Aufgabe 3:

```
6       public class Person
7       {
            16 references
8           public string FirstName;
            17 references
9           public string LastName;
            15 references
10          public DateTime DateOfBirth;
11
            8 references
12          public Person Mom;
            8 references
13          public Person Dad;
14      }
15
```

In der class Person sehen wir sehen wir das diese Klasse rekursiv ist (selbstaufrufend). Jedes Objekt vom Typ Person, referenziert jeweils die Objekte Mom und Dad von Typ Person.

Mom und Dad sind also beide jeweils so strukturiert, wie das Objekt vom Typ Person.

Als nächstes wird ein Breakpoint in Zeile 19 erstellt und wir lassen den Debugger laufen.

```
19      public static Person Find(Person person)
20      {
```

Hierbei Untersuchen wir der Person root:

```
Locals
  args [string[]]: {string[0]}
  root: {Debugging.Person}
    Dad: {Debugging.Person}
      Dad: {Debugging.Person}
        Dad: {Debugging.Person}
          Dad [Person]: null
          DateOfBirth [DateTime]: {01.02.1882 00:00:00}
          FirstName [string]: "Andi"
          LastName [string]: "ElGreco"
          Mom [Person]: null
        DateOfBirth [DateTime]: {10.06.1921 00:00:00}
        FirstName [string]: "Philip"
        LastName [string]: "Battenberg"
        Mom: {Debugging.Person}
      DateOfBirth [DateTime]: {14.11.1948 00:00:00}
      FirstName [string]: "Charlie"
      LastName [string]: "Wales"
      Mom: {Debugging.Person}
    DateOfBirth [DateTime]: {21.07.1982 00:00:00}
    FirstName [string]: "Willi"
    LastName [string]: "Cambridge"
    Mom: {Debugging.Person}
  found: {Debugging.Person}
```

Hier sieht man nun alle Eigenschaften welche die Person root hat, hierzu gehören auch die Einträge von Mom und Dad. Wenn man diese auch wieder weiter aufklappt, sieht man die Inhalte von deren Eltern. Wir können hier nun den gesamten Stammbaum mit seinen einzelnen Elementen untersuchen.

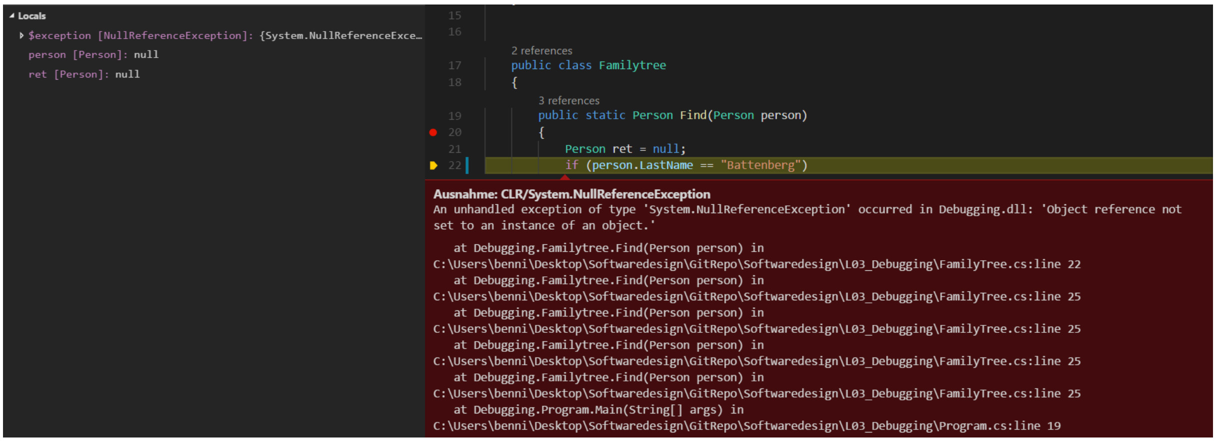
```
public static Person Find(Person person)
{
    Person ret = null;
    if (person.LastName != "Battenberg")
        return person;

    ret = Find(person.Mom);
    if (ret != null)
        return ret;
    ret = Find(person.Dad);
    return ret;
}
```

Nun untersuchen wir die **Find** Methode. Diese läuft nun rekursiv durch den Baum durch und Untersucht ob die gegebene Bedingung (also ob der Nachname ungleich Battenberg) erfüllt ist und gibt die erste Person, welche diese Bedingung erfüllt nun aus. (In unserem Fall Willi Cambridge)

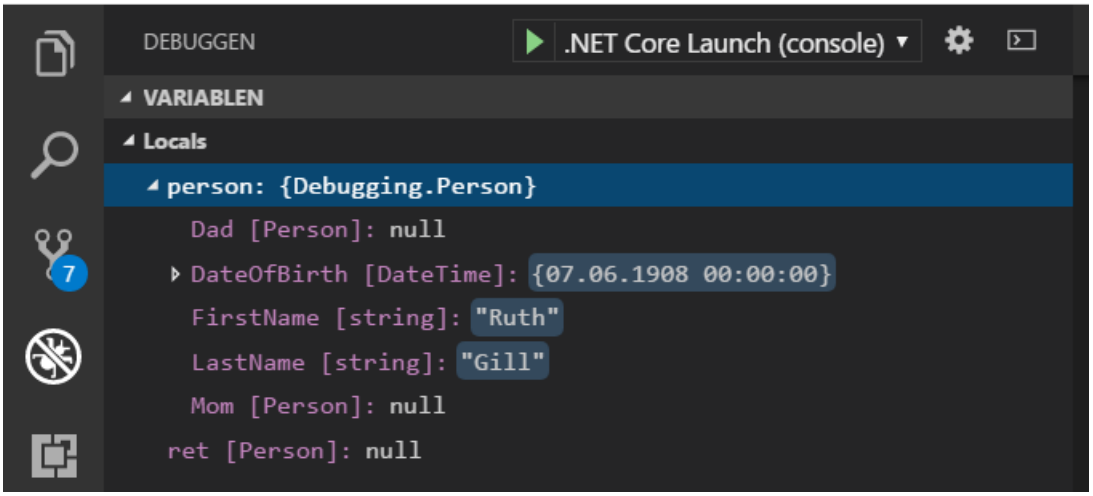
```
public static Person Find(Person person)
{
    Person ret = null;
    if (person.LastName == "Battenberg")
        return person;

    ret = Find(person.Mom);
    if (ret != null)
        return ret;
    ret = Find(person.Dad);
    return ret;
}
```

Eine Möglichkeit, den Bug zu fixen ist es zum Beispiel die Bedingung mit Battenberg von != auf == zu setzen.

Hier haben wir nun aber das Problem dass es beim vierten Personendurchlauf eine Null Exception gibt.



Grund für diese NullException ist, dass bei durchlauf für Ruth Gill keine Person für Mom mehr hinterlegt ist. Jedoch will das Programm immer nach einem Personendurchlauf mit der Mum der Person weiter suchen.



Lösung für das Problem ist, das man einfach eine if Abfrage einfügt, welche jeweils überprüft, das die Find Funktion nicht mit einem Parameter Arbeitet, welcher den Typen Null hat.

```
19          public static Person Find(Person person)
20          {
21              Person ret = null;
22              if (person.LastName == "Battenberg")
23                  return person;
24              if (person.Mom != null)
25                  ret = Find(person.Mom);
26              if (ret != null)
27                  return ret;
28              if (person.Dad != null)
29                  ret = Find(person.Dad);
30              return ret;
31          }
```