Domain: $\mathcal{B}$ = {b | b is a Bag with elements of the type TElem}

Interface (set of operations):

init(b)

pre : true

post: b ∊ $\mathcal{B}$, b is an empty Bag

add(b, e)

pre: b ∊ $\mathcal{B}$, e ∊ TElem

post: b’ ∊ $\mathcal{B}$, b’ = b U {e} (Telem *e* is added to the Bag)

remove(b, e)

pre: b ∊ $\mathcal{B}$, e ∊ TElem

post: b’ ∊ $\mathcal{B}$, b’ = b \ {e} (one ocurrence of e was removed from the Bag).

$$\text{remove} \leftarrow \begin{cases} \text{true, if an element was removed } (\text{size}(b') < size(b)) \\ \text{false, if e was not present in b } (\text{size}(b') = \text{size}(b)) \end{cases}$$

search(b, e)

pre: b ∊ $\mathcal{B}$, e ∊ TElem

post: $search \leftarrow \begin{cases} true, if\ e \in \mathbf{B} \\ false, otherwise \end{cases}$

size(b)

pre: b ∊ $\mathcal{B}$

post: size ← $the\ number\ of\ elements\ from\ b$

nrOccurrences(b, e)

pre: b ∊ $\mathcal{B}$, e ∊ Telem

post: $nrOccurrences \leftarrow the\ number\ of\ occurrences\ of\ e\ in\ b$

destroy(b)

pre: b ∊ $\mathcal{B}$

post: b was destroyed

iterator(b, i)

pre: b ∊ $\mathcal{B}$

post: i ∊ $\mathcal{I}$, i is an iterator over b

```
def createIntBag():
    intBag = Bag()
    intBag.add(6)
    return intBag


def main():

    @ b1 = createIntBag()
    print("Number of occurrences for 6: ", b1.nrOccurrences(6))
    @ b2 = createStringBag()
    print("Number of occurrences for data:", b2.nrOccurrences("data"))
```

ADT Iterator

Has access to the interior structure of the Bag and it has a current element from the Bag.

Domain: $\mathcal{I}$ = {i | i is an iterator over b ∊ $\mathcal{B}$ }

Interface:

init(i, b)

pre: b ∊ $\mathcal{B}$

post: i ∊ $\mathcal{I}$, i is an iterator over b. i refers to the first element of b, or it is invalid if b is empty

valid(i)

pre: i ∊ $\mathcal{I}$

post: $valid \leftarrow \begin{cases} true, if\ the\ current\ element\ from\ i\ is\ a\ valid\ one \\ false, otherwise \end{cases}$

first(i)

pre: i ∊ $\mathcal{I}$

post: i’ ∊ $\mathcal{I}$, the current element from i’ refers to the first element from the bag or i’ is invalid if the bag is empty

next(i)

pre: i ∊ $\mathcal{I}$, valid(i)

post: i’ ∊ $\mathcal{I}$, the current element from i’ refers to the next element from the bag

throws: exception if i is not valid

getCurrent(i)

pre: i ∊ $\mathcal{I}$, valid(i)

post: getCurrent ∊ TElem, getCurrent is the current element from i

throws: exception if i is not valid

```
def printBag(bag):

    it = bag.iterator()
    while it.valid():
        print(it.getCurrent())
        it.next()
    print("Over. Let's start again")
    it.first()
    while it.valid():
        print(it.getCurrent())
        it.next()
```