Hausaufgabe 8

Tim Wende

November 29, 2021

Nachrichtenverwaltungssystem

Führen Sie die Aufgabe aus dem Praktikum zu Ende.
Zur Erinnerung: In einem Nachrichtenverwaltungssystem können sich KundInnen für unterschiedliche Nachrichtenkanäle interessieren. Dabei kann ein/e KundIn Nachrichtenkanäle abonnieren und wieder abbestellen. Ändert sich ein Nachrichteninhalt eines Nachrichtenkanals, so werden alle AbonnentInnen (KundInnen) unmittelbar informiert. Der/Die KundIn kann zwischen verschiedenen Abrechnungsarten für seine/ihre Abonnements wählen. Für das konkrete Beispiel soll es die Varianten geben, dass für jede Nachricht ein fester Betrag abgerechnet wird und dass man nur jeweils nach jeder dritten Nachricht einen Betrag bezahlen muss. Weitere Abrechnungsarten sollen leicht hinzugefügt werden können.

a. Entwerfen Sie ausgehend von den Klassen Nachrichtenkanal und Kunde ein **Klassendiagramm**, das die genannten Forderungen umsetzt. Dabei sollen Sie für die Nachrichten und Kunden das Beobachter-Muster (Observer-Observable-Pattern) und das Strategy-Pattern für die Bezahlmethode einsetzen.

   Hierzu habe ich zuerst eine grobe Übersicht erstellt, um nicht planlos anzufangen:

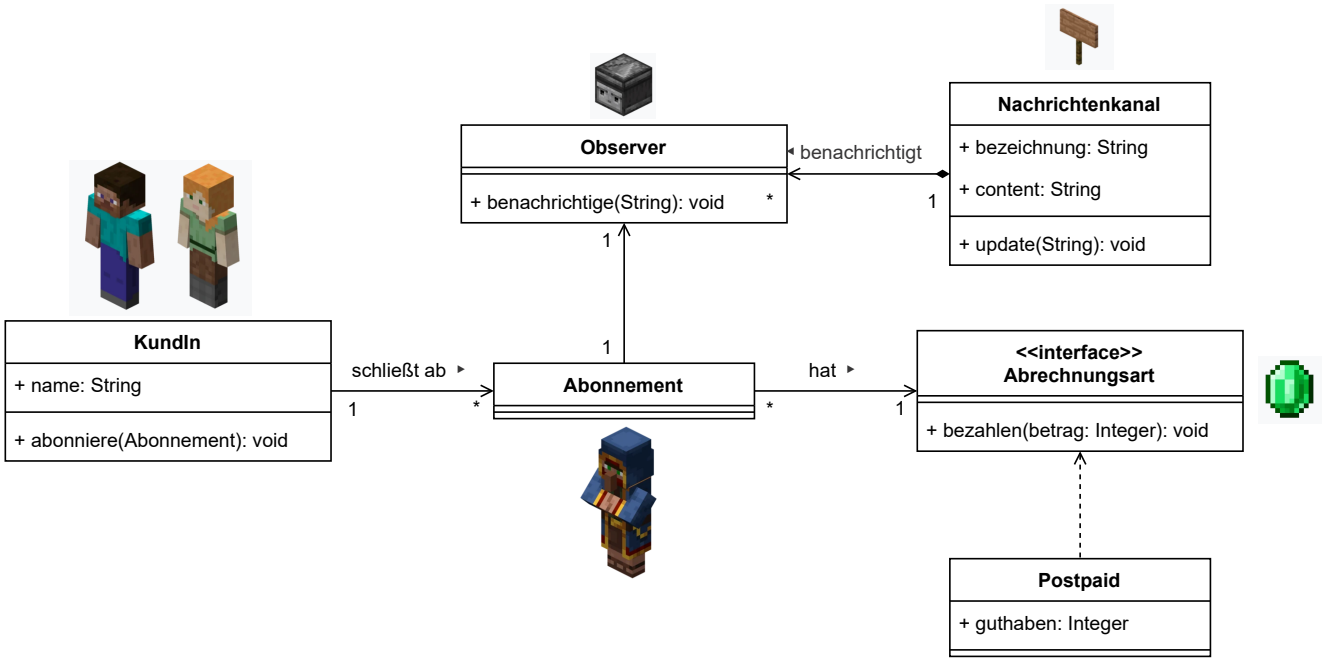

Figure 1: `class_diagram_pre`

Da diese offensichtlich von dem fertigen Code abweicht, hier nochmal eine aktualisierte Version:

