Aufgabe 1: Zimmerbelegung

Team: „was ist JAVA?“

Team-ID: 00107

3. November 2017

Inhaltsverzeichnis

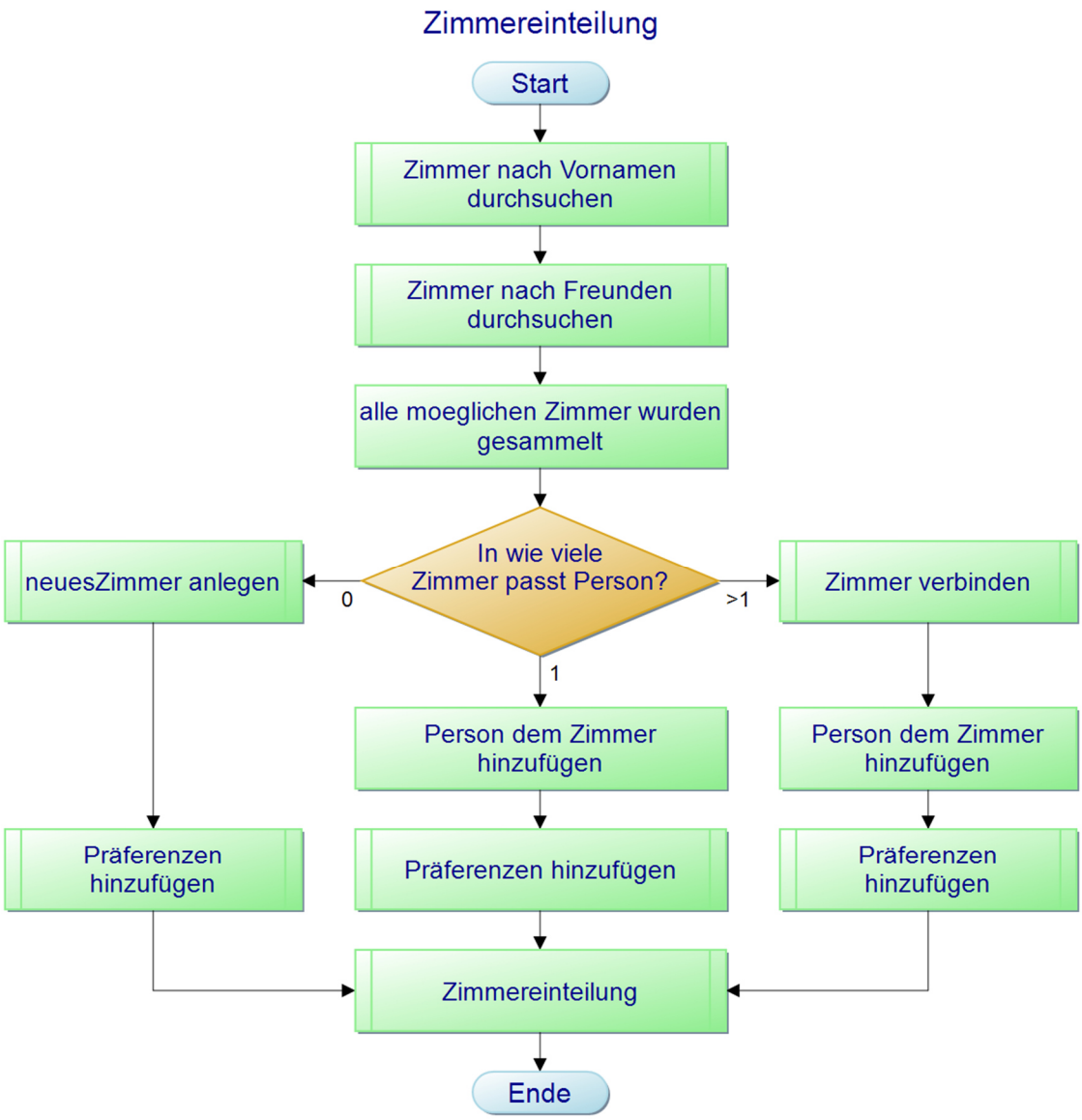
Lösungsidee

Generell werden mit unserem Algorithmus, die Kinder nacheinander in ein Zimmer eingeordnet und es bricht ab, wenn beim Einordnen an einer Stelle Präferenzen der Kinder verletzt werden. Dabei überprüfen wir zuerst, in welche Zimmer ein Kind passt, um es dann entweder einem vorhandenen Zimmer zuzuordnen oder dem Kind ein neues Zimmer zu erstellen.

Nachdem die Zimmer, in welche ein Kind passt, gezählt wurden kann es entweder sein, dass das Kind zu einem Raum dazugehört oder es passt in mehrere Zimmer. Grund dafür kann sein, dass das Kind zu unterschiedlichen Klassenkameraden in ein Zimmer will. In diesem Fall wird versucht die Zimmer zu verbinden, um den Wünschen gerecht zu werden.

Wenn dann klar ist, in welches Zimmer das Kind gehört, werden die Präferenzen hervorgehend aus den beiden Listen auf das Zimmer übertragen und das eigentliche Kind kommt natürlich auch in die Gruppe dazu. Hier wird, im Falle einer nicht möglichen Zimmerbelegung, abgebrochen, da zwei Listen sich widersprechen.

