| SWO 3x | Übung zu Softwareentwicklung mit klassischen Sprachen u. Bibliotheken 3 | WS 2014/15, Übung 07 |
|---|---|---|

**Abgabetermin: Sa in der KW 04**

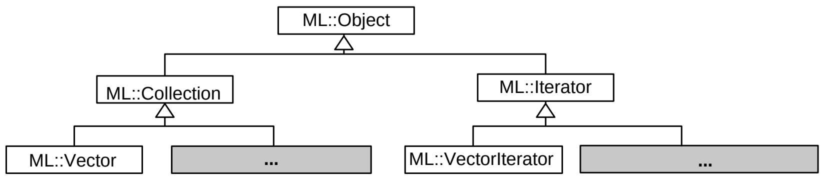
☐ **Gr. 1,** DI Franz Gruber-Leitner **Name** Roman Lumetsberger **Aufwand in h** 7

☒ **Gr. 2,** Dr. Erik Pitzer

**Punkte** ________ **Kurzzeichen Tutor / Übungsleiter** ______ / ______

## MiniLib

Studieren Sie die abstrakten Basisklassen *ML::Collection* und *ML::Iterator*. Derzeit gibt es zwei davon abgeleitete konkrete Klassen *ML::Vector* und *ML::VectorIterator*, die im Wesentlichen ein dynamisches Feld realisieren. Die Klassenhierarchie hat derzeit somit folgendes Aussehen:



## 1. Objektmengen (*sets*) (12 Punkte)

Realisieren Sie auf Basis der beiden Klassen *ML::Collection* und *ML::Iterator* zwei neue Klassen zur Repräsentation von Mengen (engl. *sets*): Entwerfen Sie eine Klasse *Set* und eine Klasse *SetIterator*. Die Klasse *Set* soll Mengen von Objekten verwalten können. Wie bei Mengen üblich, darf kein Objekt mehrmals in einem *Set* vorkommen. Überlegen Sie, welche Methoden notwendig sind, und vergessen Sie dabei nicht, auch Methoden zur Bildung der Schnittmenge, der Vereinigung und der Differenz zweier Mengen zu realisieren. Die Klasse *SetIterator* soll das "Iterieren" über die Elemente eines *Set*s ermöglichen.

Können Sie die Klassen implementieren ohne Änderungen an der MiniLib vorzunehmen? Wenn nein, welche Änderungen sind notwendig?

Dokumentieren Sie Ihren Entwurf, und zeigen Sie auf, welche Möglichkeiten sich Ihnen geboten haben, und warum Sie sich für die von Ihnen gewählte entschieden haben.

Implementieren Sie die entworfenen Klassen mit allen Methoden und testen Sie diese ausführlich. Wenn Sie Änderungen an der MiniLib vornehmen mussten, geben Sie bitte nur die geänderten Teile ab und heben Sie die Änderungen im Ausdruck hervor (unterstreichen, Leuchtmarker, ...).

## 2. Objektbehälter (*bags*) (12 Punkte)

Entwerfen Sie eine neuerliche Erweiterung der MiniLib zur Repräsentation eines allgemeinen "Objektbehälters" (engl. *bag*). Eine neue Klasse *Bag* (gemeinsam mit einer neuen Iteratorklasse *BagIterator*) soll eine Sammlung von Objekten realisieren, auf welche die gleichen Operationen wie für *Sets* anwendbar sind. Objekte der Klasse *Bag* unterscheiden sich aber von Objekten der Klasse *Set* darin, dass in *Bag*s Objekte mehrmals vorkommen dürfen. Implementieren Sie die von Ihnen entworfenen Klassen und achten Sie dabei auf eine möglichst platzsparende Speicherung von mehrfach vorkommenden Objekten. Vergessen Sie nicht, Ihre Entwurfsentscheidungen, eventuelle Probleme und Einschränkungen zu dokumentieren. Testen Sie Ihre Erweiterungen der Klassenbibliothek.

# 1 Aufgabe 1 - Objektmengen (sets)

## 1.1 Anmerkungen

Diese Aufgabe wird mit der *minilib* umgesetzt und dadurch werden alle Methoden mit einem großen Anfangsbuchstaben benannt, da dies in der minilib Konvention ist.

## 1.2 Lösungsidee

Ein *Set* ist im Grunde eine Liste, die keine mehrfachen Elemente enthalten darf. Darum wird die Klasse *Set* als Ableitung der Klasse *List*, die wir in der Übung erstellt haben, umgesetzt.