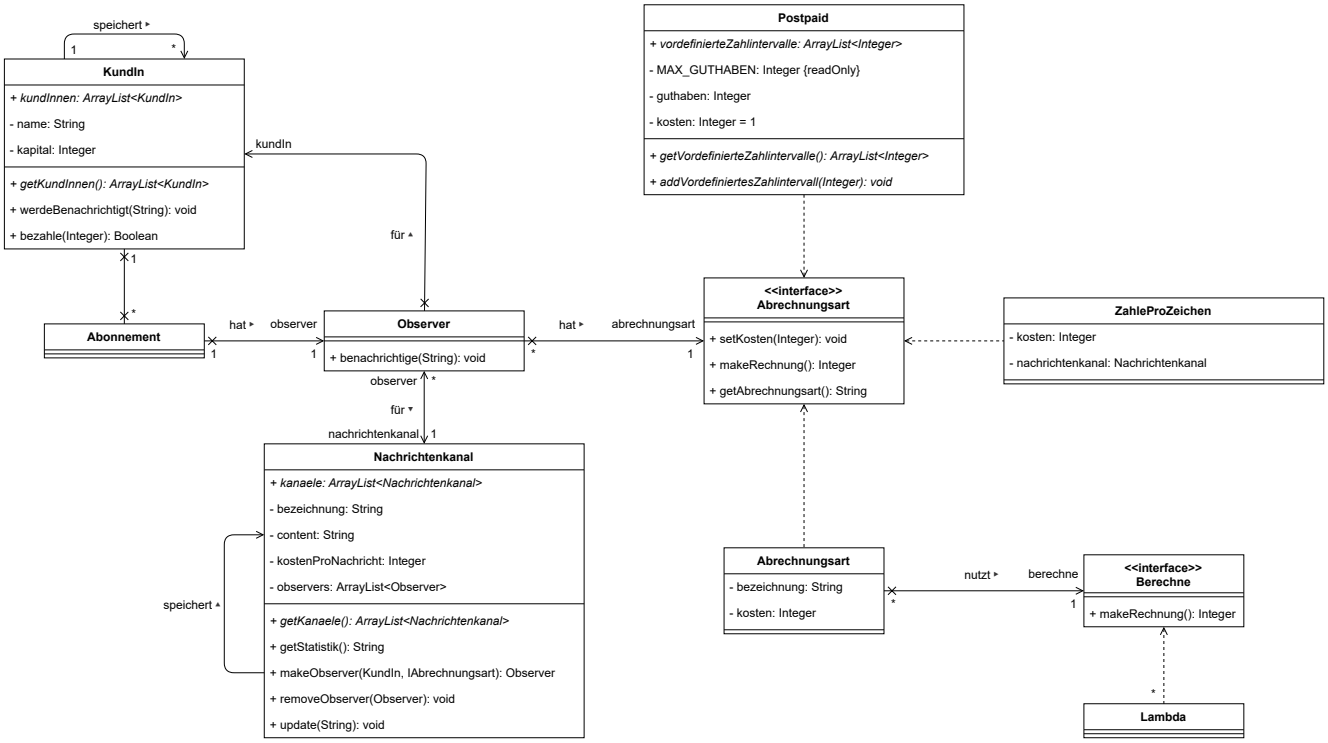

Figure 2: `class_diagram_post`

Diese unterscheiden sich nicht großartig voneinander, die zweite Version, welche nach der ersten Iteration revisiev erstellt wurde, ist jedoch ausführlicher und weiß genauer, welche Datentypen genutzt wurden.

Konstruktoren sowie getter/ setter wurden im Klassendiagramm weg gespart. Diese sind meist offensichtlich in ihrem Verhalten.

- Observer-Observable-Pattern
  Als Observer dient die Klasse „Observer“
  Als Observable dient die Klasse „Nachrichtenkanal“
  Nachrichtenkanal speichert alle zu sich gehörenden Observer und ruft deren `benachrichtige`-Methode auf, sollte sich der Content ändern. Dies geschieht weder automatisch via Trigger noch sonst irgendwie sinnvoll. Nach Rücksprache stellte sich heraus, dass der Vorgang, wie hier zu sehen, gewollt ist.

- Strategy-Pattern
  Vorab: man hat 2 Möglichkeiten Abrechnungsarten hinzuzufügen

