

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

Interface (set of operations):

init(b)

pre : true

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

add(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

post: $b' \in \mathcal{B}$, $b' = b \cup \{e\}$ (TElem e is added to the Bag)

remove(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

post: $b' \in \mathcal{B}$, $b' = b \setminus \{e\}$ (one occurrence of e was removed from the Bag).

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

search(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

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

size(b)

pre: $b \in \mathcal{B}$

post: $\text{size} \leftarrow \text{the number of elements from } b$

nrOccurrences(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

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

destroy(b)

pre: $b \in \mathcal{B}$

post: b was destroyed

iterator(b, i)

pre: $b \in \mathcal{B}$

post: $i \in \mathcal{I}$, 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: $\mathcal{I} = \{i \mid i \text{ is an iterator over } b \in \mathcal{B}\}$

Interface:

init(i, b)

pre: $b \in \mathcal{B}$

post: $i \in \mathcal{I}$, 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: $i \in \mathcal{I}$

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

first(i)

pre: $i \in \mathcal{I}$

post: $i' \in \mathcal{I}$, 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: $i \in \mathcal{I}$, $\text{valid}(i)$

post: $i' \in \mathcal{I}$, the current element from i' refers to the next element from the bag
throws: exception if i is not valid

getCurrent(i)

pre: $i \in \mathcal{I}$, $\text{valid}(i)$

post: getCurrent $\in \text{TElem}$, 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()
```

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

Interface (set of operations):

init(b)

pre : true

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

add(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

post: $b' \in \mathcal{B}$, $b' = b \cup \{e\}$ (TElem e is added to the Bag)

remove(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

post: $b' \in \mathcal{B}$, $b' = b \setminus \{e\}$ (one occurrence of e was removed from the Bag).

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

search(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

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

size(b)

pre: $b \in \mathcal{B}$

post: $\text{size} \leftarrow \text{the number of elements from } b$

nrOccurrences(b, e)

pre: $b \in \mathcal{B}$, $e \in \text{TElem}$

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

destroy(b)

pre: $b \in \mathcal{B}$

post: b was destroyed

iterator(b, i)

pre: $b \in \mathcal{B}$

post: $i \in \mathcal{I}$, 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: $\mathcal{I} = \{i \mid i \text{ is an iterator over } b \in \mathcal{B}\}$

Interface:

init(i, b)

pre: $b \in \mathcal{B}$

post: $i \in \mathcal{I}$, 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: $i \in \mathcal{I}$

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

first(i)

pre: $i \in \mathcal{I}$

post: $i' \in \mathcal{I}$, 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: $i \in \mathcal{I}$, $\text{valid}(i)$

post: $i' \in \mathcal{I}$, the current element from i' refers to the next element from the bag
throws: exception if i is not valid

getCurrent(i)

pre: $i \in \mathcal{I}$, $\text{valid}(i)$

post: $\text{getCurrent} \in \text{TElem}$, 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()
```