Es wird also eine doppelt verkettete Liste verwendet, die als Element die Klasse *Node* hat. Diese Klasse enthält dann das eigentliche Datenobjekt.
Es braucht dann nur die Methode Add überschrieben werden und um die Prüfung, ob ein Objekt bereits in der Liste vorhanden ist, erweitert werden. Diese Prüfung wird mit Hilfe der minilib Methode **IsEqualTo** realisiert.

### 1.2.1 Vereinigung - Union

Die Vereinigung zweier Mengen enthält alle Elemente der ersten und alle Elemente der zweiten Menge. Wichtig ist, dass jedes gleiche Element nur einmal in der Vereinigungsmenge vorkommt.

### 1.2.2 Schnittmenge - Intersect

Die Schnittmenge zweier Mengen enthält alle Elemente, die sowohl in der ersten, als auch in der zweiten Menge vorhanden sind.

### 1.2.3 Differenz - Difference

Die Differenz zweier Mengen enthält jene Elemente der ersten, aber nicht in der zweiten Menge vorkommen.

## 1.3 Entwurfsentscheidungen

Die Lösung dieser Aufgabe erfordert grundsätzlich keine Änderung der minilib, da beim Einfügen eines bereits im *Set* enthaltenen Objektes eine Fehlermeldung auf *cerr* ausgegeben wird. Zugegeben, das ist nicht die beste Lösung, doch in der kurzen Zeit von nur 1 Woche für die gesamte Übung, habe ich diese Variante gewählt.
Weiters kann der Verwender der Klasse mit *Contains* prüfen, ob das Element bereits vorhanden ist.

### 1.3.1 Weitere Lösungsmöglichkeiten

Folgende Lösungen wären auch möglich gewesen.

- Änderung des Rückgabeparameters der Methode *Collection::Add* von *void* auf *bool*, um den Aufrufer mitzuteilen, ob das Objekt eingefügt wurde oder nicht.
- Änderung des Rückgabeparameters der Methode *Collection::Add* von *void* auf *Object **. Der Rückgabewert würde dann das bereits enthaltene Objekt, falls es bereits vorhanden war, oder eben das einzufügende Objekt liefern.
  Damit könnte der Aufrufer unterscheiden, ob es bereits vorhanden war oder nicht und bekäme auch noch eine Referenz auf das im Set befindliche Element.

### 1.3.2 Implementierung Union, Intersect, Difference

Da in der Angabe nicht konkret erwähnt wird, wie die Signatur der Methoden auszusehen haben, wurden sie so umgesetzt, dass sie als Rückgabewert ein neues *Set* liefern, dass dann das Ergebnis der Operation beinhaltet.
Dies hat den Vorteil, dass das originale *Set* nicht verändert wird und somit weiterverwendet werden

kann.
**Dabei werden die Datenelemente aber nicht kopiert, d.h. der Aufrufer darf nicht *DeleteElements* auf das Original und Ergebnis anwenden.**

#### 1.3.3 SetIterator

Laut Angabe wird eine eigene Klasse *SetIterator* verlangt. Dieser wurde **nicht implementiert**, da das Set von *List* abgeleitet wurde und der dort vorhandene *ListIterator* bereits alle Anforderungen erfüllt, um die Elemente zu durchlaufen.

# 2 Aufgabe 2 - Objektbehälter (bags)

## 2.1 Anmerkungen

Auch diese Aufgabe wird mit der *minilib* umgesetzt und dadurch werden alle Methoden mit einem großen Anfangsbuchstaben benannt, da dies in der minilib Konvention ist.

## 2.2 Lösungsidee

Ein Bag ist ein Set, indem die Elemente mehrfach vorkommen können.
Diese Klasse wird durch ableiten der Klasse *Set* realisiert. Dabei wird auch ein neuer Knotentyp *BagNode* von *Node* abgeleitet und durch eine weitere Datenkomponente *count* erweitert. Dies wurde so gemacht, da gefordert ist, dass die mehrfachen Elemente möglichst platzsparend gespeichert werden sollen.

Folgende Methoden müssen überschrieben werden, um die Klasse korrekt zu implementieren:

- Add: Mehrfache Elemente zulassen und inkrementieren der Datenkomponente *count*.
- Remove: Dekrementieren von *count* oder löschen des Knoten ( bei count = 0 ).

#### 2.2.1 BagIterator

Der BagIterator muss für diese Aufgabe implementiert werden, da dieser die mehrfachen Elemente auch mehrfach liefern muss. Der *ListIterator* würde nämlich jedes Objekt nur einmal liefern, da er nur die Knoten durchläuft.

#### 2.2.2 Vereinigung - Union

Die Vereinigung zweier *Bags* enthält alle Elemente der ersten und alle Elemente der zweiten Menge.