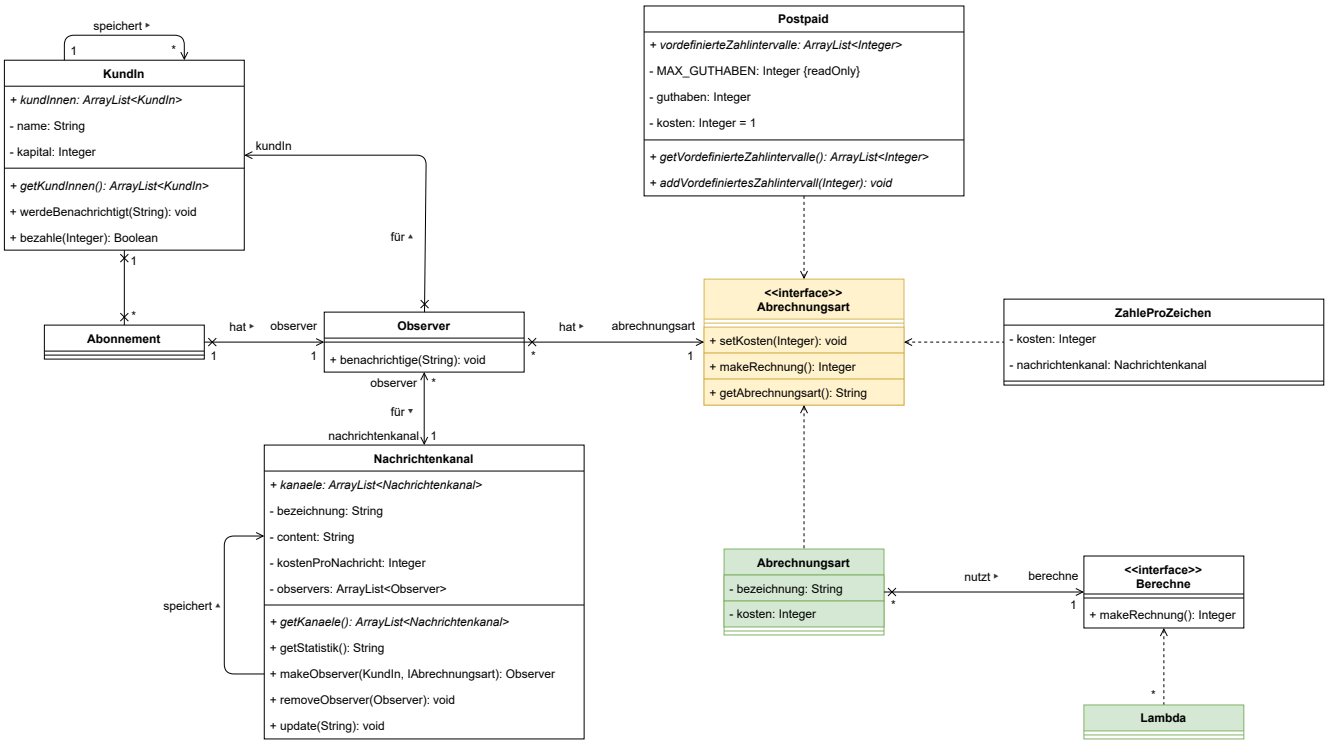

Figure 3: `class_diagram_options`

1. Man erstellt eine Klasse und implementiert „IAbrechnungsart“

```
public class ZahleJedeNachricht implements IAbrechnungsart {
    int kosten;

    public void setKosten(int kosten){
        this.kosten = kosten;
    }

    public int makeRechnung(){
        return -1 * betrag;
    }

    public String getAbrechnungsart(){
        return "Zahle jede Nachricht";
    }
}
```

2. Man erstellt ein Objekt der Klasse „Abrechnungsart“ mit einer `Bezeichnung` und einem `Lambda Ausdruck`, welcher als Parameter und return einen int erhält/ zurückgibt.

```
new Abrechnungsart("Zahle jede Nachricht", betrag -> -1 * betrag);
```

Für „einfache“ Abrechnungsarten sei die zweite Methode zu bevorzugen.

b. Geben Sie dann `Implementierungen` aller Klassen an. Beachten Sie, dass für versandte Nachrichten eine Abrechnung erfolgen muss. Binden Sie den Nutzungsdialog (siehe Main.java und Beispiel-Output) mit ein. Mit ihm kann man neue Nachrichten und Kunden anlegen. Außerdem kann der Kunden Nachrichtenkanäle mit einer auswählbaren Abrechnungsart wählen und Informationen über alle Nachrichtenkanäle zusammen mit den angefallenen Abokosten anzeigen. Natürlich kann man auch die Nachrichteninhalte eines Nachrichtenkanals verändern. Der Nutzungsdialog kann nach einigen Schritten wie folgt aussehen, Eingaben sind umrandet (Die Datei Main.java sollte entsprechend ergänzt werden).

Schauen wir uns dazu erstmal den gegebenen sowie meinen Output an:

