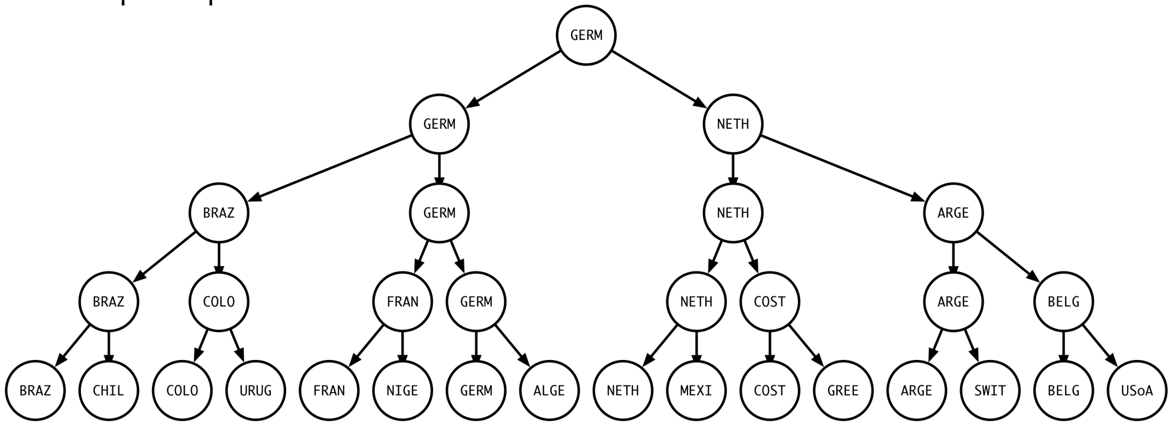
Bracket Challenge
COSC 3327 Algorithms and Data Structures
Due: See Canvas
Late Deadline: See Canvas

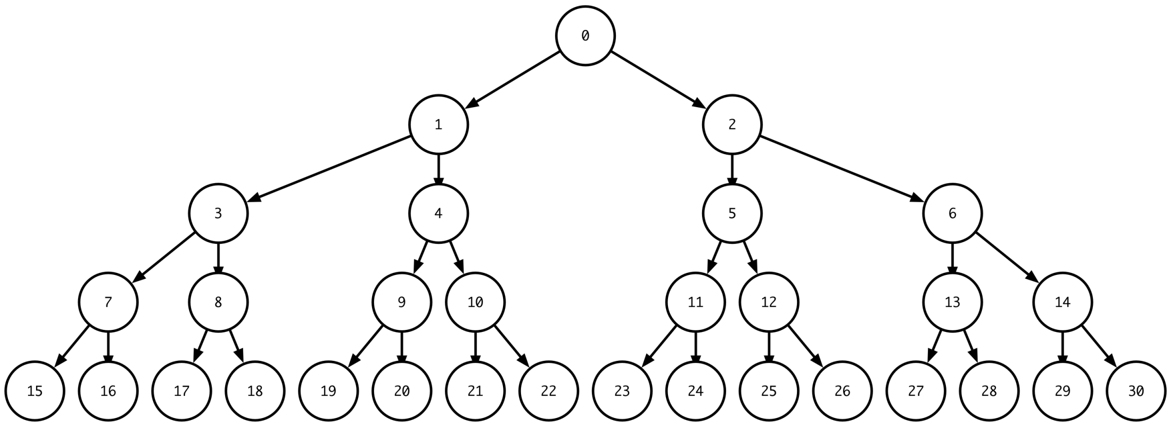
Background

Tournaments (or parts of tournaments) often have a bracket structure, especially when the tournament is "single-elimination." For instance, here are the results of the 2014 World Cup soccer championship:



Internal Representation

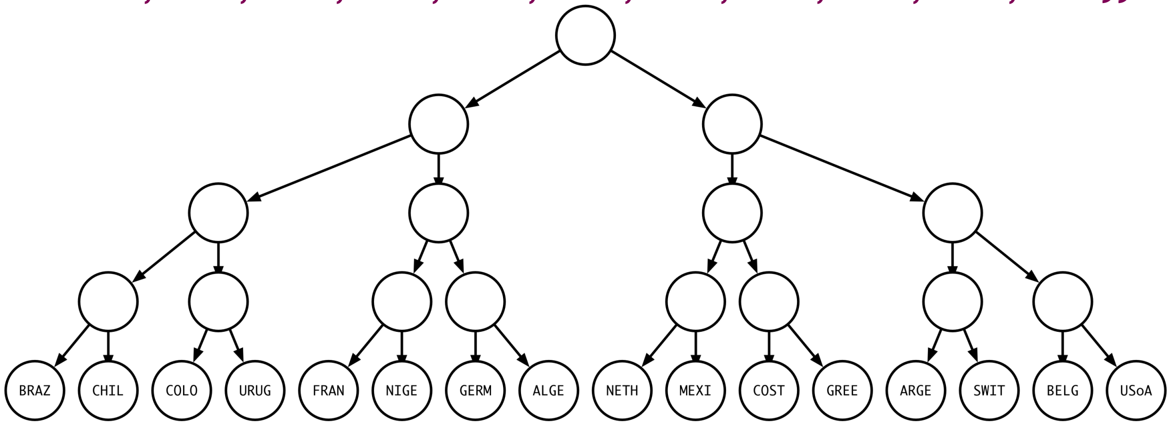
Your job is to model bracket predictions by maintaining a list of participants and predictions as follows:



Behavior

Constructor call [note that * below indicates a null in the List]:

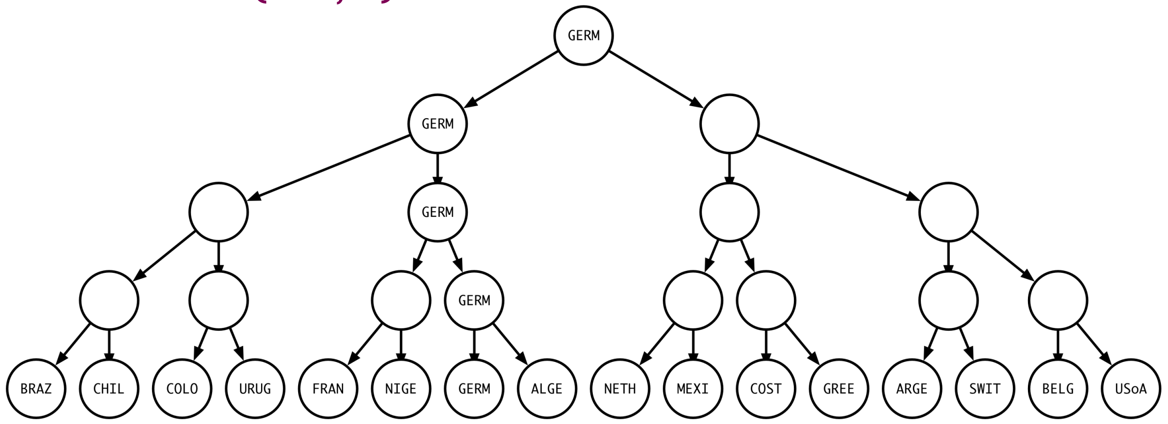
A. `new BracketImpl_LastName(Arrays.asList(BRAZ, CHIL, COLO, URUG, FRAN, NIGE, GERM, ALGE, NETH, MEXI, COST, GREE, ARGE, SWIT, BELG, USoA))`



```
[*, *, *, *, *, *, *, *, *, *, *, *, *, *, *, BRAZ, CHIL, COLO, URUG, FRAN,
NIGE, GERM, ALGE, NETH, MEXI, COST, GREE, ARGE, SWIT, BELG, USoA]
```