#### 2.2.3 Schnittmenge - Intersect

Die Schnittmenge zweier *Bags* enthält alle Elemente, die sowohl in der ersten, als auch in der zweiten Menge vorhanden sind. Sollte ein Element mehrfach vorkommen, dann wird das Minimum genommen.

#### 2.2.4 Differenz - Difference

Die Differenz zweier *Bags* wird als A ohne B implementiert. Dabei wird die Anzahl der Vorkommen in B von der Anzahl in A subtrahiert. Kommt das Element in B nicht vor, dann wird die Anzahl von A ins Ergebnis übernommen.

## 2.3 Entwurfsentscheidungen

Die Methoden *Union*, *Intersect*, *Difference* wurden in *Bag* mit der oben angegebenen Logik neu implementiert. Hier könnte man sich vielleicht überlegen, diese in Collection zu definieren, damit sie für alle Collections anwendbar sind.

## 2.4 Sourcecode

### List.h

```cpp
/*****************************************************************************
  List.h
  Roman Lumetsberger

  Header for class List, ListIterator
******************************************************************************/
#ifndef LIST_H
#define LIST_H

#include <MLCollection.h>
#include "Node.h"

namespace ML {

class List : public Collection
{
 protected:
    Node *head;
    int size;

    virtual Node *Find(Object *o) const;
    //Method used for creating the list node
    virtual Node *CreateNode(Object *o) const;

  public:
    List();
    virtual ~List();

    virtual int Size() const;
    virtual void Add(Object *o);
    virtual Object *Remove(Object *o);
    virtual bool Contains(Object *o) const;
     //clears collection, does not delete elements
    virtual void Clear();
    virtual Iterator *NewIterator() const;

};

class ListIterator : public Iterator {
  //alloe NewIterator to call the private constructor
  friend Iterator *List::NewIterator() const;

  private:
    Node *current;
  private:
    ListIterator(Node *head);

  public:
    virtual ~ListIterator();
    Object *Next();
};

}
#endif // LIST_H
```

### Node.h

```
/*****************************************************************************
  Node.h
  Roman Lumetsberger

  Header for class Node
*****************************************************************************/
#ifndef NODE_H
#define NODE_H

#include <string>
#include <MLObject.h>

namespace ML {

class Node : public ML::Object
{
  public:
    Object *value;
    Node *prev, *next;

    Node(Object *value = nullptr,
         Node *prev = nullptr,
         Node *next = nullptr);

    virtual ~Node();

    virtual std::string AsString() const;
};

}

#endif // NODE_H
```

### Set.h

```
/*****************************************************************************
  Set.h
  Roman Lumetsberger

  Header for class Set
*****************************************************************************/
#ifndef SET_H
#define SET_H

#include <MLCollection.h>
#include "List.h"

namespace ML  {

class Set : public List
{
  public:
    Set();
    virtual ~Set();
    virtual void Add(Object *o) override;

    virtual Set *Union(Set *other) const;
    virtual Set *Intersect(Set *other) const;
    virtual Set *Difference(Set *other) const;
};

}
#endif // SET_H
```

**BagNode.h**

```
/*******************************************************************************
  BagNode.h
  Roman Lumetsberger

  Header for class BagNode
*******************************************************************************/
#ifndef BAGNODE_H
#define BAGNODE_H

#include "Node.h"

namespace ML {

class BagNode : public Node
{
  protected:

  public:
    int count; //public, because only used in bag

    BagNode(Object *value = nullptr,
      BagNode *prev = nullptr,
      BagNode *next = nullptr);
    virtual ~BagNode();

    std::string AsString() const override;

};
}
#endif // BAGNODE_H
```

**Bag.h**

```
/*******************************************************************************
  Bag.h
  Roman Lumetsberger

  Header for class Bag, BagIterator
*******************************************************************************/
#ifndef BAG_H
#define BAG_H

#include "BagNode.h"
```

```cpp
#include "Set.h"

namespace ML {

class Bag : public Set
{
  protected:
    virtual BagNode *Find(Object *o) const override;
    virtual BagNode *CreateNode(Object *o) const override;
  public:
    Bag();
    virtual ~Bag();

    virtual void Add(Object *o) override;
    virtual Object *Remove(Object *o) override;
    //Needed, because Collection would delete objects twice
    virtual void DeleteElements() override;

    virtual Bag *Union(Bag *other) const;
    virtual Bag *Intersect(Bag *other) const;
    virtual Bag *Difference(Bag *other) const;

    virtual Iterator *NewIterator() const;
};

class BagIterator : public Iterator {
  //allow NewIterator to call the private constructor
  friend Iterator *Bag::NewIterator() const;

  private:
    BagNode *current;
    int currentNodeCount;

  private:
    BagIterator(BagNode *head);

  public:
    virtual ~BagIterator();
    Object *Next();
};


}
#endif // BAG_H
```