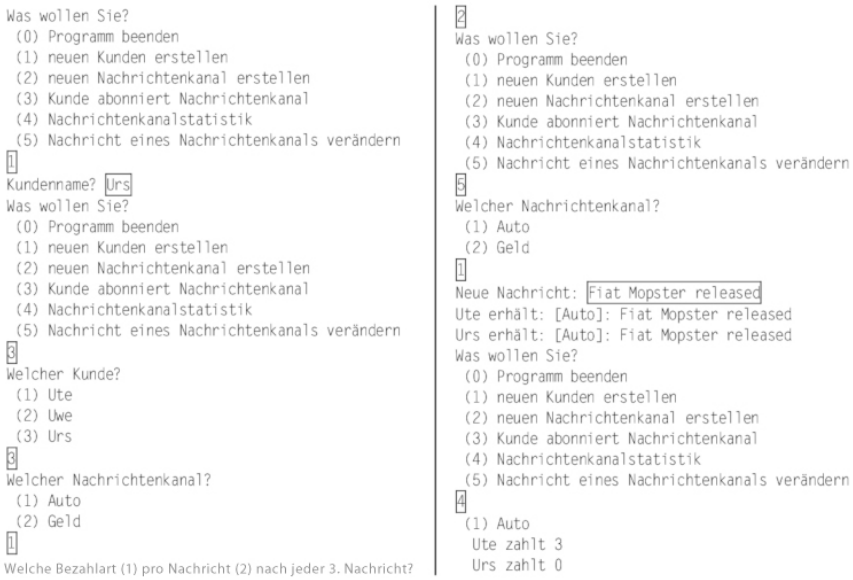

Figure 4: `class_diagram_01`

Figure 5: `class_diagram_02`

```
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
1
Name des/der KundIn: Tim
+---------------------------------------------------------------------------+
|  Name:    Tim                                                             |
|  Kapital: 5€                                                              |
+---------------------------------------------------------------------------+
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
2
Bezeichnung des Nachrichtenkanals: heise.de/newsticker
+---------------------------------------------------------------------------+
|  Bezeichnung: heise.de/newsticker                                         |
|  Content    : null                                                        |
+---------------------------------------------------------------------------+
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
3
Welcher Nachrichtenkanal?
Es wurde automatisch heise.de/newsticker gewählt
Welche/r KundIn?
Es wurde automatisch Tim gewählt
Nach wie vielen Nachrichten soll gezahlt werden?
1: 1
2: 3
2
+---------------------------------------------------------------------------+
|  KundIN:         Tim                                                      |
|  Abrechnungsart: Postpaid (alle 3 Nachrichten)                            |
+---------------------------------------------------------------------------+
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
5
Neue Nachricht: Welcher Nachrichtenkanal?
Es wurde automatisch heise.de/newsticker gewählt
UPDATE 1
#heise.de/newsticker update: UPDATE 1
    @Tim: 0
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
```

```
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
5
Neue Nachricht: Welcher Nachrichtenkanal?
Es wurde automatisch heise.de/newsticker gewählt
UPDATE 2
#heise.de/newsticker update: UPDATE 2
    @Tim: 0
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
5
Neue Nachricht: Welcher Nachrichtenkanal?
Es wurde automatisch heise.de/newsticker gewählt
UPDATE 3
#heise.de/newsticker update: UPDATE 3
    @Tim: -3
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
2
Bezeichnung des Nachrichtenkanals: ili.fh-aachen.de
+---------------------------------------------------------------------------+
|  Bezeichnung: ili.fh-aachen.de                                            |
|  Content    : null                                                        |
+---------------------------------------------------------------------------+
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
4
Welcher Nachrichtenkanal?
1: heise.de/newsticker
2: ili.fh-aachen.de
1
heise.de/newsticker - 1€/Nachricht - Anzahl der AbonnentInnen: 1
Was wollen Sie?
 (0) Programm beenden
 (1) neuen Kunden erstellen
 (2) neuen Nachrichtenkanal erstellen
 (3) Kunde abonniert Nachrichtenkanal
 (4) Nachrichtenkanalstatistik
 (5) Nachricht eines Nachrichtenkanals verändern
0
```

Zusätzlich der Output von ein paar automatisch getesteten Klassen bzw. erstellten Objekten:

```
+-------------------------------------------------------------------------+
|  Name:    Tim Wende                                                     |
|  Kapital: 5€                                                            |
+-------------------------------------------------------------------------+
+-------------------------------------------------------------------------+
|  Name:    Paddel                                                        |
|  Kapital: 10€                                                           |
+-------------------------------------------------------------------------+
+-------------------------------------------------------------------------+
|  Bezeichnung: heise.de/newsticker                                       |
|  Content:     null                                                      |
|                                                                         |
|  AbonnentInnen (2):                                                     |
|  +--------------------------------------------------------------------+ |
|  |  KundIN:         Tim Wende                                         | |
|  |  Abrechnungsart: Postpaid                                          | |
|  |                                                                    | |
|  |  KundIN:         Paddel                                            | |
|  |  Abrechnungsart: Postpaid (alle 3 Nachrichten)                     | |
|  +--------------------------------------------------------------------+ |
+-------------------------------------------------------------------------+

#heise.de/newsticker update: Kurz informiert: Rangliste aller Pokemon
    @Tim Wende: -3
    @Paddel: 0
#heise.de/newsticker update: "Pokemon strahlender Diamant" angespielt
    !Tim Wende hat nicht bezahlt. Das Abonnement wird aufgelöst!
    @Paddel: 0
#heise.de/newsticker update: Praxisversuch: Panflam auf 6000°C erhitzen
    @Paddel: -9

+-------------------------------------------------------------------------+
|  Name:    Tim Wende                                                     |
|  Kapital: 2€                                                            |
+-------------------------------------------------------------------------+
+-------------------------------------------------------------------------+
|  Name:    Paddel                                                        |
|  Kapital: 1€                                                            |
+-------------------------------------------------------------------------+
+-------------------------------------------------------------------------+
|  Bezeichnung: heise.de/newsticker                                       |
|  Content:     Praxisversuch: Panflam auf 6000°C erhitzen                |
|                                                                         |
|  AbonnentInnen (1):                                                     |
|  +--------------------------------------------------------------------+ |
|  |  KundIN:         Paddel                                            | |
|  |  Abrechnungsart: Postpaid (alle 3 Nachrichten)                     | |
|  +--------------------------------------------------------------------+ |
+-------------------------------------------------------------------------+
```

