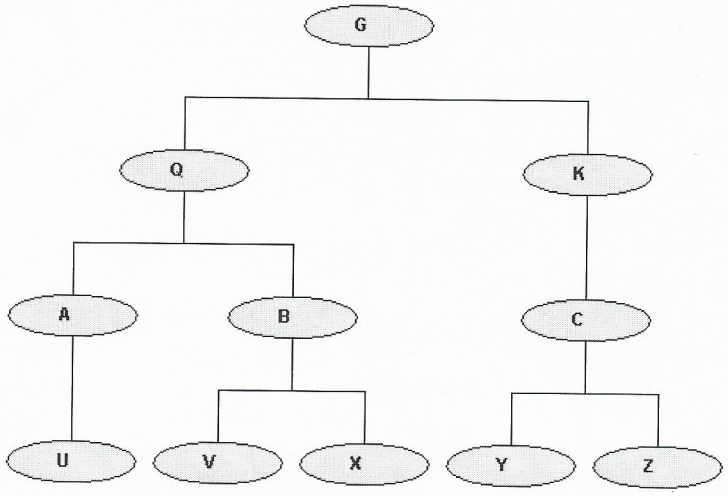
Aufgabe 1 (14 Punkte):

Gegeben sei der folgende Baum:



a) [2] Geben Sie die **inorder**-Reihenfolge der Knoten an:

U, A, Q, V, B, X, G, Y, C, Z, K

b) [2] Geben Sie die **preorder**-Reihenfolge der Knoten an:

G, Q, A, U, B, V, X, K, C, Y, Z

c) [2] Geben Sie die **postorder**-Reihenfolge der Knoten an:

U, A, V, X, B, Q, Y, Z, C, K, G

d) [2] Geben Sie die **levelorder**-Reihenfolge der Knoten an:

G, Q, K, A, B, C, U, V, X, Y, Z

<u>Nehmen Sie für die folgenden Fragen an, dass der Baum eine Containmenthierarchie repräsentiert</u>:

e) [2] Könnte der Knoten A ein JFrame-Objekt repräsentieren? Antwort <u>ohne Begründung.</u>

O ja

☒ nein

f) [2] Welche Knoten sind intermediäre Container?

Q, A, B, K, C

g) [2] Welcher der Knoten Q, K ist zur Aufnahme der Toolbar geeignet?

O Knoten ____, weil _____

☒ keiner, weil die Toolbar in den JFrame implementiert werden sollte, also G (ContentPane)

123

Aufgabe 2 (24 Punkte):
a) [6 *vgl. Hinweis 8*] Geben Sie an, ob die folgenden Aussagen richtig oder falsch sind:

	richtig	falsch
JFrame hat eine eigene Methode zur Aufnahme einer JMenuBar	⊗	○
intermediäre Container können intermediäre Container enthalten	⊗	○
Eine Zeichenkomponente muss die Superklasse JPaintComponent haben	○	⊗

6

b) [2] Welcher LayoutManager ordnet die Kindkomponenten wie Fließtext auf einer Schreibfläche an?

FlowLayout

2

c) [2] Geben Sie zwei verschiedene Swing-Klassen für atomare Komponenten an:

JButton, JTextField

2

d) [2] Welchen Eventtyp senden JMenuItems beim Anklicken?

ActionEvent

2

e) [2] Wofür steht die Abkürzung I18N?

Internationalisation

2

f) [2] Welches Layout soll der Container haben, in den ein ToolBar eingefügt wird?