**List.cpp**

```cpp
/*****************************************************************************
  List.cpp
  Roman Lumetsberger

  Implementation for class List, ListIterator
*****************************************************************************/
#include "List.h"
#include <cassert>
namespace ML {

List::List() : head(nullptr), size(0) {
  Register("List","Collection");
}

List::~List() {
  Clear();
}

int List::Size() const {
  return size;
}

Node *List::Find(Object *o) const {
  assert(o != nullptr);
  Node *cur = head;
  while(cur != nullptr &&
        !o->IsEqualTo(cur->value)) {
    cur = cur->next;
  }
  return cur;
}

Node *List::CreateNode(Object *o) const {
  return new Node(o);
}

Object *List::Remove(Object* o) {
  assert(o != nullptr);
  Node *n = Find(o);

  if(n == nullptr) return nullptr;
  if(n == head) {
    head = head->next;
  }

  if(n->prev != nullptr) n->prev->next = n->next;
  if(n->next != nullptr) n->next->prev = n->prev;

  size--;
  Object *value = n->value;
  delete n;
  return value;
}

bool List::Contains(Object* o) const {
  return Find(o) != nullptr;
}

void List::Clear() {
  Node * current = head;
  while(current != nullptr) {
    Node *tmp =current;
    current = current->next;
    delete tmp;
  }
  head = nullptr;
  size = 0;
}

```

```cpp
Iterator *List::NewIterator() const {
  return new ListIterator(head);
}

void List::Add(Object* o) {
  assert(o != nullptr);
  Node *n = CreateNode(o);
  size++;

  if( head == nullptr) {
    head = n;
  }
  else {
    Node *last = head;
    while(last->next != nullptr) {
      last = last->next;
    }

    last->next = n;
    n->prev = last;
  }
}

ListIterator::ListIterator(Node *head) :current(head) {
  Register("ListIterator","Iterator");
}

ListIterator::~ListIterator() {}
Object *ListIterator::Next() {
  if(current == nullptr) return nullptr;
  Object *o=current->value;
  current = current->next;
  return o;
}


}
```

**Node.cpp**

```cpp
/***************************************************************************
  Node.cpp
  Roman Lumetsberger

  Implementation for class Node
****************************************************************************/
#include "Node.h"

using namespace std;

namespace ML {

Node::Node(Object * value, Node *prev, Node* next)
 : value(value), prev(prev), next(next) {
   Register("Node","Object");
 }

Node::~Node(){
  /* nothing todo */
}

std::string Node::AsString() const {
    if( value == nullptr)
      return "<nullptr>";

    return value->AsString();
}

}
```

**Set.cpp**

```
/*********************************************************************************
  Set.cpp
  Roman Lumetsberger

  Implementation for class Set
*********************************************************************************/
#include "Set.h"
#include "List.h"
#include <cassert>
#include <iostream>

using namespace std;

namespace ML {

Set::Set(): List() {
  Register("Set","List");
}

Set::~Set() {
  /* nothing todo */
}
void Set::Add(Object* o) {
  if(!List::Contains(o)) {
    List::Add(o);
  }
  else {
    cerr << o->AsString() << " already in this set!" << endl;
  }
}

Set *Set::Union(Set* other) const {
  assert(other != nullptr);
  Set *result = new Set();

  //add elements from first set
  Iterator *it = NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    if(!result->Contains(o)) {
      result->Add(o);
    }
  }
  delete it;

  //add elements from second set;
  it = other->NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    if(!result->Contains(o)) {
      result->Add(o);
    }
  }
  delete it;
  return result;
}

Set *Set::Intersect(Set* other) const {
  assert(other != nullptr);
  Set *result = new Set();

  Iterator *it = NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    if(other->Contains(o)) {
      result->Add(o);
    }
  }
  delete it;
  return result;
}

Set *Set::Difference(Set* other) const {
  assert(other != nullptr);
  Set *result = new Set();

  Iterator *it = NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    if(!other->Contains(o)) {
      result->Add(o);
    }
  }
  delete it;
  return result;

}

}
```