Diesen Prozess visualisiert folgender Programm-Ablauf-Plan:



Umsetzung

Die Zimmer werden als einzelne Objekte in einer ArrayList dargestellt und besitzen als Attribute jeweils selbst zwei Listen: zum einen die Zimmergenossen-Liste, welche alle Kinder, die in dem Zimmer sind als String enthält und zum anderen eine ungewollte-Zimmergenossen-Liste, welche Namen von Kindern enthält, die nicht in das Zimmer dazukommen sollen.

Um durch alle Kinder der Liste durchzulaufen, wird eine rekursive Methode genutzt, welche mit ihrem Rückgabewert angibt, ob ein Kind einem Zimmer zugeordnet werden kann oder ob es Unstimmigkeiten zwischen den gewollten und ungewollten Zimmergenossen gibt.

Diese Methode ist in drei Teile eingeteilt:

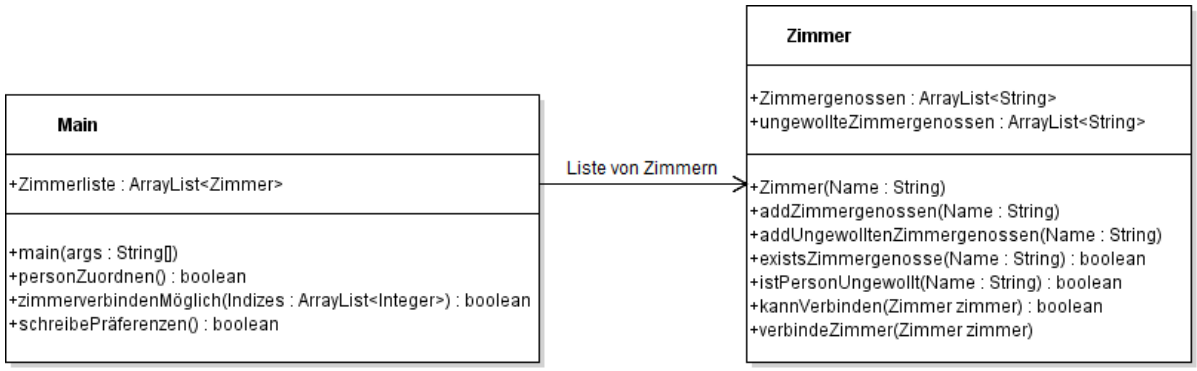
1. Zuerst wird nach Zimmern gesucht, in welche das Kind reinpasst.
    a. Hier wird eine mögliche Leerzeile übersprungen und dann zuerst nur der Name eingelesen. Nach diesem Namen wird zunächst in allen Zimmergenossen-Listen der

vorhandenen Zimmer gesucht und falls er vorkommt wird der Index des Zimmers in eine Liste mit Zimmerindizes hinzugefügt. In dieser Liste sammeln sich alle Indizes der Zimmer, in die das Kind hineinwill oder soll.