Diese wurden von meiner Testklasse erstellt:

```
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;

public class Test {
	/*
	 * Diese Klasse ist an diversen TODO-Stellen unvollstaendig und muss ergaenzt
	 * werden.
	 */
	// merkt man; wieso Main als Objekt???

	public static final int MAX_WIDTH = 75;
	public static final Scanner scanner = new Scanner(System.in);

//private static ArrayList<KundIn> kundInnen = new ArrayList<KundIn>(); // nein :/
//private static ArrayList<Nachrichtenkanal> kanaele = new ArrayList<Nachrichtenkanal>(); // nein :/

	public static void main(String[] args) {
		//simuliere();

		dialog();
	}

	/**
	 * static simuliere
	 *
	 * testet ein bisschen vor sich hin
	 */
	public static void simuliere(){
		KundIn tim = new KundIn("Tim Wende");
		KundIn paddel = new KundIn("Paddel", 10);
		//KundIn lambdaUser = new KundIn("LambdaUser", 160);

		Nachrichtenkanal heise = new Nachrichtenkanal("heise.de/newsticker", 3);

		new Abonnement(heise.makeObserver(tim, new Postpaid()));
		new Abonnement(heise.makeObserver(paddel, new Postpaid(3)));

		// Strategy-Pattern mit lambda Expressions > Strategy-Pattern
		//new Abonnement(heise.makeObserver(lambdaUser,
		//		new Abrechnungsart("Zahle jede Nachricht", betrag -> -1 * betrag)));
		//new Abonnement(heise.makeObserver(lambdaUser,
		//		new Abrechnungsart("Zahle jede Nachricht doppelt", betrag -> -2 * betrag)));
		//new Abonnement(heise.makeObserver(lambdaUser,
		//		new Abrechnungsart("Premium User; muss nichts zahlen", betrag -> 0)));
		//new Abonnement(heise.makeObserver(lambdaUser,
		//		new Abrechnungsart("Wie ich will", betrag -> 7)));

		for (KundIn kundIn : KundIn.getKundInnen()) {
			kundIn.print();
		}
		for (Nachrichtenkanal kanal : Nachrichtenkanal.getKanaele()) {
			kanal.print();
		}

		System.out.println();
		heise.update("Kurz informiert: Rangliste aller Pokemon");
		heise.update("\"Pokemon strahlender Diamant\" angespielt");
		heise.update("Praxisversuch: Panflam auf 6000C erhitzen");
```

```
        System.out.println();
        for (KundIn kundIn : KundIn.getKundInnen()) {
                kundIn.print();
        }
        for (Nachrichtenkanal kanal : Nachrichtenkanal.getKanaele()) {
                kanal.print();
        }
}

/**
 * Macht ne Box um Output drum herum
 *
 * @param content Output
 * @return drum herumte Box
 */
public static String box(String content) {
        return box(content, MAX_WIDTH);
}

/**
 * Macht ne Box um Output drum herum
 *
 * @param content Output
 * @return drum herumte Box
 */
public static String box(String content, int width) {
        String ret = "+" + "-".repeat(width - 2) + "+";
        for (String line : content.split("\n"))
                ret +=        "\n|  " + line +
                        " ".repeat(width - 6 - line.length()) + "  |";
        return ret + "\n+" + "-".repeat(width - 2) + "+";
}

// ab hier vorgegebene Main.java "leicht" angepasst

/**
 * static vorgegebener dialog
 *
 * @author Tim Wende, swt Leuts
 */
public static void dialog(){
        Postpaid.addVordefiniertesZahlintervall(1);
        Postpaid.addVordefiniertesZahlintervall(3);

        int eingabe;

        do {
                System.out.println("Was wollen Sie?\n"
                                +" (0) Programm beenden\n"
                                +" (1) neuen Kunden erstellen\n"
                                +" (2) neuen Nachrichtenkanal erstellen\n"
                                +" (3) Kunde abonniert Nachrichtenkanal\n"
                                +" (4) Nachrichtenkanalstatistik\n"
                                +" (5) Nachricht eines Nachrichtenkanals verndern");
                eingabe = nummerWaehlen();
                switch(eingabe){
                case 1:
                        neue_rKundIn();
                        break;
                case 2:
                        neuerNachrichtenkanal();
                        break;
```

```
                case 3:
                        neuesAbonnement();
                        break;
                case 4:
                        nachrichtenkanalStatistik();
                        break;
                case 5:
                        nachrichtVeraendern();
                        break;
                }
        } while (eingabe != 0);
}

/**
 * static String eingeben
 *
 * @return eingegebener String
 */
private static String textEingeben() {
        return scanner.next();
}

/**
 * static recursive int eingeben
 *
 * @return eingegebener int
 */
private static int nummerWaehlen() {
        try {
                return scanner.nextInt();
        } catch (InputMismatchException e) {
                scanner.next();
                return nummerWaehlen();
        }
}

/**
 * static generic waehle Objekt aus einer Liste
 * Wir wollen ja keine code dopplung z.B. in neuesAbonnement; nicht wahr //@swt
 *
 * @param <O> Generischer Objekttyp in der ArrayList
 * @param list ArrayList
 * @return Generisches Objekt an der gewaehlten stelle
 */
public static <O> O objectWaehlen(ArrayList<O> list) {
        if (list.size() == 0)
                throw new ArithmeticException("Liste ist leer");
        if (list.size() == 1) {
                System.out.println("Es wurde automatisch " + list.get(0) + " gewhlt");
                return list.get(0);
        }
        int i = 1;
        for (O object : list)
                System.out.println(i++ + ": " + object);
        do
                i = nummerWaehlen();
        while (i <= 0 || i > list.size());
        return list.get(i - 1);
}
```

```java
// Die folgenden Methoden koennen natuerlich nicht zur Automatisierung genutzt werden,
// da per console Text eingegeben werden muss
// Dies ist natuerlich hoechst sinnvoll; danke @swt

/**
 * static Waehle KundIn
 *
 * @return gewaehlte/r KundIn
 */
public static KundIn kundInWaehlen() {
        System.out.println("Welche/r KundIn?");
        return (KundIn) objectWaehlen(KundIn.getKundInnen());
}

/**
 * static Waehle Nachrichtenkanal
 *
 * @return gewaehlter Nachrichtenkanal
 */
public static Nachrichtenkanal nachrichtenkanalWaehlen() {
        System.out.println("Welcher Nachrichtenkanal?");
        return (Nachrichtenkanal) objectWaehlen(Nachrichtenkanal.getKanaele());
}

/**
 * static Waehle vordefiniertens Zahlungsintervall
 *
 * @return gewaehltes Zahlungsintervall
 */
public static int vordefiniertenZahlintervallWaehlen() {
        System.out.println("Nach wie vielen Nachrichten soll gezahlt werden?");
        return (int) objectWaehlen(Postpaid.getVordefinierteZahlintervalle());
}

/**
 * static nachrichtVeraendern
 */
public static void nachrichtVeraendern() {
        System.out.print("Neue Nachricht: ");
        nachrichtenkanalWaehlen().update(textEingeben());
}

/**
 * static nachrichtenkanalStatistik
 */
public static void nachrichtenkanalStatistik() {
        System.out.println(nachrichtenkanalWaehlen().getStatistik());
}

/**
 * static neuesAbonnement
 */
private static void neuesAbonnement() {
        (new Abonnement(nachrichtenkanalWaehlen().makeObserver(kundInWaehlen(),
                new Postpaid(vordefiniertenZahlintervallWaehlen())))).print();
}
```

```
        /**
         * static neuerNachrichtenkanal
         */
        private static void neuerNachrichtenkanal() {
                System.out.print("Bezeichnung des Nachrichtenkanals: ");
                (new Nachrichtenkanal(textEingeben(), 1)).print();;
        }

        /**
         * static neue_rKundIn
         */
        private static void neue_rKundIn() {
                System.out.print("Name des/der KundIn: ");
                (new KundIn(textEingeben())).print();
        }
}
```