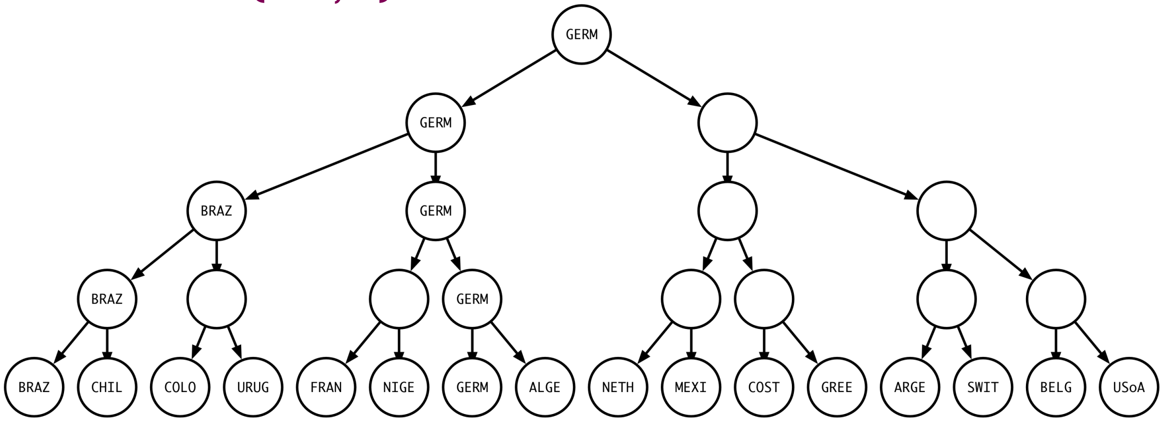
```
getMaxLevel() --> 4
getGroupings(0) --> {{BRAZ}, {GREE}, {GERM}, {NETH}, {COST}, {COLO}, {BELG}, {MEXI},
{NIGE}, {FRAN}, {URUG}, {SWIT}, {ARGE}, {USoA}, {CHIL}, {ALGE}}
getGroupings(1) --> {{BRAZ, CHIL}, {NIGE, FRAN}, {GREE, COST}, {MEXI, NETH}, {SWIT,
ARGE}, {COLO, URUG}, {BELG, USoA}, {GERM, ALGE}}
getGroupings(2) --> {{BELG, SWIT, ARGE, USoA}, {NIGE, FRAN, GERM, ALGE}, {BRAZ,
COLO, URUG, CHIL}, {GREE, MEXI, NETH, COST}}
getGroupings(3) --> {{GREE, BELG, MEXI, SWIT, NETH, ARGE, USoA, COST}, {BRAZ, COLO,
NIGE, FRAN, URUG, GERM, CHIL, ALGE}}
getGroupings(4) --> {{BRAZ, GREE, GERM, NETH, COST, COLO, BELG, MEXI, NIGE, FRAN,
URUG, SWIT, ARGE, USoA, CHIL, ALGE}}
```

B. `setWinCount(GERM, 4):`



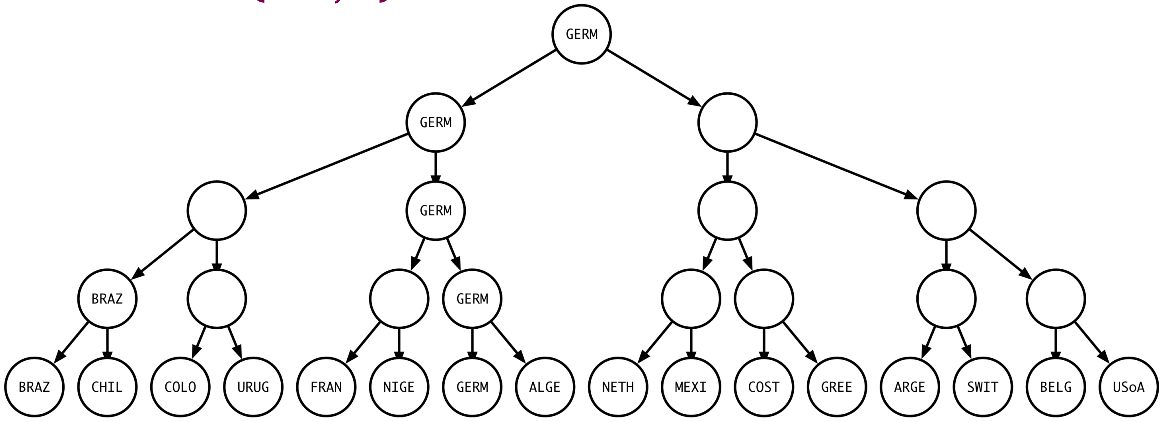
```
[GERM, GERM, *, *, GERM, *, *, *, *, *, GERM, *, *, *, *, BRAZ, CHIL, COLO,
URUG, FRAN, NIGE, GERM, ALGE, NETH, MEXI, COST, GREE, ARGE, SWIT, BELG,
USoA]
```

C. `setWinCount(BRAZ, 2):`



```
[GERM, GERM, *, BRAZ, GERM, *, *, BRAZ, *, *, GERM, *, *, *, *, BRAZ, CHIL,
COLO, URUG, FRAN, NIGE, GERM, ALGE, NETH, MEXI, COST, GREE, ARGE, SWIT,
BELG, USoA]
```

