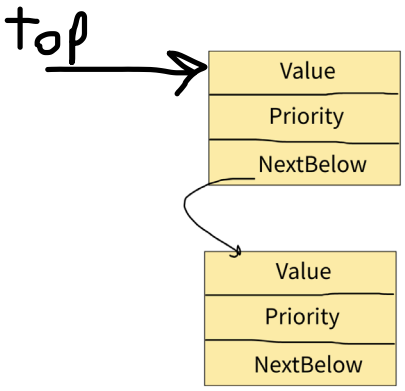
Generic Priority Stack

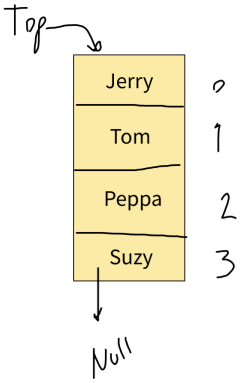
Your task is to complete the implementation of the class below.

```
// You are not allowed to use complex data structures such as ArrayList, HashMap, etc.
// You are also not allowed to use Collections and Arrays *classes*.
// You are allowed to use arrays, of course.
// Only submit your own original work.

public class PriorityStack<T> {

   class Container<T> {

      T value;
      boolean hasPriority;
      Container<T> nextBelow;
   }

   private Container<T> top; // top of the stack element

   private int size;

   public void push(T value) {

   }

   public void push(T value, boolean hasPriority) {

   }

   public T pop() {
      // remove and return the top item
      // if no item found (size == 0) then throw NoSuchElementException

   }

   public T popPriority() {
      // find item with priority starting from the top, remove it and return it
      // if no priority item found then remove and return the top item
      // if stack is empty then throw NoSuchElementException

      return pop();
   }
```

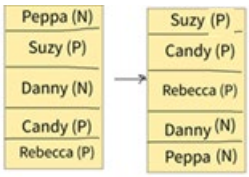
```
    public int hasValue(T value) {
        // returns -1 if value is not on the stack
        // this code only looks for the *first* occurence of the value, starting from top
        // WARNING: you must call value.equals(item.value) to determine whether
        // two values are equal, just like you would do for a String
        // returning value 0 means the value is on top of the stack,
        // 1 means 1 below the top, and so on...

        return -1; // not found
    }

    public T removeValue(T value) {
        // removes the first item from top containing the value and returns the value
        // if item with value is not found throw NoSuchElementException

        throw new NoSuchElementException();
    }

    public int getSize() {
        return size;
    }

    public void reorderByPriority() {
        // reorder items (re-create a new stack, if you like)
        // where all priority items are on top and non-priority items are below them
        // Note: order within the priority items group and non-priority items group must remain
the same
        // Suggestion: instead of reordering the existing stack items
        // it may be easier to re-create a new stack with items in the order you need

    }

    @Override
    public String toString() {
        // return string describing the contents of the stack, starting from the top
        // Use value.toString() to convert values kept in the stack to strings.
        // Format exactly like this (assuming T is a string to keep it simple):
        // "[Jerry:N,Terry:N,Martha:P,Tom:P,Jimmy:N]"
        // N means item has no priority, P means item has priority
        // For full marks you must use StringBuilder, no + (string concatenation) allowed.

    }
```

```
public T[] toArrayReversed(Class<T> type) { // Note: this is "the twist"
   // return array with items on the stack
   // WARNING: element 0 of the array must be the BOTTOM of the stack
    // Collect items on your way back, just before returning.
}

// NOTE: you are only allowed to add private methods and private fields (if needed)
```





hasValue("Tom")=1



Reorder