   b. Anschließend wird die Präferiertenliste des Kindes eingelesen und in allen Zimmern nach allen Namen, die auf dieser Liste stehen, gesucht. Falls auf diese Weise ein Zimmer gefunden wird bedeutet dies, dass das aktuelle Kind zu einem Kind auf seiner Präferiertenliste hinzukommen kann und der Index des Zimmers wird dann ebenfalls auf die Indexliste hinzugefügt.

2. Als nächstes wird unterschieden zwischen einem, keinem oder mehreren gefundenen Zimmern für eine Person:
   a. Falls es nur ein Zimmer ist, wird die Person diesem einfach hinzugefügt und im Anschluss wird die Methode zum Schreiben der Präferenzen eines Kindes aufgerufen.
   b. Wenn die Person in mehrere Zimmer passt, werden die Zimmergenossen-Listen aller Zimmer nach Unstimmigkeiten überprüft, um herauszufinden, ob man die Zimmer verbinden kann. Wenn das Verbinden nicht möglich ist, kann es hier auch zum Abbruch kommen, weil dann die aktuelle Person nicht nach ihren Wünschen zu Kindern, welche schon zu verschiedenen Zimmern sortiert wurden, dazukommen kann. Wenn man die Zimmer verbinden kann, werden alle Zimmergenossen der hinteren Zimmer dem zuerst gefundenen Zimmer (meistens eines, in welches das aktuelle Kind schon eingefügt wurde) hinzugefügt. Dann werden auch die ungewollte-Zimmergenossen-Listen der Zimmer zusammengefügt und die hinteren Zimmer zuletzt gelöscht.
   c. Wenn die Person zu keinem bisherigen Zimmer passt, wird ihr ein neues Zimmer erstellt. Dabei wird der Konstruktor für Zimmer mit dem Namen des Kindes aufgerufen und wieder mit schreibePräferenzen() werden die beiden Listen des Zimmers erweitert.
3. Zuletzt, wird die Methode neu aufgerufen, damit eine neue Person eingelesen wird. Da die gesamte Methode einen Wahrheitswert zurückgibt, welcher angibt, ob die nächste Person ohne Widerspruch in die Zimmerbelegung eingeordnet werden kann, wird hier bei false als Rückgabe abgebrochen.

Hier noch ein Klassendiagramm, welches die einfache Beziehung zwischen der Zimmerklasse und der Main darstellt und die wichtigen Methoden beinhaltet.



Beispiel

```
Beispielliste:

Anna
+ Berta
-

Berta
+ Anna
-


Chiara
+ Diana
-


Diana
+ Chiara Berta
-

Elisa
+
-   Anna
```

Hier ist ein Bespiel mit 5 Kindern, welche in ein Zimmer zu viert und ein Zimmer für Elisa alleine eingeordnet werden.

Zuerst wird hier Anna eingelesen, welche in keinem Zimmer zuvor vorkommt und deshalb in ein Neues kommt. Ihr Name wird in keinem Zimmer gefunden, da noch keine existieren und genauso existiert Berta noch in keinem Zimmer. Das Zimmer wird mit Anna erstellt und anschließend Berta hinzugefügt.

Nachdem die Methode neu gestartet wurde, wird dann Berta eingelesen, die in Zimmer 0 gefunden wird und durch Anna auf der + Liste noch einmal für dieses Zimmer gefunden wird. Damit gehört sie nur Zimmer 0 an und beim Übertragen der beiden Listen zu der Person wird nichts Neues hinzugefügt.

