Hilfestellung Teil 1

Die Aufgabe ist schwierig zu beschreiben und entsprechend sind die hier gelisteten Tipps möglicherweise nur eingeschränkt hilfreich.

- Es ist sinnvoll, die Karten nach zwei Schemas zu sortieren. Einmal nach der Kartentyp (5 identisch, 4 identisch, etc.) und dann pro Typ noch nach dem Wert der einzelnen Karten (A, K, Q, ...).
- Ein möglicher Datentyp, um all diese Dinge zu berücksichtigen, ist eine `ArrayList<ArrayList<String[]> >`. Was ist damit gemeint? Die äußere ArrayList enthält insgesamt 7 ArrayLists, welche jeweils alle Karten nach Kartentyp sortiert enthalten. Die erste innere ArrayList enthält also z. B. alle Karten, bei denen alle 5 identisch sind, etc. Pro innerer ArrayList wird ein String-Array gespeichert. Dieses stellt die beiden String-Bestandteile jeder Zeile dar. Dadurch ist gewährleistet, dass der Kartenwert jeder Karte immer fest zugeordnet ist und auffindbar bleibt (auch nach Umsortierungen).
- Die inneren String-Arrays können als erstes erstellt werden.
- Pro String-Array muss nun der Typ der Karte ermittelt werden und das String-Array der passenden ArrayList zugefügt werden.
- Am Ende muss man alle eingetragenen ArrayLists in der richtigen Reihenfolge durchlaufen, die String-Arrays pro ArrayList korrekt sortieren und alle Werte zusammenrechnen.
- Für die korrekte Sortierung wird man wiederum vermutlich nicht herumkommen einen eigenen `Comparator` zu erstellen, mit dem man eine eigene/neue Sortierreihenfolge erzwingen kann.

Dieser nachfolgende Lösungsvorschlag ist an einzelnen Stellen **nicht** besonders schön programmiert. Funktionieren tut er dennoch.

Lösungsvorschlag

```
Comparator<String[]> cardComparator = new Comparator<String[]>() {
        // Die eigene Reihenfolge definieren
        String customOrder = "AKQJT98765432"; // Hier deine gewünschte
Reihenfolge einfügen

        @Override
        public int compare(String[] arr1, String[] arr2) {
            String s1 = arr1[0];
            String s2 = arr2[0];

            int minLength = Math.min(s1.length(), s2.length());
            for (int i = 0; i < minLength; i++) {
                int index1 = customOrder.indexOf(s1.charAt(i));
                int index2 = customOrder.indexOf(s2.charAt(i));
                if (index1 != index2) {
                    return Integer.compare(index1, index2);
                }
            }
```

```
            return Integer.compare(s1.length(), s2.length());
        }
    };

private int getType(String hand) {
    boolean fiveOfAKind = false;
    boolean fourOfAKind = false;
    boolean threeOfAKind = false;
    int pairs = 0;

    char[] chars = hand.toCharArray();
    Arrays.sort(chars);

    char lastChar = chars[0];
    int counter = 1;
    for (int i = 1; i < chars.length; i++) {
        if (chars[i] != lastChar) {

            if (counter == 4) {
                fourOfAKind = true;
            } else if (counter == 3) {
                threeOfAKind = true;
            } else if (counter == 2) {
                pairs++;
            }
            lastChar = chars[i];
            counter = 1;
            continue;
        }

        counter++;
    }
    if (counter == 5) {
        fiveOfAKind = true;
    } else if (counter == 4) {
        fourOfAKind = true;
    } else if (counter == 3) {
        threeOfAKind = true;
    } else if (counter == 2) {
        pairs++;
    }

    if (fiveOfAKind) {
        return 0;
    } else if (fourOfAKind) {
        return 1;
    } else if (threeOfAKind && pairs == 1) {
        return 2;
    } else if (threeOfAKind && pairs == 0) {
```

```
        return 3;
    } else if (pairs == 2) {
        return 4;
    } else if (pairs == 1) {
        return 5;
    } else {
        return 6;
    }
}


public long partOne() {
    ArrayList<String[]> lines = new ArrayList<String[]>();

    for (String line: inputLines) {
        lines.add(line.split(" "));
    }

    // Die folgende Datenstruktur enthält 7 ArrayLists für String[] (die
einzelnen Zeilen).
    // In der ersten ArrayList werden die stärksten Kartentypen gespeichert
und in den darauffolgenden die schwächeren.
    ArrayList<ArrayList<String[]>> ranks = new
ArrayList<ArrayList<String[]>>();
    for (int i = 0; i < 7; i++) {
        ranks.add(new ArrayList<String[]>());
    }

    // sortiere die einzelnen Zeilen nach Karten-Typ-Rang. Es ist noch NICHT
nach Kartenwert innerhalb eines Kartentyps sortiert.
    for (String[] line: lines) {
        ranks.get(getType(line[0])).add(line);
    }

    long result = 0;
    int rankCounter = 1;
    for (int r = ranks.size() - 1 ; r >= 0; r--) {
        ArrayList<String[]> rank = ranks.get(r);

        rank.sort(cardComparator);
        for (int l = rank.size() - 1; l >= 0; l--) {
            String line = rank.get(l)[1];
            result += rankCounter * Integer.parseInt(line);
            rankCounter++;
        }
    }

    return result;
}
```

Hilfestellung Teil 2

Für Teil 2 benötigt man an zwei Stellen kleinere Anpassungen.

