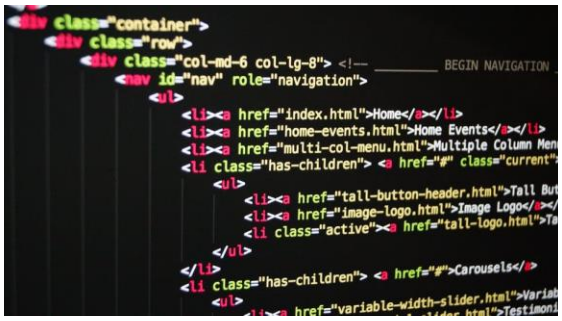


NETbuilder Assessment

This document contains a set of tasks to be completed in your preferred programming language (Java or Javascript). Once complete please submit your code and documentation (if required) to Kevin Short at QA as soon as you finish. Immedaite questions can be directed to Bob Harvey at NETbuilder Digital (bob.harvey@netbuilder.com)

Task 1 – should be between 30mins - 1hr

Task 2 – a longer assisgment (half a day)

Please do your best to complete the tasks in the advised timeframe. Note: there is no one solution to each task, so you will be judged on a well engineered approach, rather than a full completion.

Alphabet Position Replacement

Given a string, replace every letter with its position in the alphabet. If anything in the text isn’t a letter, ignore it and don’t return it.

A is 1, B is 2, C is 3, etc.

As an example:

```
replaceLetterWithPosition(“This NETbuilder assessment is way too easy.”)
```

Should return:

```
“20 8 9 19 14 5 20 2 21 9 12 4 5 18 1 19 19 5 19 19 13 5 14 20 9 19 23 1 25
20 15 15 5 1 19 25”
```

Family tree task on next page

Family Tree

We need a system that can learn facts about family relationships, check their consistency and answer queries about them.

The task

Create a class `Family` with the following methods. All arguments are strings: names of persons. Upon the first use of a name, that name is added to the family.

- `male(name)` and `female(name)` returning `boolean`

  Define the gender (corresponding to the method name) of the given person. Return `false` when these assignments cannot be made because of conflicts with earlier registered information.

- `isMale(name)` and `isFemale(name)` returning `boolean`

  Return `true` when the person has the said gender. When no gender was assigned, both methods should return `false`

- `setParent(childName, parentName)` returning `boolean`

  Defines the child-parent relationship between two persons. Returns `false` when the relationship cannot be made because of conflicts with earlier registered information.

- `getParents(name)` and `getChildren(name)` returning `array` of `string`

  Return the names of the person's parents/children in alphabetical order

Deducing information

When things can be implied from given information, it should be done.

For instance, a parent's gender can be determined as soon as the other parent's gender becomes known:

```
const fam = new Family();
fam.setParentOf("Vera", "George");
fam.setParentOf("Vera", "Vanessa");
fam.female("Vanessa");
fam.female("George"); // false, because:
fam.isMale("George"); // ...this is true.
```

Also `setParentOf` can return `false`. For example, if the relationship would infer that one becomes their own ancestor:

```
fam = new Family();
fam.setParentOf("Vera", "George");
fam.setParentOf("George", "Vera"); // false
```

Details, rules, assumptions

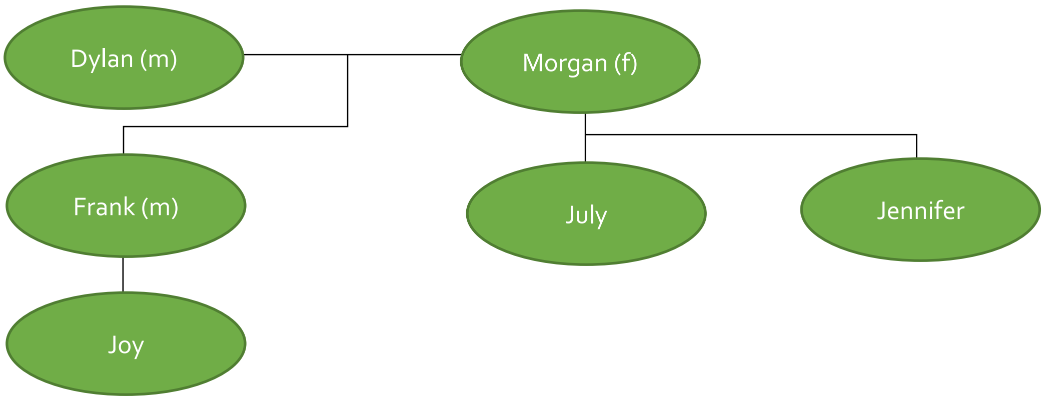
Although the task relates to genealogy, the rules of this task are not claimed to be realistic. Several simplifications and rules apply, which may not hold in real life:

- Strings are case sensitive, but there are no tests playing around with "Peter", "PETER" and "PeTeR".
- People are uniquely identified by their name. For instance, there are no two different people called "Jim" in the same family.
- Once a person has an assigned gender, it cannot be changed.
- No gender conclusions should be made from personal names: "Bob" could well be a woman and "Susan" a man.
- People cannot have more than one mother and one father.
- The terms "parents" and "children" refer to the relatives in the immediate previous/next generations only, not to more remote ancestors or descendants.
- Incest may occur, so, for example, one's parent may at the same time be their grandparent.
- One cannot be their own ancestor.
- Age is not accounted for. Even if some incestuous relationships would infer that one's parent is more than 5 generations older, it should be allowed.
- In case a name's first occurrence is in a call of one of the two gender querying methods, the return value will always be false, as that new person does not have a known gender.
- In case a name's first occurrence is in a call of one of the two relation querying methods, the return value will always be an empty array/list, as there are no relationships known yet in which that new person participates.
- For the reasons in the preceding two bullet points it should not matter whether you actually store that name in these cases in your data structure, or not. In the latter case you would only store it at the next occasion when that name is mentioned in a call of one of the three other methods, that actually add information. The described interface has no way to query the difference between these two possible implementations, so you can choose freely.

Family tree task cont. on next page

Example

Consider the following family tree:



It could be created step by step with the following code — the expected return value for each method call is indicated in comments:

```
const fam = new Family();
fam.setParentOf("Frank", "Morgan");         // true
fam.setParentOf("Frank", "Dylan");          // true
fam.male("Dylan");                          // true
fam.setParentOf("Joy", "Frank");            // true
fam.male("Frank");                          // true
fam.male("Morgan");                         // false
// (Morgan is a woman because she both is Frank's parent, but not his
father)
fam.setParentOf("July", "Morgan");          // true
// (The preceding assertion was rejected, so there is no conflict)
fam.isMale("Joy") || fam.isFemale("Joy");  // false
// (We know Joy is Frank's child, but we can't derive Joy's gender)
fam.getChildrenOf("Morgan");                // ["Frank", "July"]
fam.setParentOf("Jennifer", "Morgan");      // true
fam.getChildrenOf("Morgan");                // ["Frank", "Jennifer", "July"]
fam.getChildrenOf("Dylan");                 // ["Frank"]
// (That is all we know for sure)
fam.getParentsOf("Frank");                  // ["Dylan", "Morgan"]
fam.setParentOf("Morgan", "Frank");         // false
// (It is impossible to be the parent of your parent)
```