Chiara wird als nächstes eingelesen. Sie gehört noch zu keinem Zimmer und auch Diana auf der Liste kann nicht gefunden werden. Deshalb kommen beide in ein neues Zimmer. (wie bei Anna)

Diana wird zunächst in Zimmer 1 gefunden und auch Chiara gehört zu diesem Zimmer. Allerdings wird genauso Berta in einem Zimmer gefunden, sie gehört zu Zimmer 0. Deshalb wird überprüft, ob man die Zimmer verbinden kann. Dies ist in diesem Fall möglich, da Anna und Berta nichts gegen Chiara und Diana haben und umgekehrt. Beim Hinzufügen kommen Chiara und Diana zu Zimmer 0 dazu und die Ungewollten-Liste wird auch hinzugefügt. Zuletzt, muss Zimmer 1 gelöscht werden, was da die Liste eine dynamische Datenstruktur ist keine Auswirkung auf ggf. mehr Zimmer hinter 1 hat.

Elisa wird zuletzt eingelesen und kann zu keinem anderen Zimmer dazukommen. Ihr wird ein neues Zimmer erstellt und Anna auf die Ungewollten-Liste geschrieben.

Die Rekursion endet beim Lesen der nächsten Zeile, welche als Null erkannt wird. Dann wird true zurückgegeben und die Methode beendet. Zuletzt wird in der Main alles ausgegeben und in eine Datei geschrieben.

Quellcode

```
import java.io.*;
import java.util.ArrayList;

public class ZimmerbelegungMain2{

  private static ArrayList<Zimmer> zimmerliste = new ArrayList<Zimmer>();
  private static File fileIn = new File("zimmerbelegung.txt");
  private static FileReader fr;
  private static BufferedReader reader;
  private static File fileOut = new File("zimmerbelegung_loesung.txt");
  private static FileWriter fw;
  private static BufferedWriter writer;

  public static void main(String[] args) {

    try {
      fr = new FileReader(fileIn);
      reader = new BufferedReader(fr);
      fw = new FileWriter(fileOut);
      writer = new BufferedWriter(fw);

      if (checkPersonGrouping()) {
        System.out.println("Es funktioniert!");
        for (int i = 0; i < zimmerliste.size(); i++) {
          System.out.println("Das " + i + ". Zimmer: ");
          writer.write("Zimmer " + (i+1) + ":" +
System.getProperty("line.separator"));
          zimmerliste.get(i).writeZimmergenossen(writer);
        }
      } else {
        System.out.println("Diese Schueler koennen sich auf keine Zimmerbelegung
einigen!");
        writer.write("Die Kinder können sich auf keine Zimmerbelegung einigen!");
      }

      reader.close();
      writer.close();
    } catch (IOException e) {
      System.err.println("Die Datei " + fileIn + " konnte nicht bearbeitet
werden!");
    }

  }

  private static boolean checkPersonGrouping() {
    String line;
    try {
      line = reader.readLine();
      if (line == null) {
        System.out.println("Die Datei ist fertig gelesen!");
        return true;
      }
      if (line.equals("")) {
        line = reader.readLine();
      }

      String name = line;
      System.out.println("Eine neue Person: " + name);

      ArrayList<Integer> zimmertemp = new ArrayList<Integer>();
      for (int i = 0; i < zimmerliste.size(); i++) {
        // Ist die Person schon in einem Zimmer?
        if (zimmerliste.get(i).existsZimmergenosse(name)) {
          System.out.println("Die Person " + name + " gehoert Zimmer " + i + "
an.(Meth1)");
```

Für die Zimmereinteilung wird java.io für das Lesen und Schreiben von Dateien und java.util für die Liste als dynamische Datenstruktur importiert.

Hier werden dann der Reader und der Writer erstellt und dazugehörig das File. Auch die Zimmerliste wird erstellt.

Reader und Writer werden fertig zu einem BufferedReader und BufferedWriter.

Die rekursive Methode wird aufgerufen und wenn es true zurückgibt wird die Zimmerbelegung ausgegeben.