**BagNode.cpp**

```
/*********************************************************************************
  BagNode.cpp
  Roman Lumetsberger

  Implementation for class BagNode
*********************************************************************************/
#include "BagNode.h"
#include <MLObject.h>
#include "Node.h"
#include <sstream>

using namespace std;

namespace ML {
```

```cpp

BagNode::BagNode(Object *value,
      BagNode *prev,
      BagNode *next) : Node(value, prev, next), count(1) {
  Register("BagNode","Node");
}

BagNode::~BagNode() { /* nothing todo */ }

std::string BagNode::AsString() const {
  if( value == nullptr)
    return "<nullptr>";
  stringstream ss;
  ss << value->AsString() << "(" << count << ")";
  return ss.str();
}


}
```

**Bag.cpp**

```cpp
/*********************************************************************************
  Bag.cpp
  Roman Lumetsberger

  Implementation for class Bag, BagIterator
*********************************************************************************/
#include "Set.h"
#include "Bag.h"
#include <cassert>
#include <iostream>

using namespace std;

namespace ML {

Bag::Bag() :Set() {
  Register("Bag", "Set");
}

Bag::~Bag(){
  Clear();
}

BagNode *Bag::Find(Object* o) const {
  Node *parentNode = Set::Find(o);
  if(parentNode == nullptr ) return nullptr;

  BagNode *bagNode = dynamic_cast<BagNode *>(parentNode);
  assert(bagNode != nullptr);
  return bagNode;
}

BagNode *Bag::CreateNode(Object *o) const {
  return new BagNode(o);
}

```

```cpp
void Bag::Add(Object* o) {
  BagNode *node = Find(o);
  if(node != nullptr) {
    node->count++;
    size++;
  }
  else {
    Set::Add(o);
  }
}

Object *Bag::Remove(Object* o) {
  BagNode *node = Find(o);
  if(node != nullptr) {
    if(node->count > 1) {
      node->count--;
      size--;
    }
    else {
      Set::Remove(o);
    }

    return node->value;
  }

  return nullptr;
}

void Bag::DeleteElements() {
  Node * current = head;
  while(current != nullptr) {
    Node *tmp =current;
    current = current->next;
    delete tmp->value;
    delete tmp;
  }
  head = nullptr;
  size = 0;
}

Iterator *Bag::NewIterator() const {
  BagNode *bagHead = nullptr;
  if(head != nullptr) {
    bagHead = dynamic_cast<BagNode*>(head);
    assert(bagHead !=nullptr);
  }
  return new BagIterator(bagHead);
}

Bag *Bag::Union(Bag* other) const {
  assert(other != nullptr);
  Bag *result = new Bag();

  //add elements from first Bag
  Iterator *it = NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    result->Add(o);
  }
  delete it;

  //add elements from second Bag;
  it = other->NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    result->Add(o);
  }
  delete it;
  return result;
}

Bag *Bag::Intersect(Bag* other) const {
  assert(other != nullptr);
  Bag *result = new Bag();

  Iterator *it = NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    if(other->Contains(o)) { // other bag contains o
      //Object not already in result bag
      if(!result->Contains(o)) {
        BagNode *thisBagNode = Find(o);
        BagNode *otherBagNode = other->Find(o);

        int count = min(thisBagNode->count, otherBagNode->count);
        for(int i = 0; i < count; i++){
          result->Add(o);
        }
      }
    }
  }
  delete it;
  return result;
}

Bag *Bag::Difference(Bag* other) const {
  assert(other != nullptr);
  Bag *result = new Bag();

  Iterator *it = NewIterator();
  for(Object *o = it->Next(); o != nullptr;o = it->Next()) {
    //Object not already in result bag
    if(!result->Contains(o)) {
      BagNode *thisBagNode = Find(o);
      BagNode *otherBagNode = other->Find(o);

      int count = thisBagNode->count;
      if(otherBagNode != nullptr)
        count -= otherBagNode->count;

      for(int i = 0; i < count; i++){
        result->Add(o);
      }
    }
  }
  delete it;
  return result;

}
```

```cpp




BagIterator::BagIterator(BagNode *head) :current(head), currentNodeCount(0) {
  Register("BagIterator","Iterator");
  if(head != nullptr) {
    currentNodeCount = head->count;
  }
}

BagIterator::~BagIterator() {/*nothing todo */}

Object *BagIterator::Next() {
  if(current == nullptr) return nullptr;
  Object *o=current->value;
  currentNodeCount--;
  if(currentNodeCount == 0) {
    if(current->next != nullptr) {
      current = dynamic_cast<BagNode*>(current->next);
      assert(current != nullptr);
      currentNodeCount = current->count;
    }
    else {
      current = nullptr;
    }
  }
  return o;
}

}
```