Schauen wir uns zuerst die kleine Hilfsklasse `Abonnement` an:
Sollte `getStatistik` den bereits bezahlten Betrag ausgeben, kann man dies hier implementieren. Ansonsten kann die Klasse im Grunde auch gelöscht werden.

```
/**
 * Abonnement Klasse
 *
 * Theoretisch koennte man diese Klasse weglassen und Observer zu einem Abonnement umstrukturieren.
 * Da ich diese Klasse jedoch logisch sinnvoll finde, bleibt alles so wie es hier ist.
 *
 * @author Tim Wende
 *
 */
public class Abonnement {
        private Observer observer;

        /**
         * default ctor
         *
         * @param observer Observer
         */
        public Abonnement(Observer observer) {
                this.observer = observer;
        }

        /**
         * toString in schoen
         */
        public void print() {
                System.out.println(Test.box(observer.toString()));
        }
}
```

Schauen wir uns nun KundIn an:

```
import java.util.ArrayList;

/**
 * Kunde Klasse
 *
 * @author Tim Wende
 *
 */
public class KundIn {
	private static ArrayList<KundIn> kundInnen = new ArrayList<>(); // wieso sollte Main hierfr

	/**
	 * Gibt die static ArrayList mit allen KundInnen zurck
	 * Wenn man auf Datenkapselung achten würde, könnte man kundInnen clonen,
	 * so einen Schabernack machen wir hier aber natürlich nicht.
	 *
	 * @return alle KundInnen
	 */
	public static ArrayList<KundIn> getKundInnen() {
		return kundInnen;
	}

	private String name;
	private int kapital;

	/**
	 * default ctor, welcher genutzt wird, um chancengleich allen KundInnen dasselbe
	 * Startkapital zur Verfuegung zu stellen
	 *
	 * @param name
	 */
	KundIn(String name){
		this(name, 5); // default kapital: 5 Geld
	}

	/**
	 * ctor, welcher genutzt wird, falls man kapitalistisch die Armutsschere aufreissen moechte,
	 * durch den manche KundInnen einen privilegierteren Start in die Abonnementwelt erhalten
	 *
	 * @param name
	 * @param kapital
	 */
	KundIn(String name, int kapital){
		this.name = name;
		this.kapital = kapital;
		kundInnen.add(this);
	}

	/**
	 * Gibt den Namen des/r KundIn zurueck
	 *
	 * @return Name des/r KundIn
	 */
	public String getName() {
		return name;
	}
```

```java
        /**
         * Platzhalter; normale KundInnen wuerden hier lesen
         *
         * @param nachricht Nachricht des Nachrichtenkanals
         */
        public void werdeBenachrichtigt(String nachricht) {}

        /**
         * Bezahle
         *
         * @param betrag Betrag
         * @return true, falls bezahlt werden kann; false, wenn nicht
         */
        public boolean bezahle(int betrag) {
                if (kapital + betrag >= 0) {
                        kapital += betrag;
                        System.out.println("    @" + name + ": " + betrag);
                        return true;
                }
                return false;
        }

        /**
         * toString
         */
        public String toString() {
                return name;
        }

        /**
         * toString in schoen
         */
        public void print() {
                System.out.println(Test.box("Name: " + " ".repeat(3) + this.name + "\nKapital: " + t
        }
}
```