BorderLayout

2

g) [4] Geben Sie möglichst genau an, woran man erkennt, daß die Objekte einer Klasse **`SelectionEvents`** aussenden können:

Die Klasse muss folgende Methoden aufweisen:
public void addSelectionListener(SelectionListener sl){...}
public void removeSelectionListener(SelectionListener sl){...}

4

h) [2] mit welcher Anweisung wird eine Exception geworfen, die anzeigt, dass eine Datei nicht gefunden wurde und die dabei auch mitteilt, dass der Pfad nicht existiert:

throw FileNotFoundException
throw new FileNotFoundException();

1

i) [2] Woran erkennt man, dass die Objekte einer Klasse eine checked Exception werfen könnten?

an dem Wort throws + Exception-Art neben den Methodenköpfen

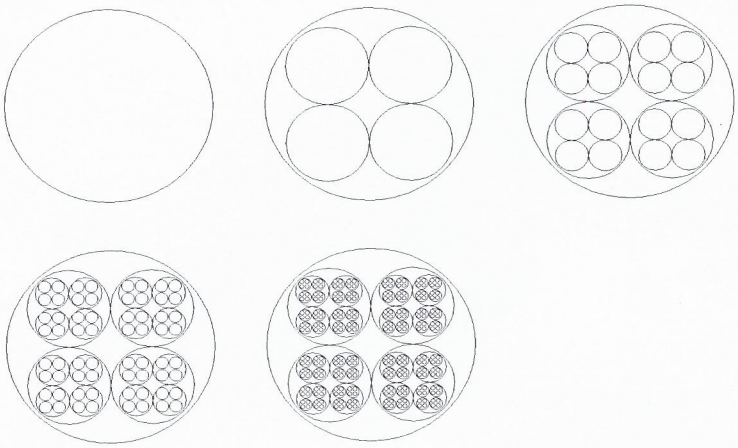
2

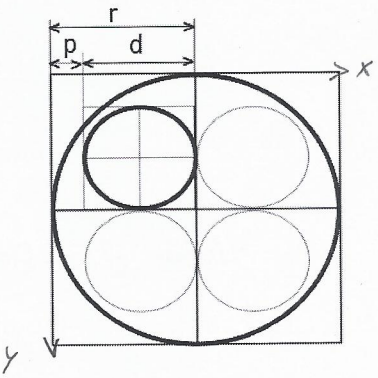
Aufgabe 3 (12 Punkte):

12

Die folgende Bildreihe veranschaulicht den Aufbau eines **rekursiv** gezeichneten Musters aus Kreisen:

Das Muster besteht aus einem Kreis mit Radius r (1. Bild). Dieser Kreis enthält vier innere Kreise mit den Durchmessern d (2. Bild). Jeder dieser Kreise enthält wiederum 4 kleinere Kreise (3.Bild). Diese Kreise enthalten noch kleinere Kreise usw., solange der Radius einen Wert größer als 4 Pixel hat.





Hinweise:

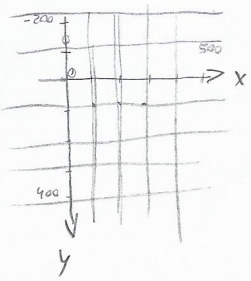
1) vgl. Bild: Für den Randabstand p gilt: $p = 0{,}18 \cdot r$
Für den Durchmesser d gilt: $d = 0{,}82 \cdot r$

2) Die Reihenfolge, in der Sie die Kreise zeichen, darf von der obigen Beschreibung beliebig abweichen.

Aufgabe (12 Punkte): Ergänzen Sie die folgende Methode. Verwenden Sie **Rekursion**:

*/** Zeichnet das oben beschriebene Muster durch Rekursion. **x** und **y** sind die Koordinaten der linken oberen Ecke des Rechtecks, das das Muster umfasst. **r** ist der Radius des äußersten Kreises des Musters. */*

```
public static void drawCR(Graphics g, int x, int y, int r) {
    if( r <= 4 ) {
        return;
    }
    int p = (int)(0.18*r);
    int d = (int)(0.82*r);
    g.drawOval(x, y, 2*r, 2*r);
    drawCR(g, x+p, y+p, d/2);
    drawCR(g, x+r, y+p, d/2);
    drawCR(g, x+p, y+r, d/2);
    drawCR(g, x+r, y+r, d/2);
}
```