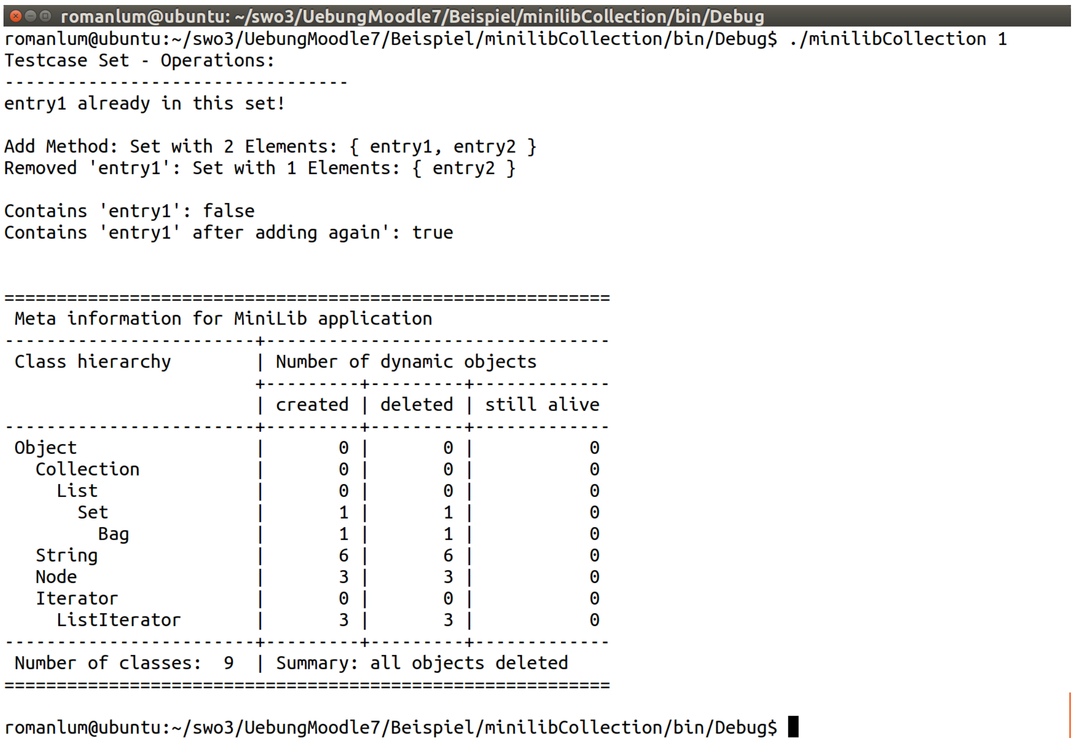
**main.cpp**

```cpp
#include <iostream>
#include "Set.h"
#include "Bag.h"
#include <MLString.h>
#include <cassert>

using namespace std;
using namespace ML;

int main(int argc ,char** argv)
{
  if(argc != 2) {
    cerr << "Wrong parameter count" << endl;
    cerr << "Usage: " << argv[0] << " testcase";
    return 0;
  }

  Set *testSet = new Set();
  Bag *testBag = new Bag();
  String *testEntry = new String("entry1");
  String *testEntry2 = new String("entry2");
  String *testEntry3 = new String("entry3");

  int testcase = atoi(argv[1]);
  switch(testcase) {
    case 1:
      {
        cout << "Testcase Set - Operations:" << endl;
        cout << "----------------------------------" << endl;
        testSet->Add(new String("entry1"));
        testSet->Add(new String("entry2"));
        testSet->Add(testEntry);

        cout << endl;
        cout << "Add Method: " << *testSet << endl;

        Object *removedEntry =  testSet->Remove(testEntry);
        assert(removedEntry->IsEqualTo(testEntry));
        cout << "Removed 'entry1': " << *testSet << endl << endl;
        delete removedEntry;

        cout << "Contains 'entry1': " << boolalpha << testSet->Contains(testEntry) << endl;
        testSet->Add(new String("entry1"));
        cout << "Contains 'entry1' after adding again': " << boolalpha << testSet->Contains(testEntr

      }
      break;

    case 2:
      {
        cout << "Testcase Set - Clear:" << endl;
        cout << "----------------------------------" << endl;

        testSet->Add(testEntry);
        testSet->Add(testEntry2);
        testSet->Add(testEntry3);
        testSet->Clear();
        cout << "Nodes deleted but Strings not" << endl;
        WriteMetaInfo();
        cout << endl;
        break;
      }
    case 3:
      {
        cout << "Testcase Set - DeleteElements:" << endl;
        cout << "----------------------------------" << endl;

        testSet->Add(testEntry);
        testSet->Add(testEntry2);
        testSet->Add(testEntry3);
        testSet->DeleteElements();
        cout << "Nodes and Strings deleted" << endl;
        WriteMetaInfo();
        testEntry = nullptr;
        testEntry2 = nullptr;
        testEntry3 = nullptr;
        cout << endl;
        break;
      }
    case 4:
      {
        cout << "Testcase Set - Union:" << endl;
        cout << "----------------------------------" << endl;

        testSet->Add(testEntry);
        testSet->Add(testEntry2);

        Set *secondSet = new Set();
        secondSet->Add(testEntry3);

        Set *result = testSet->Union(secondSet);
        cout << "Set1: " << *testSet << endl;
        cout << "Set2: " << *secondSet << endl;
        cout << "Union Result: " << *result << endl;
        delete secondSet;
        delete result;
        testSet->Clear();

        break;
      }
    case 5:
      {
        cout << "Testcase Set - Difference:" << endl;
        cout << "----------------------------------" << endl;

        testSet->Add(testEntry);
        testSet->Add(testEntry2);
        testSet->Add(testEntry3);

        Set *secondSet = new Set();
        secondSet->Add(testEntry3);

        Set *result = testSet->Difference(secondSet);

        cout << "Set1: " << *testSet << endl;
        cout << "Set2: " << *secondSet << endl;
        cout << "Difference Result: " << *result << endl;

        delete secondSet;
        delete result;
        testSet->DeleteElements();
        testEntry = nullptr;
        testEntry2 = nullptr;
        testEntry3 = nullptr;

        break;
      }
      case 6:
      {
        cout << "Testcase Set - Intersect:" << endl;
        cout << "----------------------------------" << endl;

        testSet->Add(testEntry);
        testSet->Add(testEntry2);
        testSet->Add(testEntry3);

        Set *secondSet = new Set();
        secondSet->Add(testEntry3);

        Set *result = testSet->Intersect(secondSet);

        cout << "Set1: " << *testSet << endl;
        cout << "Set2: " << *secondSet << endl;
        cout << "Intersect Result: " << *result << endl;

        delete secondSet;
        delete result;
        testSet->DeleteElements();
        testEntry = nullptr;
        testEntry2 = nullptr;
        testEntry3 = nullptr;
        break;
      }

      case 7:
      {
        cout << "Testcase Bag - Operations:" << endl;
        cout << "----------------------------------" << endl;
        testBag->Add(new String("entry1"));
        testBag->Add(new String("entry2"));
        testBag->Add(testEntry);

        cout << endl;
        cout << "Add Method: " << *testBag << endl;

        Object *removedEntry =  testBag->Remove(testEntry);
        assert(removedEntry->IsEqualTo(testEntry));
        cout << "Removed 'entry1': " << *testBag << endl << endl;
        removedEntry =  testBag->Remove(testEntry);
        cout << "Removed 'entry1': " << *testBag << endl << endl;

        delete removedEntry;
        cout << "Contains 'entry1': " << boolalpha << testBag->Contains(testEntry) << endl;
        testBag->Add(new String("entry1"));
        cout << "Contains 'entry1' after adding again': " << boolalpha << testBag->Contains(testEntr
        break;
      }
      case 8:
      {
        cout << "Testcase Bag - Clear:" << endl;
        cout << "----------------------------------" << endl;

        testBag->Add(testEntry);
        testBag->Add(testEntry2);
        testBag->Add(testEntry3);
        testBag->Clear();
        cout << "Nodes deleted but Strings not" << endl;
        WriteMetaInfo();
        cout << endl;
        break;
      }
    case 9:
      {
        cout << "Testcase Bag - DeleteElements:" << endl;
        cout << "----------------------------------" << endl;

        testBag->Add(testEntry);
        testBag->Add(testEntry2);
      testBag->Add(testEntry3);
      testBag->Add(testEntry3);
      testBag->DeleteElements();
      cout << "Nodes and Strings deleted" << endl;
      WriteMetaInfo();
      testEntry = nullptr;
      testEntry2 = nullptr;
      testEntry3 = nullptr;
      cout << endl;
      break;
    }
    case 10:
    {
      cout << "Testcase Bag - Union:" << endl;
      cout << "----------------------------------" << endl;

      testBag->Add(testEntry);
      testBag->Add(testEntry2);

      Bag *secondBag = new Bag();
      secondBag->Add(testEntry);
      secondBag->Add(testEntry3);
      secondBag->Add(testEntry3);

      Bag *result = testBag->Union(secondBag);
      cout << "Bag1: " << *testBag << endl;
      cout << "Bag2: " << *secondBag << endl;
      cout << "Union Result: " << *result << endl;
      delete secondBag;
      delete result;
      testBag->Clear();

      break;
    }
    case 11:
    {
      cout << "Testcase Bag - Difference:" << endl;
      cout << "----------------------------------" << endl;

      testBag->Add(testEntry);
      testBag->Add(testEntry2);
      testBag->Add(testEntry2);
      testBag->Add(testEntry3);
      testBag->Add(testEntry3);
      testBag->Add(testEntry3);

      Bag *secondBag = new Bag();
      secondBag->Add(testEntry3);
      secondBag->Add(testEntry2);
      secondBag->Add(testEntry);
      secondBag->Add(testEntry);

      Bag *result = testBag->Difference(secondBag);

      cout << "Bag1: " << *testBag << endl;
      cout << "Bag2: " << *secondBag << endl;
      cout << "Difference Result: " << *result << endl;