Von hier aus gehen wir weiter zu `Observer`:

```
public class Observer {
        KundIn kundIn;
        IAbrechnungsart abrechnungsart;
        Nachrichtenkanal nachrichtenkanal;

        /**
         * default ctor
         *
         * @param kundIn KundIn
         * @param abrechnungsart Objekt des Interfaces Abrechnungsart
         * @param nachrichtenkanal Nachrichtenkanal
         */
        public Observer(KundIn kundIn, IAbrechnungsart abrechnungsart, Nachrichtenkanal nachrichtenk
                this.kundIn = kundIn;
                this.abrechnungsart = abrechnungsart;
                this.nachrichtenkanal = nachrichtenkanal;
        }

        /**
         * Benachrichtige den/r KundIn
         *
         * @param nachricht Nachricht
         */
        public void benachrichtige(String nachricht) {
                if (kundIn.bezahle(abrechnungsart.makeRechnung()))
                        kundIn.werdeBenachrichtigt(nachricht);
                else {
                        nachrichtenkanal.removeObserver(this);
                        System.out.println("    !" + kundIn.getName() +
                                " hat nicht bezahlt. Das Abonnement wird aufgelst!");
                }
        }

        /**
         * toString
         */
        public String toString() {
                return "KundIN: " + " ".repeat(8) + kundIn.getName() + "\nAbrechnungsart: " + abrech
        }
}
```

Und nun zu Nachrichtenkanal:

```
import java.util.ArrayList;

public class Nachrichtenkanal {
	private static ArrayList<Nachrichtenkanal> kanaele = new ArrayList<Nachrichtenkanal>();

	/**
	 * Gibt die static ArrayList mit allen Kanaelen zurueck
	 *
	 * @return alle Kanaele
	 */
	public static ArrayList<Nachrichtenkanal> getKanaele() {
		return kanaele;
	}

	private String bezeichnung;
	private String content;
	private int kostenProNachricht;
	private ArrayList<Observer> observers = new ArrayList<>();

	/**
	 * default ctor
	 *
	 * @param bezeichnung Bezeichnung des Nachrichtenkanals
	 */
	public Nachrichtenkanal(String bezeichnung) {
		// 1 Geld pro Nachricht; das ist ja fast schon schlimmer als mms
		this(bezeichnung, 1);
	}

	/**
	 * default ctor
	 *
	 * @param bezeichnung Bezeichnung des Nachrichtenkanals
	 * @param kostenProNachricht Kosten pro Nachricht
	 */
	public Nachrichtenkanal(String bezeichnung, int kostenProNachricht) {
		this.bezeichnung = bezeichnung;
		this.kostenProNachricht = kostenProNachricht;
		kanaele.add(this);
	}

	/**
	 * gibt die Bezeichnung des Nachrichtenkanals zurueck
	 *
	 * @return Bezeichnung des Nachrichtenkanals
	 */
	public String getBezeichnung() {
		return bezeichnung;
	}

	/**
	 * Gibt den Content des Nachrichtenkanals zurueck
	 *
	 * @return Content des Nachrichtenkanals
	 */
	public String getContent() {
		return content;
	}
```

```java
/**
 * gibt die Kosten pro Nachricht zurueck
 *
 * @return Kosten pro Nachricht
 */
public int getKostenProNachricht() {
        return kostenProNachricht;
}

/**
 * gibt die Statistik des Nachrichtenkanals zurueck
 *
 * @return Statistik des Nachrichtenkanals
 */
public String getStatistik() {
        return bezeichnung + " - " + kostenProNachricht +
                "€/Nachricht - Anzahl der AbonnentInnen: " + observers.size();
}

/**
 * Erstellt einen Observer fuer ein passendes Abonnement
 *
 * @param kundIn KundIn
 * @param abrechnungsart Objekt des Interfaces Abrechnungsart
 * @return Observer
 */
public Observer makeObserver(KundIn kundIn, IAbrechnungsart abrechnungsart) {
        abrechnungsart.setKosten(kostenProNachricht);
        Observer observer = new Observer(kundIn, abrechnungsart, this);
        observers.add(observer);
        return observer;
}

/**
 * Loescht dem Observer, falls ein/e KundIn nicht gezahlt haben sollte
 *
 * @param observer zu loeschender Observer
 */
public void removeObserver(Observer observer) {
        observers.remove(observer);
}

/**
 * Setzt den Content des Kanals ohne ueberpruefung dessen neu
 *
 * @param content zu setzender Content
 */
public void update(String content) {
        System.out.println("#" + bezeichnung + " update: " + content);
        this.content = content;
        for (Observer observer : (ArrayList<Observer>) observers.clone())
                observer.benachrichtige(content);
}

/**
 * toString
 */
public String toString() {
        return bezeichnung;
}
```

```
        /**
         * toString in schoen
         */
        public void print() {
                String ret = "Bezeichnung: " + bezeichnung + "\nContent: " + " ".repeat(4) + content

                if (observers.size() == 0) {
                        System.out.println(Test.box(ret));
                        return;
                }

                String observerbox = "";
                String div = "";
                for (Observer observer : this.observers) {
                        observerbox += div + observer.toString();
                        div = "\n\n";
                }
                System.out.println(Test.box(ret + "\n\nAbonnentInnen (" + observers.size() +
                        "):\n" + Test.box(observerbox, Test.MAX_WIDTH - 6)));
        }
}
```