1. Bei der Erfassung der Types muss anders vorgegangen werden. Man kann die Methode `getType()` so umprogrammieren, dass sie nun die Joker J zunächst bewusst ignoriert. Anschließend nimmt man den errechneten Karten-Typ und korrigiert ihn pro vorhandenen Joker 'nach oben'. Beispiel: 4JJKQ wird zunächst exklusive der zwei J als 'High Card' klassifiziert. Nimmt man im ersten Durchlauf einen Joker hinzu, so **muss** aus einer 'High Card' immer Päärchen werden. Nimmt man im zweiten Durchlauf den zweiten Joker hinzu, so **muss** aus einem Päärchen ein Drilling werden. Das ist damit die finale Kategorie dieser Karte.
2. Der Comparator muss minimal umgeschrieben werden, indem man die Reihenfolge der Zeichen ändert.

Lösungsvorschlag

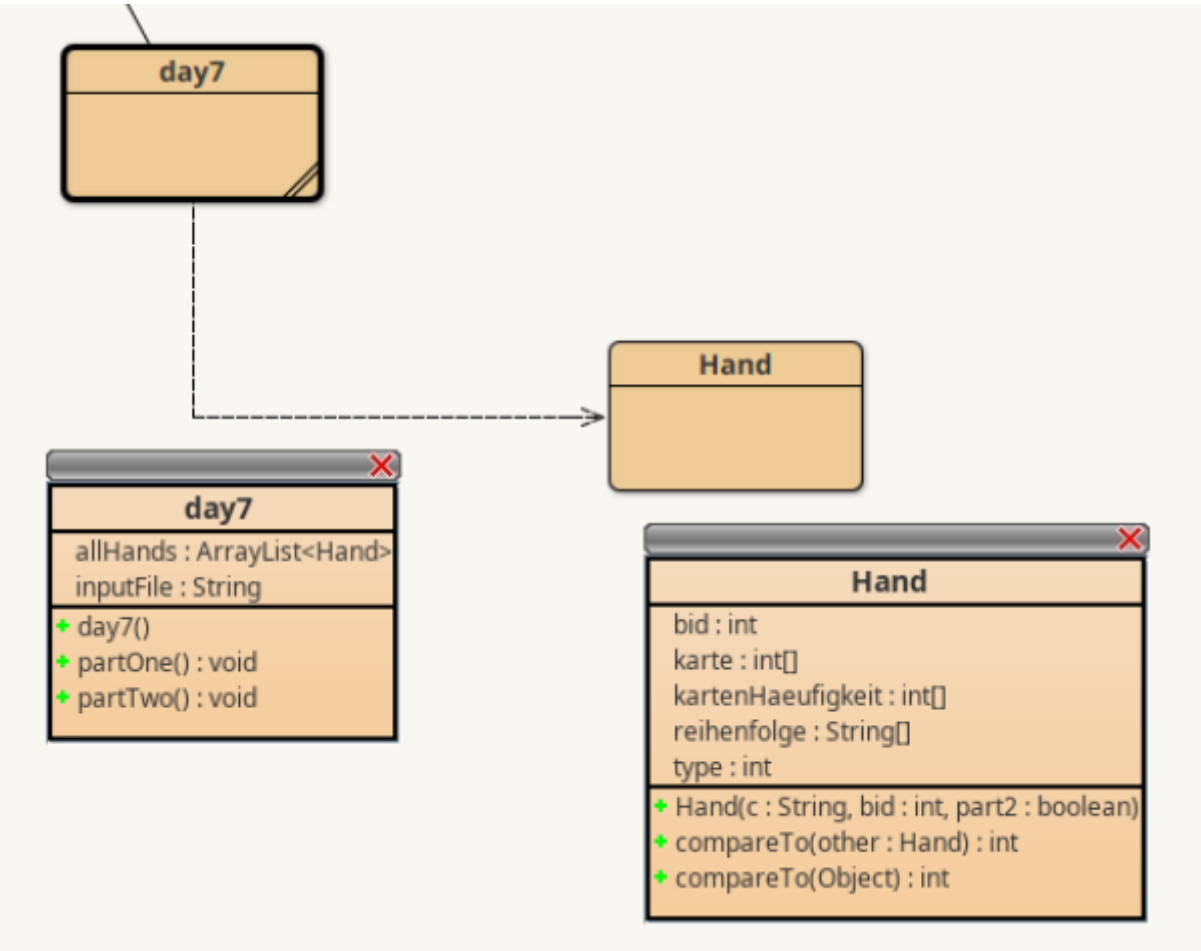
```
private int getType2(String hand) {
    boolean fiveOfAKind = false;
    boolean fourOfAKind = false;
    boolean threeOfAKind = false;
    int pairs = 0;

    char[] chars = hand.toCharArray();
    Arrays.sort(chars);

    char lastChar = chars[0];
    int counter = 1;
    for (int i = 1; i < chars.length; i++) {
        if (chars[i] == 'J') {
            continue;
        }

        if (chars[i] != lastChar) {

            if (counter == 4) {
                fourOfAKind = true;
            } else if (counter == 3) {
                threeOfAKind = true;
            } else if (counter == 2) {
                pairs++;
            }
            lastChar = chars[i];
            counter = 1;
            continue;
        }

        counter++;
    }
```

```
    if (counter == 5) {
        fiveOfAKind = true;
    } else if (counter == 4) {
        fourOfAKind = true;
    } else if (counter == 3) {
        threeOfAKind = true;
    } else if (counter == 2) {
        pairs++;
    }

    if (fiveOfAKind) {
        return 0;
    } else if (fourOfAKind) {
        return 1;
    } else if (threeOfAKind && pairs == 1) {
        return 2;
    } else if (threeOfAKind && pairs == 0) {
        return 3;
    } else if (pairs == 2) {
        return 4;
    } else if (pairs == 1) {
        return 5;
    } else {
        return 6;
    }
}

private int pimpType(int type, String card) {
    int amountOfJ = 0;
    for (int i = 0; i < card.length(); i++) {
        if (card.charAt(i) == 'J') {
            amountOfJ++;
        }
    }

    int result = type;
    for (int i = 0; i < amountOfJ; i++) {
        if (type == 0 || type == 1) {
            type = 0;
        } else if (type == 2 || type == 3) {