```

```
        delete secondBag;
        delete result;
        testBag->DeleteElements();
        testEntry = nullptr;
        testEntry2 = nullptr;
        testEntry3 = nullptr;

        break;
      }
      case 12:
      {
        cout << "Testcase Bag - Intersect:" << endl;
        cout << "----------------------------------" << endl;

        testBag->Add(testEntry);
        testBag->Add(testEntry);
        testBag->Add(testEntry2);
        testBag->Add(testEntry3);
        testBag->Add(testEntry3);
        testBag->Add(testEntry3);
        testBag->Add(testEntry3);

        Bag *second = new Bag();

        second->Add(testEntry2);
        second->Add(testEntry2);
        second->Add(testEntry3);
        second->Add(testEntry3);


        Bag *result = testBag->Intersect(second);

        cout << "Bag1: " << *testBag << endl;
        cout << "Bag2: " << *second << endl;
        cout << "Intersect Result: " << *result << endl;

        delete second;
        delete result;
        testBag->DeleteElements();
        testEntry = nullptr;
        testEntry2 = nullptr;
        testEntry3 = nullptr;
        break;
      }

  }
  testSet->DeleteElements();
  testBag->DeleteElements();
  delete testSet;
  delete testBag;
  delete testEntry;
  delete testEntry2;
  delete testEntry3;
  WriteMetaInfo();

}
```