6

Aufgabe 4 (16 Punkte):

Gegeben sei:

```
public interface Function {
    public abstract int value (int argument);
}
```

Schreiben Sie eine Klasse **FunctionPlotter**, deren Objekte als Komponenten in einer Swing-GUI verwendet werden können. Objekte dieser Klasse sollen in einem Bereich der Ausdehnung von 0 bis 500 in x- und von -200 bis 400 in y-Richtung eine mathematische Funktion als roten Linienzug vor einem grauen Gitternetz zeichnen. Das Gitternetz soll einen Linienabstand von 25 Pixeln besitzen. Die mathematische Funktion wird als Konstruktorparameter des Typs **Function** an das **FunctionPlotter**-Objekt übergeben.

Hinweise: (1) Sie dürfen annehmen, dass die Werte der math. Funktion nie kleiner als -200 oder größer als 400 werden. (2) Beachten Sie die Richtung der y-Achse!

```
import java.awt.Graphics;
import java.swing.Component;
import Function;
import java.awt.Color;
public class FunctionPlotter implements Component {
    protected Function function;

    public FunctionPlotter (Function function, Graphics g ){
        this.function = function;
        g.setColor
        drawFunction (g, function);
        drawGrid(g);
    }

    public void drawFunction (Graphics g, Function function ){
        g.setColor(Color.RED);
        g.drawLine( function.getArgument(), function.getValue(), function.getArgument+500, function.getValue+100);
    }

    public void drawGrid(Graphics g){
        g.setColor(Color.GRAY);
        int xs = 0;
        int xe = 500;
        int ys = -200;
        int ye = 400;
        for ( xs; xs < xe; xs+=25){
            g.drawLine (xs,ys,xs,ye);
        }
        for(ys; ys<ye; ys+=25){
            g.drawLine (xs, ys, xe, ys);
        }
    }
}
```

3

paint Component

(-6)

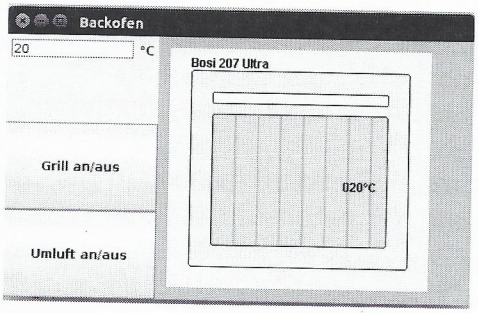
(-1)

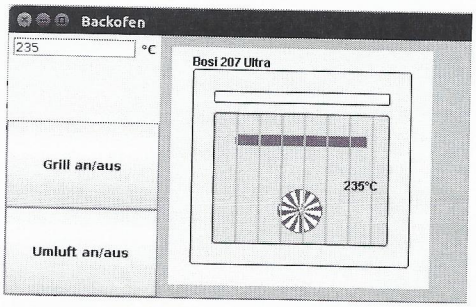
24

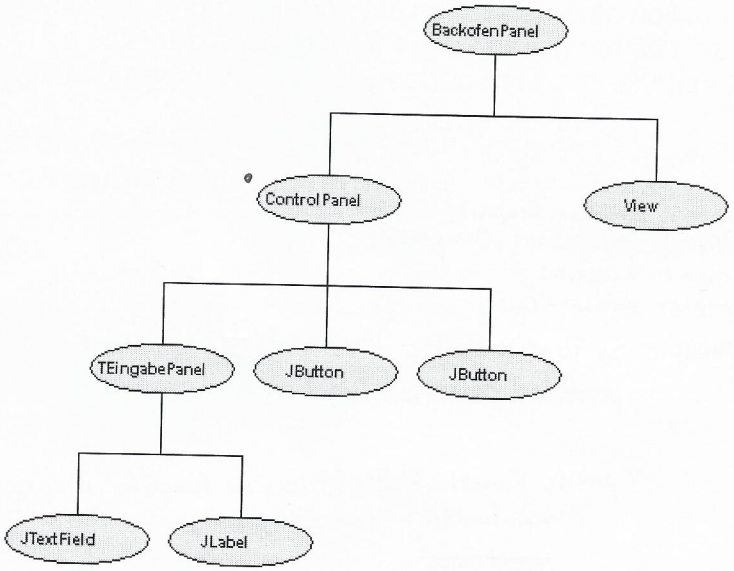
Aufgabe 5 (24 Punkte):

