Übungsblatt 8 (10 Punkte)

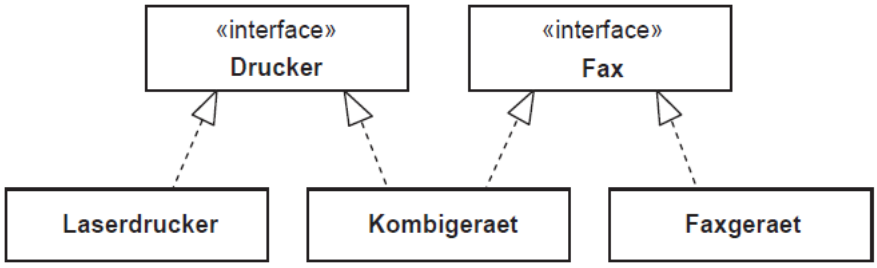
7.1 Schnittstellen implementieren (4 Punkte)

Die Klasse `Laserdrucker` soll die Schnittstelle `Drucker`, die Klasse `Faxgeraet` – die Schnittstelle `Fax` und die Klasse `Kombigeraet` – die beiden Schnittstellen `Fax` und `Drucker` implementieren. Die Schnittstellen `Fax` und `Drucker` sind gegeben durch:

```
// Datei: Fax.java
public interface Fax {
   String faxsimulation = . . .  ;
   public void senden (String sendeRef);
}

// Datei: Drucker.java
public interface Drucker {
   String drucksimulation = . . . ;
   public void drucken (String druckRef);
}
```

Das folgende Bild zeigt die Implementierung der Schnittstellen:



- Kommentieren Sie die beiden Schnittstellen in Javadoc-Format.
- Implementieren Sie die Klassen `Laserdrucker`, `Faxgeraet` und `Kombigeraet` so, dass die Klasse `TestGeraete`

```
// Datei: TestGeraete.java
public class TestGeraete {
   public static void main (String[] args) {
      Laserdrucker l1 = new Laserdrucker();
      Laserdrucker l2 = new Laserdrucker();
      Faxgeraet f1 = new Faxgeraet();
      Faxgeraet f2 = new Faxgeraet();
      Kombigeraet k1 = new Kombigeraet();
      Kombigeraet k2 = new Kombigeraet();
      f1.senden ("Dies ist ein Test");
```

```
        f2.senden ("Dies ist ein Test");
        l1.drucken ("Dies ist ein Test");
        l2.drucken ("Dies ist ein Test");
        k1.senden ("Dies ist ein Test");
        k2.senden ("Dies ist ein Test");
        k1.drucken ("Dies ist ein Test");
        k2.drucken ("Dies ist ein Test");
    }
}
```

die folgende Ausgabe erzeugt:

```
Absender ist: Fax1
Das Senden wird simuliert
Dies ist ein Test
Absender ist: Fax2
Das Senden wird simuliert
Dies ist ein Test
Drucker Laser1 meldet sich
Das Drucken wird simuliert
Dies ist ein Test
Drucker Laser2 meldet sich
Das Drucken wird simuliert
Dies ist ein Test
Absender ist: Kombigerät1
Das Senden wird simuliert
Dies ist ein Test
Absender ist: Kombigerät2
Das Senden wird simuliert
Dies ist ein Test
Kombigerät Kombigerät1 meldet sich
Das Drucken wird simuliert
Dies ist ein Test
Kombigerät Kombigerät2 meldet sich
Das Drucken wird simuliert
Dies ist ein Test
```

7.2 Interface **Vergleichbar** (6 Punkte)

Gegeben sind die Schnittstelle und die beiden nachfolgenden Klassen:

```
public interface Vergleichbar {

        /**
         * vergleicht this mit dem als Parameter uebergebenen Objekt
         * @param obj übergebenes Objekt, mit dem this verglichen wird
         * @return -1, falls this kleiner ist als das Parameterobjekt; 0, falls
            beide Objekte gleich gross sind; 1, falls this groesser ist als das
            Parameterobjekt
```

```
        */
        public abstract int vergleicheMit(Vergleichbar obj);
}

public class NuetzlicheFunktionen {

        /**
        * bestimmt ein kleinstes (auf Basis der Vergleichbar-Implementierung)
                                                Element des Parameter-Arrays
        * @param array übergebenes Array
        * @return ein kleinstes Element des übergebenen Arrays
        */
        public static Vergleichbar kleinstesElement(Vergleichbar[] array) {
                …
        }
}

public class Integer {

        private int wert;

        public Integer(int w){
                wert = w;
        }

        public int getWert() {
                return wert;
        }
}
```