Falls es nicht klappt, wird eine Ausgabe getätigt.

Reader und Writer werden geschlossen

Eine Exception, welche beim Lesen passieren kann, wird abgefangen.

Die rekursive Methode:

```
          zimmertemp.add(i);
        }
      }

      line = reader.readLine();
      line = line.substring(2); // + weggekürzt
      if (!line.isEmpty()) {
        String[] temp2 = line.split(" ");
        for (int i = 0; i < zimmerliste.size(); i++) {
          for (int j = 0; j < temp2.length; j++) {
            if (zimmerliste.get(i).existsZimmergenosse(temp2[j])) {
              if (zimmerliste.get(i).istPersonUngewollt(name)) {
                System.out.println(
                "Zimmer " + i + " mag nicht, dass " + temp2[j] + " zu ihnen
kommt!");
                return false;
              } else {
                System.out.println("Person " + name + " kann in Zimmer  " + i + "
dazukommen.(Meth2)");
                boolean zimmerExists = false;
                for (int k = 0; k < zimmertemp.size(); k++) {
                  if (i == zimmertemp.get(k)) {
                    zimmerExists = true;
                  }
                }
                if (!zimmerExists) {
                  zimmertemp.add(i);
                }
              }
            }
          }
        }
      }
      if (zimmertemp.size() == 1) {
        if (!zimmerliste.get(zimmertemp.get(0)).existsZimmergenosse(name)) {
          zimmerliste.get(zimmertemp.get(0)).addZimmergenossen(name);
          System.out.println("Dem Zimmer " + zimmertemp.get(0) + " wird Person " +
name + " hinzugefügt.");
        }
        if (!writePreferences(line, zimmertemp.get(0))) {
          // Die Präferenzen werden geschrieben: Fehler
          System.out.println("Es funktioniert nicht (Person einfuegen1) !");
          return false;
        }
      } else if (zimmertemp.size() > 1) {
        // kann man diese Zimmer verbinden?
        if (!zimmerverbindenMoeglich(zimmertemp)) {
          System.out.println("Zimmer verbinden nicht möglich.");
          return false;
        }

        // man kann die Zimmer verbinden
        for(int i = 1; i < zimmertemp.size(); i++) {

zimmerliste.get(zimmertemp.get(0)).verbindeZimmer(zimmerliste.get(zimmertemp.get(i
)));
          System.out.println(zimmertemp.get(i) + " wurde dem Zimmer " +
zimmertemp.get(0) + " hinzugefügt.");
          zimmerliste.remove(zimmertemp.get(i));
          System.out.println(zimmertemp.get(i) + " wurde gelöscht.");
          zimmertemp.remove(i);
        }
        if (!zimmerliste.get(zimmertemp.get(0)).existsZimmergenosse(name)) {
          zimmerliste.get(zimmertemp.get(0)).addZimmergenossen(name);
          System.out.println("Dem Zimmer " + zimmertemp.get(0) + " wird Person " +
name + " hinzugefügt.");
        }
        if (!writePreferences(line, zimmertemp.get(0))) {
          // Die Präferenzen werden geschrieben: Fehler
          System.out.println("Es funktioniert nicht (Person einfuegen1) !");
          return false;
        }
```

Nächste Zeile wird eingelesen und falls diese Null ist, ist die Datei fertig gelesen und es wird abgebrochen.

Die zuerst gelesene Zeile ist der Name des Kindes

Alle Zimmer werden nach dem Namen des Kindes durchsucht und der Zimmerindex einer Liste hinzugefügt.

Die nächste Zeile wird eingelesen, welche die Kinder enthält, zu denen das aktuelle Kind dazukommen möchte.

Es wird nach einem Zimmer gesucht, in welchem eine Freundin ist, damit das Kind diesem Zimmer hinzukommen kann.