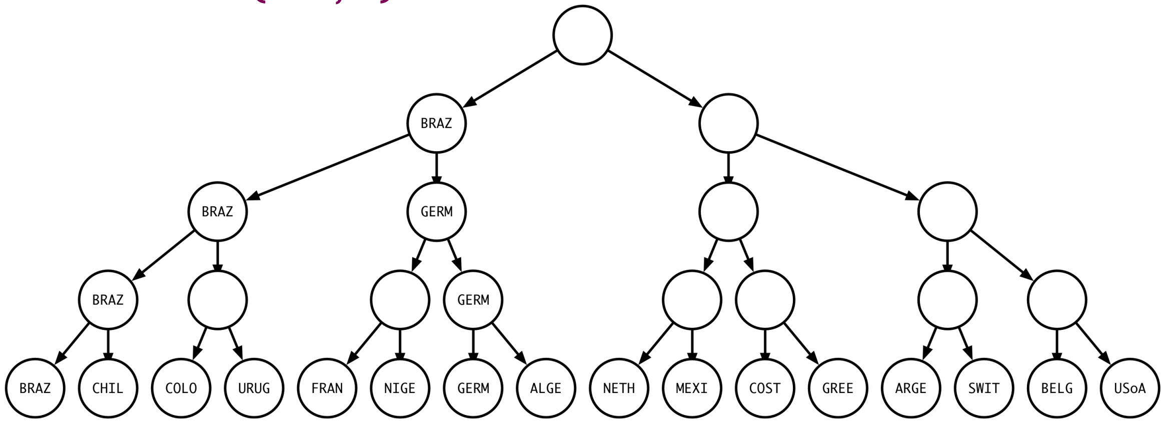
D. `setWinCount(BRAZ, 1):`



```
[GERM, GERM, *, *, GERM, *, *, BRAZ, *, *, GERM, *, *, *, *, BRAZ, CHIL,
COLO, URUG, FRAN, NIGE, GERM, ALGE, NETH, MEXI, COST, GREE, ARGE, SWIT,
BELG, USoA]
```

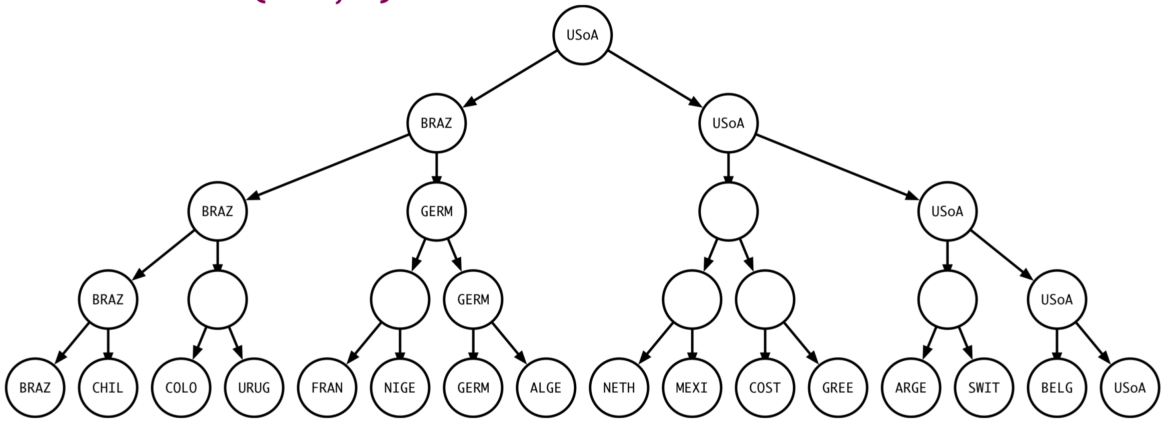
```
getViableParticipants({URUG}) --> {URUG}

getViableParticipants({BRAZ, CHIL}) --> {BRAZ}

getViableParticipants({BRAZ, CHIL, COLO, URUG}) --> {BRAZ, COLO, URUG}

getViableParticipants({FRAN, NIGE, GERM, ALGE}) --> {GERM}

getViableParticipants({NETH, MEXI, COST, GREE}) -->
                                                  {NETH, MEXI, COST, GREE}

getViableParticipants({BRAZ, CHIL, COLO, URUG, FRAN, NIGE, GERM, ALGE}) -->
                                                  {GERM}
```

