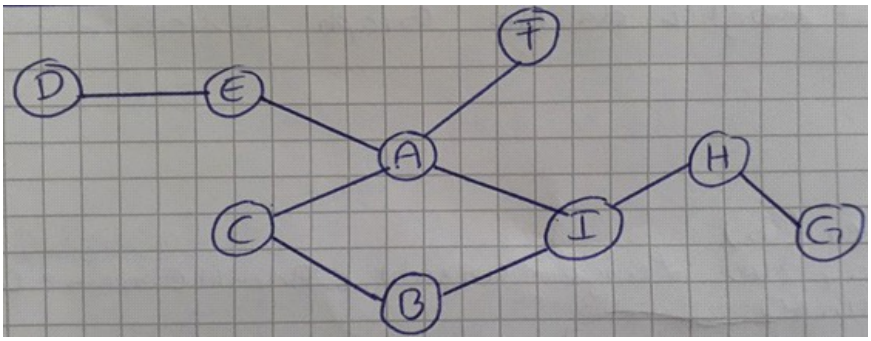
Die minimale Anzahl an Ebenen beträgt also 10.
Die maximale Anzahl an Ebenen beträgt 534, wenn der Baum zu einer Liste entartet ist.
Dieser Fall tritt auf, wenn die Knoten in alphabetischer Reihenfolge eingefügt werden.

3c z.B. HWZ, FWZ, LWZ, DK, HK, LB, SK, KK, LK 7

Preorder-Durchlauf: HWZ, FWZ, DK, HK, LWZ, LB, KK, LK, SK

Postorder-Durchlauf: DK,HK, FWZ, KK, LK, LB, SK, LWZ, HWZ

4a 5



	A	B	C	D	E	F	G	H	I
A			1		1	1			1
B			1						1
C	1	1							
D					1				
E	1			1					
F	1								
G								1	
H							1		1
I	1	1						1	

4b Der Algorithmus arbeitet nach dem Prinzip der Tiefensuche. Es wird die Adjazenzmatrix *matrix* und die Anzahl der Knoten *anzahl* benötigt. 9

```
Steht für die Reihenfolge der Räume
reihenfolge = Array der Raumindices mit der Länge anzahl
Enthält die bereits besuchten Räume – Startwerte false
besucht = Array mit der Länge anzahl
Nächste Position zum Einfügen in die Reihenfolge
aktuelleStelle = 0

Methode GibReihenfolgeRoboter(startknoten)
	geheTiefer(startknoten)
	return reihenfolge
endeMethode

methode geheTiefer(k)
	besucht[k] = true
	wiederhole für m = 0 bis m = anzahl-1
		wenn matrix[k][m] ungleich 0 und besucht[m] == false dann
			reihenfolge[aktuelleStelle] = m
			aktuelleStelle = aktuelleStelle + 1
			geheTiefer(m)
		endeWenn
	endeWiederhole
endeMethode
```

Die Reihenfolge des Roboters lautet:
A -> C -> B -> I -> H -> G (zurück bis A) -> E -> D (zurück bis a) -> F

Ein Durchlauf ist möglich, da der Graph zusammenhängend ist.

4c 7

```
Methode GibAnzahlEinsamerKnoten(GRAPH g)
	Enthält alle Knoten, die nur einen Nachbarn haben
	einsam = Array der Knotenindices der Länge anzahl
	Zählt die Anzahl der Knoten mit nur einem Nachbarn
	anzahlEinsam = 0
	Zählt für jeden Knoten die Anzahl der Nachbarn
	anzahlNachbarn
	wiederhole für k = 0 bis k = anzahl-1
		Geht für jeden einzelnen Knoten alle mögliche Nachbarknoten
		durch
		anzahlNachbarn = 0
		wiederhole für m = 0 bis m = anzahl -1
			Ist der Zelleneintrag ungleich 0, so sind die Knoten
			benachbart
			wenn matrix[k][m] ungleich 0 dann
				anzahlNachbarn = anzahlNachbarn + 1
			endeWenn
		endeWiederhole
		Wenn der aktuelle Knoten nur einen Nachbarn hat dann wird er
		in die Liste der Knoten mit nur einem Nachbarn eingefügt
		wenn anzahlNachbarn ist gleich 1 dann
			einsam[anzahlEinsam] = k
			anzahlEinsam = anzahlEinsam + 1
		endeWenn
	endeWiederhole
	return einsam
endeMethode
```

5a Um eine hohe Unabhängigkeit zwischen Anzeigeoberfläche und Programmlogik zu realisieren, sollten Model und View tatsächlich getrennt werden. Der Controller dient als intelligenter Adapter zwischen Model und View, der Steuerbefehle der View an das Model aufbereitet und filtert. 4

5b Button geklickt 3
→ Information des Controllers über die Eingabe
→ Weiterleitung an das Model

Nach Erhöhung der Raumtemperatur sendet das Model an die View die neue Ist-Temperatur.

Empfehlenswert wäre hier eine erweiterte Anzeige, die auch die momentan eingestellte Temperatur (Soll-Temperatur) darstellt. Diese könnte entweder auch vom Model geliefert werden oder im Zuge der beschriebenen Interaktion vom Controller.