Nur falls dieses Zimmer noch nicht erkannt wurde, wird der Index der Liste hinzugefügt.

```
    } else {
      // ein neues Zimmer wird angelegt
      System.out.println("Ein neues Zimmer mit " + name);
      zimmerliste.add(new Zimmer(name));
      if (!writePreferences(line, zimmerliste.size() - 1)) {
        // Die Präferenzen werden geschrieben: Fehler
        System.out.println("Es funktioniert nicht (Zimmer erstellen) !");
        return false;
      }
    }

    if (!checkPersonGrouping()) {
      // falls die nächste Zeile eine Ungereimtheit findet
      System.out.println("Es funktioniert nicht (Neue Person)!");
      return false;
    }

  } catch (IOException e) {
    System.err.println("Fehler beim Bearbeiten der Namenszeile!");
    e.printStackTrace();
  }
  return true;
}



/**
* Diese Funktion ueberprueft, ob sich mehrere Zimmer verbinden lassen,
* damit den Wuenschen der Kinder entsprochen wird
*/
private static boolean zimmerverbindenMoeglich(ArrayList<Integer> zimmerNr) {
  for (int i = 0; i < zimmerNr.size(); i++) {
    for (int j = 0; j < zimmerNr.size(); j++) {
      if (i != j) {
        if (!zimmerliste.get(zimmerNr.get(i))
        .kannVerbinden(zimmerliste.get(zimmerNr.get(j)))) {
          System.out.println("Es funktioniert nicht (Zimmer verbinden) !");
          return false;
        }
      }
    }
  }
  return true;
}



/**
* Diese Funktion überprueft die Uebereinstimmung einer Person mit einem
* Zimmer und ergänzt die Listen des angegebenen Zimmers mit den Listen der
* letzten Person
*
* @param lastLine
*            ist entweder die Präferiertenliste oder null
* @param zimmerNr
*            ist die Nummer des zu überprüfenden Zimmers
* @return returns true, wenn das Hinzufügen in das Zimmer möglich ist
*/
private static boolean writePreferences(String lastLine, int zimmerNr) {
  try {
    String prefLine = lastLine;
    System.out.println(prefLine);
    if (prefLine.startsWith("+"))
      prefLine = prefLine.substring(2);
    if (!prefLine.isEmpty()) {
      String[] temp = prefLine.split(" ");
      for (int i = 0; i < temp.length; i++) {
        // hier wird die Wunschliste ruebergeschrieben

        if (zimmerliste.get(zimmerNr).istPersonUngewollt(temp[i])) {
          System.out.println("In das Zimmer kann nicht: " + temp[i]);
          return false;
        } else if (!zimmerliste.get(zimmerNr).existsZimmergenosse(temp[i])) {
          zimmerliste.get(zimmerNr).addZimmergenossen(temp[i]);
          System.out.println("Neue Person in dem Zimmer: " + temp[i]);
```

Falls das Kind in nur ein Zimmer passt, wird es diesem hinzugefügt.

Auch die Präferenzen werden auf das Zimmer übertragen. Falls es dabei einen Fehler gibt, wird alles abgebrochen.

Falls es mehrere Zimmer für das Kind gibt, wird zuerst geschaut, ob man diese Zimmer verbinden kann. Wenn es nicht geht, wird abgebrochen.

Dann werden alle Zimmer ab Index 1 dem 0. Zimmer hinzugefügt und danach gelöscht.

Die Person wird dann dem ersten Zimmer hinzugefügt.

Und die Präferenzen werden übertragen.

Zuletzt gibt es die Möglichkeit, dass ein Kind noch zu keinem Zimmer gehört und dann wird ein neues Zimmer erstellt.