Die folgende Abbildung zeigt die grafische Benutzeroberfläche einer Applikation, die entsprechend den im Unterricht besprochen Mustern Model-View-Controller-Architektur und SwingStandardForm programmiert wurde.

Die Applikation zeigt einen Backofen, bei dem die Temperatur eingestellt werden kann. Zusätzlich können unabhängig voneinander ein Grill und ein Umluftventilator an- oder abgeschaltet werden. In den beiden Bildern links sind zwei verschiedene Betriebszustände dargestellt. Im Bild rechts ist ein Teil der Containmenthierarchie dargestellt.







Von der Applikation sei bekannt, dass Model, View, Controller und das BackofenPanel in main() erzeugt werden.
Die GUI ist folgendermassen aufgebaut: Das **BackofenPanel** enthält ein **ControlPanel** und den **View** (die Ansicht des Backofens). Das ControlPanel wiederum enthält ein **TEingabePanel** und die beiden Schaltflächen. Das **TEingabePanel** enthält ein Eingabefeld und eine Beschriftungsfläche für die Temperatureinheit (vgl. Abb. rechts).

Gegeben sind die folgenden Klassen, die entsprechend ihrer Rolle in der MVC-Architektur benannt sind. Alle diese Klassen haben sinnvolle Konstruktoren.

```
class Model        // enthält u.a. die Methoden:
      public void umluftUmschalten()
      public void grillUmschalten()
      public void setT(int temperature)

class View         // die Ansicht des  Backofens

class Controller   // der zentrale Controller

class TEingabePanel // Komponente zur Temperatureingabe

class Befehl {
      public static final String GRILL_AN_AUS   = "GRILL_AN_AUS";
      public static final String UMLUFT_AN_AUS  = "UMLUFT_AN_AUS";
      public static final String SET_T          = "SET_T";
}
```

Beachten Sie bei den folgenden Aufgabenteilen das Einhalten von Model-View-Controller-Architektur und SwingStandardForm!

a) [4] Ergänzen Sie die Programmzeile

```
public class View extends JPanel implements PropertyChangeListener {
```

4

b) [10] Schreiben Sie für die oben beschriebene Applikation die Klasse

```
public class ControlPanel extends JPanel {

    public ControlPanel (ActionListener al ){
        this.setLayout(new GridLayout(3,0));
        this.add(new TEingabePanel(al));
        JButton grill = new JButton("Grill an/aus");
        JButton umluft = new JButton("Umluft an/aus");
        grill.setActionCommand(Befehl.GRILL_AN_AUS);
        umluft.setActionCommand(Befehl.UMLUFT_AN_AUS);
        this.add(grill);
        this.add(umluft);

    }

}
```

10

d) [10] Schreiben Sie für die oben beschriebene Applikation die main()-Methode in der Klasse BackofenApplikation:

```
public class BackofenApplikation                                    {

    // Model, View, Controller und das BackofenPanel werden in main() erzeugt
    public static void main (String [] args)                        {

        Model model = new Model();
        View view = new View(model);
        Controller controller = new Controller(model);

        JFrame frame = new JFrame("Backofen");
        Container container = frame.getContentPane();
        container.setLayout(new BorderLayout());
        container.add(new BackofenPanel(controller, view), BorderLayout.CENTER);
        model.addPropertyChangeListener(view);

        frame.setSize(800,800);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

    }
}
```

10