Dieser benötigt ein Objekt des Intefaces mit der statischen Klasse IAbrechnungsart:

```
/**
 * Abrechnungsart Interface
 *
 * @author Tim Wende
 *
 */
public interface IAbrechnungsart {
        /**
         * Setze die Kosten pro Nachricht
         *
         * @param kosten Kosten pro Nachricht
         */
        public void setKosten(int kosten);

        /**
         * Erstelle eine Rechung
         *
         * @return Betrag der Rechnung
         */
        public int makeRechnung();

        /**
         * Gibt den Namen der Abrechnungsart zurueck
         * .getType() moeglich, hier aber nicht genutzt
         *
         * @return NAme der Abrechnungsart
         */
        public String getAbrechnungsart();
}
```

Implementierte Klassen sind beispielsweise:

```java
import java.util.ArrayList;

public class Postpaid implements IAbrechnungsart {
	private static ArrayList<Integer> vordefinierteZahlintervalle = new ArrayList<>();

	/**
	 * Gibt die static ArrayList mit allen vordefinierten Zahlintervallen zurueck
	 *
	 * @return alle vordefinierten Zahlintervalle
	 */
	public static ArrayList<Integer> getVordefinierteZahlintervalle(){
		return vordefinierteZahlintervalle;
	}

	/**
	 * fuegt ein Zahlungsintervall der Abrechnungsart hinzu
	 *
	 * @param intevall Intervall (in Nachrichten)
	 */
	public static void addVordefiniertesZahlintervall(int intevall) {
		vordefinierteZahlintervalle.add(intevall);
	}

	private final int MAX_GUTHABEN;
	private int guthaben;
	private int kosten = 1;

	/**
	 * default ctor
	 */
	public Postpaid() {
		this(1); // zahle jede Nachricht
	}

	/**
	 * default ctor
	 * Kein Mengenrabatt enthalten!
	 * Bei 1 Geld pro Nachricht zahlt man bei max_guthaben =1 exakt gleichviel wie bei =2
	 * Der einzige Unterschied ist die Haeufigkeit. (2x 1 Geld / 1x 2 Geld)
	 *
	 * Dies wird natuerlich nicht ueber verschiedene Klassen geloest,
	 * alles andere waere ja auch peinlich tbh
	 *
	 * @param max_guthaben Nach wie vielen Nachrichten soll gezahlt werden
	 */
	public Postpaid(int max_guthaben) {
		MAX_GUTHABEN = max_guthaben;
		guthaben = MAX_GUTHABEN;
	}

	@Override
	public void setKosten(int kosten) {
		this.kosten = kosten;
	}
```

```
        @Override
        public int makeRechnung() {
                if (guthaben <= 1) {
                        guthaben = MAX_GUTHABEN;
                        return -1 * kosten * MAX_GUTHABEN;
                }
                guthaben--;
                return 0;
        }

        @Override
        public String getAbrechnungsart() {
                return "Postpaid" + ((MAX_GUTHABEN == 1) ? " (alle " + MAX_GUTHABEN + " Nachrichten)
        }
}
```

Sowie:

```
/**
 * Weitere Abrechungsart, welche leicht hinzugefuegt wurde
 *
 * @author Tim Wende
 *
 */
public class ZahleProZeichen implements IAbrechnungsart {
        int kosten;
        Nachrichtenkanal nachrichtenkanal;

        /**
         * default ctor
         *
         * @param nachrichtenkanal Nachrichtenkanal
         */
        public ZahleProZeichen(Nachrichtenkanal nachrichtenkanal) {
                this.nachrichtenkanal = nachrichtenkanal;
        }

        @Override
        public void setKosten(int kosten) {
                this.kosten = kosten;
        }

        @Override
        public int makeRechnung() {
                return -1 * nachrichtenkanal.getContent().length() * kosten;
        }

        @Override
        public String getAbrechnungsart() {
                return "ZahleProZeichen";
        }

}
```

Des weiteren existiert die Klasse `Abrechnungsart`, welche `IAbrechnungsart` Lambda-fähig macht:

```
/**
 * Schoenes Abrechnungsartengeruest zum noch leichteren Hinzufuegen von neuen Abrechnungsartens
 * via Lambda Expressions
 *
 * @author Tim Wende
 *
 */
public class Abrechnungsart implements IAbrechnungsart {
        String bezeichnung;
        IBerechne berechne;
        int kosten;

        public Abrechnungsart(String bezeichnung, IBerechne berechne) {
                this.bezeichnung = bezeichnung;
                this.berechne = berechne;
        }

        @Override
        public void setKosten(int kosten) {
                this.kosten = kosten;
        }

        @Override
        public int makeRechnung() {
                return berechne.makeRechnung(kosten);
        }

        @Override
        public String getAbrechnungsart() {
                return bezeichnung;
        }

}
```

Diese benötigt:

```
/**
 * Interface um IAbrechnungsart weg zu Lambda'en
 * So muss man nicht fuer jede weiter Abrechnungsart eine neue Klasse erstellen
 * Dies ist vermutlich die einfachste Moeglichkeit eine sehr simple Abrechnungsart hinzuzufuegen.
 *
 * @author Tim Wende
 *
 */
public interface IBerechne {
        /**
         * Erstelle eine Rechung
         *
         * @param kosten Kosten pro Nachricht
         * @return Betrag der Rechnung
         */
        int makeRechnung(int kosten);
}
```