E. `setWinCount(BRAZ, 3):`



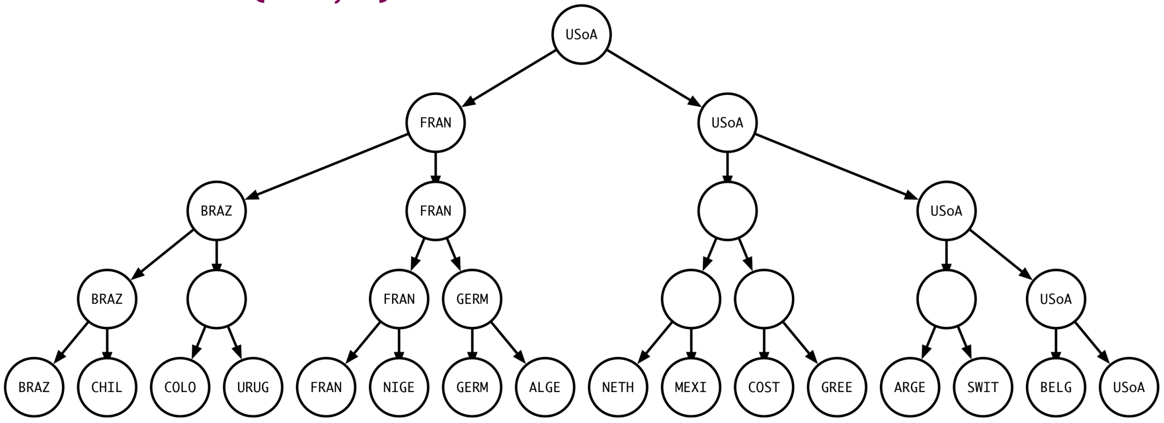
```
[*, BRAZ, *, BRAZ, GERM, *, *, BRAZ, *, *, GERM, *, *, *, *, BRAZ, CHIL,
COLO, URUG, FRAN, NIGE, GERM, ALGE, NETH, MEXI, COST, GREE, ARGE, SWIT,
BELG, USoA]
```

F. `setWinCount(USoA, 4):`



```
[USoA, BRAZ, USoA, BRAZ, GERM, *, USoA, BRAZ, *, *, GERM, *, *, *, USoA,
BRAZ, CHIL, COLO, URUG, FRAN, NIGE, GERM, ALGE, NETH, MEXI, COST, GREE,
ARGE, SWIT, BELG, USoA]
```

G. `setWinCount(FRAN, 3):`



`[USoA, FRAN, USoA, BRAZ, FRAN, *, USoA, BRAZ, *, FRAN, GERM, *, *, *, USoA, BRAZ, CHIL, COLO, URUG, FRAN, NIGE, GERM, ALGE, NETH, MEXI, COST, GREE, ARGE, SWIT, BELG, USoA]`

Tips

- Make a "tournament of Integers" with test cases to more easily debug that participants are being placed at the correct indices in the predictions list.
- Notice that a call to `setWinCount()` potentially requires updates to `getMaxLevel()` nodes. In particular, notice that `setWinCount(FRAN, 3)` above required changing three specific nodes (corresponding to list elements with indices 19, 9, 4, 1) to FRAN and checking that USoA could remain in the top-most node (corresponding to the list element with index 0).

Deliverables

Upload to Canvas:

- `BracketImpl_`*`LastName`* (which extends `BracketAbstract` ← **`do not upload`**).
- Any other supporting classes.

Constraints

- Do not rename `BracketAbstract`.
- You must properly maintain the `predictions` list according to the internal representation above. It is extremely important to store participants at exactly the right indices.
- Make your preconditions as executable as possible; proper precondition violation handling will be tested.
- Don't add any instance variables to your Impl; use only the `List predictions`, which is inherited from `BracketAbstract`.
- Ensure that I will be able to compile and run your code.
- Ensure that your Impl class and any supporting classes that you create are named with the LastName.java suffix and are in the package named 'tournament'.
- Ensure that you use the interfaces provided above without changing the interface name, package, or method signatures. Do not add any additional methods to any of interfaces.

Code snippets

(see starter kit for full preconditions).

```
package tournament;

//This Bracket concept is not conscious of regions
public interface Bracket<P>
{
    public int getMaxLevel();
    public Set<Set<P>> getGroupings(int level);
    public Set<P> getViableParticipants(Set<P> grouping);
    public void setWinCount(P participant,
                                        int exactWinCount);
    public boolean equals(Object obj);
}
```

```
package tournament;

public abstract class BracketAbstract<P> implements Bracket<P>
{
    ...
    //See STARTER_KIT
}
```

```
package tournament;

//All soccer teams that have played in the World Cup
//(up to and including the 2014 World Cup)
public enum FIFASoccerTeam
{
    ALGERIA, ANGOLA, ARGENTINA, AUSTRALIA, AUSTRIA,
    BELGIUM, BOLIVIA, BOSNIA_HERZEGOVINA, BRAZIL, BULGARIA,
    ...
    //See STARTER_KIT for complete file
}
```