            type = 1;
        } else if (type == 4) {
            type = 2;
        } else if (type == 5) {
            type = 3;
        } else {
            type = 5;
        }
    }

    return type;
```

```
}

public long partTwo() {
    ArrayList<String[]> lines = new ArrayList<String[]>();

    for (String line: inputLines) {
        lines.add(line.split(" "));
    }

    // Die folgende Datenstruktur enthält 7 ArrayLists für String[] (die
einzelnen Zeilen).
    // In der ersten ArrayList werden die stärksten Kartentypen gespeichert
und in den darauffolgenden die schwächeren.
    ArrayList<ArrayList<String[]>> ranks = new
ArrayList<ArrayList<String[]>>();
    for (int i = 0; i < 7; i++) {
        ranks.add(new ArrayList<String[]>());
    }

    // sortiere die einzelnen Zeilen nach Karten-Typ-Rang. Es ist noch NICHT
nach Kartenwert innerhalb eines Kartentyps sortiert.
    for (String[] line: lines) {
        int type = getType2(line[0]);
        type = pimpType(type, line[0]);
        ranks.get(type).add(line);
    }

    long result = 0;
    int rankCounter = 1;
    for (int r = ranks.size() - 1 ; r >= 0; r--) {
        ArrayList<String[]> rank = ranks.get(r);

        rank.sort(cardComparator2);
        for (int l = rank.size() - 1; l >= 0; l--) {
            String line = rank.get(l)[1];
            result += rankCounter * Integer.parseInt(line);
            rankCounter++;
        }
    }

    return result;
}
```

Variante 2: Objekte

Variante zwei wieder mit Objekten:



Wie bei den Sortieralgorithmen hier erläutert, kann man bei Java-Klassen das "Comparable"-Interface implementieren, damit ist es möglich, in der Methode `compareTo` eigene Regeln festzulegen, nach denen die Objekte dieser Klasse sortiert werden sollen. Genau das benötigen wir hier:

- Wir erzeugen für jede Zeile in der Eingabe ein Hand-Objekt, für den wir den Typ bestimmen und in einem Attribut ablegen.
- Dann wollen wir die Objekte sortieren - allerdings reicht der Vergleich der Typen nicht aus, weil man bei zwei "Hands" desselben Typs die einzelnen Karten vergleichen soll.
- Also implementieren wir in der Klasse Hand die Methode `compareTo`, die genau das macht.

Die Lösung für Teil 1 und Teil 2 findet sich im selben Code, ich habe versucht das verständlich zu kommentieren. Wenn du nicht viel Programmiererfahrung hast, musst du wahrscheinlich auch an anderer Stelle noch nachlesen:

- Teil 1: Hand.java, day7.java
- Teil 1&2: Hand.java day7.java