- Implementieren Sie die Methode `kleinstesElement()` der Klasse `NuetzlicheFunktionen`. (Das Array kann u.U. mehrere kleinste Elemente haben.) Achten Sie auf eine robuste Implementierung und sehen Sie Warnungen vor, falls das Array noch gar keine Elemente enthält.
- Leiten Sie von der Klasse `Integer` eine Klasse `VInteger` ab, die das Interface `Vergleichbar` implementiert.
- Schreiben Sie ein kleines Testprogramm, das zunächst ein Array mit `VInteger`-Objekten erzeugt und initialisiert, anschließend die Methode `kleinstesElement()` mit diesem Array aufruft und den *Wert* des ermittelten kleinsten Elements auf den Bildschirm ausgibt.

7.3 Verschlüsselung: Zusatzaufgabe (6 Zusatzpunkte)

In dieser Aufgabe geht es um die Verschlüsselung von Texten. Klartexte über einem Klartextalphabet werden durch die Anwendung von Verschlüsselungsalgorithmen in Geheimtexte über einem Geheimtextalphabet überführt. Verschlüsselungsverfahren sollen gewährleisten, dass nur Befugte bestimmte Botschaften lesen können. Schauen Sie sich dazu das folgende Interface an:

```
interface Chiffrierung {
   public char chiffrieren(char zeichen);
   public char dechiffrieren(char zeichen);
}
```

Das Chiffrieren ist ein Verschlüsselungsverfahren, bei dem jeder Buchstabe in einem Text durch einen anderen Buchstaben ersetzt wird. Die Methode `chiffrieren()` des Interfaces soll eine entsprechende Umsetzung des übergebenen Zeichen vornehmen. Die Methode `dechiffrieren()` bildet die reverse Funktion.
Ihre Aufgabe besteht nun darin, das Interface zweimal zu implementieren:

- Bei der ersten Implementierung sollen Sie die so genannte *Caesar-Verschiebung* implementieren. Die Caesar-Verschiebung beruht auf einem Geheimtextalphabet, das um eine bestimmte Stellenzahl *n* gegenüber dem Klartextalphabet verschoben ist. Beispiel für n = 3: a -> d, b -> e, c -> f, ..., w -> z, x -> a, y -> b, z -> c. Chiffriert werden sollen hier nur Kleinbuchstaben. Alle anderen Zeichen sollen unverändert zurückgegeben werden. Implementieren Sie neben den beiden Methoden des Interfaces einen Konstruktor, dem die Stellenzahl als Parameter übergeben wird
- Bei der zweiten Implementierung des Interface übergeben Sie im Konstruktor explizit das Geheimtextalphabet als 26elementiges `char`-Array *m*. Die Chiffrierung erfolgt hierbei durch: a -> m[0], b -> m[1], ..., z -> m[25]. Chiffriert werden sollen auch hier nur Kleinbuchstaben. Alle anderen Zeichen sollen unverändert zurückgegeben werden.
- Implementieren Sie anschließend die folgende Klasse:

  ```
  public class Verschluesselung {
     public static String verschluesseln(String klartext,
                                         Chiffrierung schluessel);
     public static String entschluesseln(String geheimtext,
                                         Chiffrierung schluessel);
  }
  ```

  zum Ver- bzw. Entschlüsseln von Botschaften gemäß eines zwischen Sender und Empfänger vereinbarten Chiffrierungsschlüssels (Hinweis: Die Klasse java.lang.String stellt eine Methode char charAt(int index) zur Verfügung, die den Charakter an der index-ten Stelle liefert).
- Implementieren Sie weiterhin ein Testprogramm (= Aufruf aller Methoden) für die Klasse `Verschluesselung`, in dem beide Chiffrierungsverfahren eingesetzt werden.