Hier passiert die Rekursion,

```
          }
        }
      }
      System.out.println("Die Preferiertenliste wurde geschrieben!");
      String dislikeLine = reader.readLine();
      System.out.println(dislikeLine);
      dislikeLine = dislikeLine.substring(2);
      if (!dislikeLine.isEmpty()) {
        String[] temp = dislikeLine.split(" ");
        for (int i = 0; i < temp.length; i++) {
          // hier wird die Ungewunschtenliste ruebergeschrieben
          if (zimmerliste.get(zimmerNr).existsZimmergenosse(temp[i])) {
            System.out.println("Person mag nicht: " + temp[i] + " aus dem
Zimmer");
            return false;
          } else if (!zimmerliste.get(zimmerNr).istPersonUngewollt(temp[i])) {
            zimmerliste.get(zimmerNr).addUngewolltenZimmergenossen(temp[i]);
            System.out.println("ungewuenschte Person im Zimmer" + zimmerNr + ": "
+ temp[i]);
          }
        }
      }

    } catch (IOException e) {
      System.err.println("Fehler beim Schreiben der Preferierten-
/Ungewuenschtenzeile!");
      e.printStackTrace();
    }
    return true;
  }

}
```

indem die Methode für das Lesen einer neuen Person aufgerufen wird. Falls es nicht klappt wird false zurückgegeben.

Eine Input/ Output – Exception wird abgefangen

Ist es möglich, mehrere Zimmer aus einer Indexliste zu verbinden? Der Rückgabewert gibt dies an.

Diese Methode schreibt die Präferenzen einer Person auf die Listen des Zimmers zu dem das Kind hinzukommt.

Eine Person wird dem Zimmer hinzugefügt, falls sie nicht schon existiert oder garnicht in das Zimmer soll.

	Genauso werden ungewünschte Personen auf das Zimmer übertragen, falls sie nicht Zimmergenossen sind.

Lösungen der Aufgaben

Aufgabe 1:

Paula

Person mag nicht Anna aus Zimmer 0!

Die Kinder können sich auf keine Zimmerbelegung einigen!

Aufgabe 2:

Zimmer 1:

Alina, Lilli

Zimmer 2:

Emma, Mia, Zoe

Zimmer 3:

Lara

Aufgabe 3:

Zimmer 1:

Hannah

Zimmer 2:

Lea, Celine, Lena, Lina, Clara

Zimmer 3:

Sarah, Sophie, Alina, Josephine, Vanessa, Leonie, Lilli, Pia, Annika, Melina, Kim, Katharina, Pauline

Zimmer 4:

Laura, Charlotte, Lara, Jasmin, Merle, Celina, Nina, Miriam, Luisa, Jessika

Zimmer 5:

Nele, Michelle, Antonia, Anna, Julia, Lisa, Emily, Sofia, Marie, Jana, Johanna, Carolin Emma, Larissa

Aufgabe 4:

Zimmer 1:

Anna, Julia, Miriam, Jessika, Jasmin, Pauline

Zimmer 2:

Hannah, Celine, Annika, Pia, Lisa, Carolin, Celina, Lara, Nele, Michelle, Charlotte, Emma, Luisa, Nina, Clara, Kim, Lina, Merle, Vanessa

Zimmer 3:

Sarah, Marie, Johanna, Lea, Laura, Katharina, Leonie, Larissa, Alina, Jana, Josephine, Lilli, Lena, Melina, Emily, Sophie, Sofia

Zimmer 4:

Antonia

Aufgabe 5:

Marie

Person mag nicht: Marie aus dem Zimmer

Die Kinder können sich auf keine Zimmerbelegung einigen!

Aufgabe 6:

Zimmer 1:

Anna, Carolin, Clara, Lena, Nina, Antonia, Leonie, Lina, Kim, Lisa, Nele, Luisa, Marie, Julia, Laura, Katharina, Pauline, Emma, Miriam, Lara, Jessika, Alina, Charlotte, Josephine, Pia

Zimmer 2:

Lea, Michelle

Zimmer 3:

Annika, Celina

Zimmer 4:

Jasmin, Merle, Sofia

Zimmer 5:

Celine, Larissa, Hannah, Sophie, Sarah, Vanessa, Jana, Johanna, Melina, Lilli, Emily