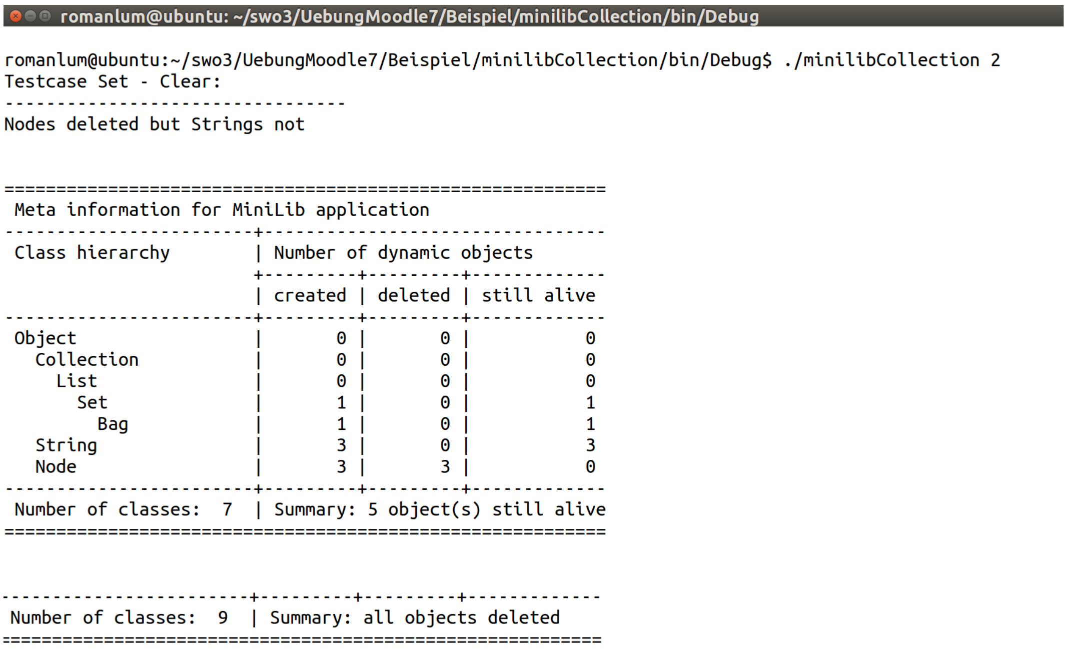
## 2.5 Testfälle

### 2.5.1 Testfall 1 - Set Operationen


romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug

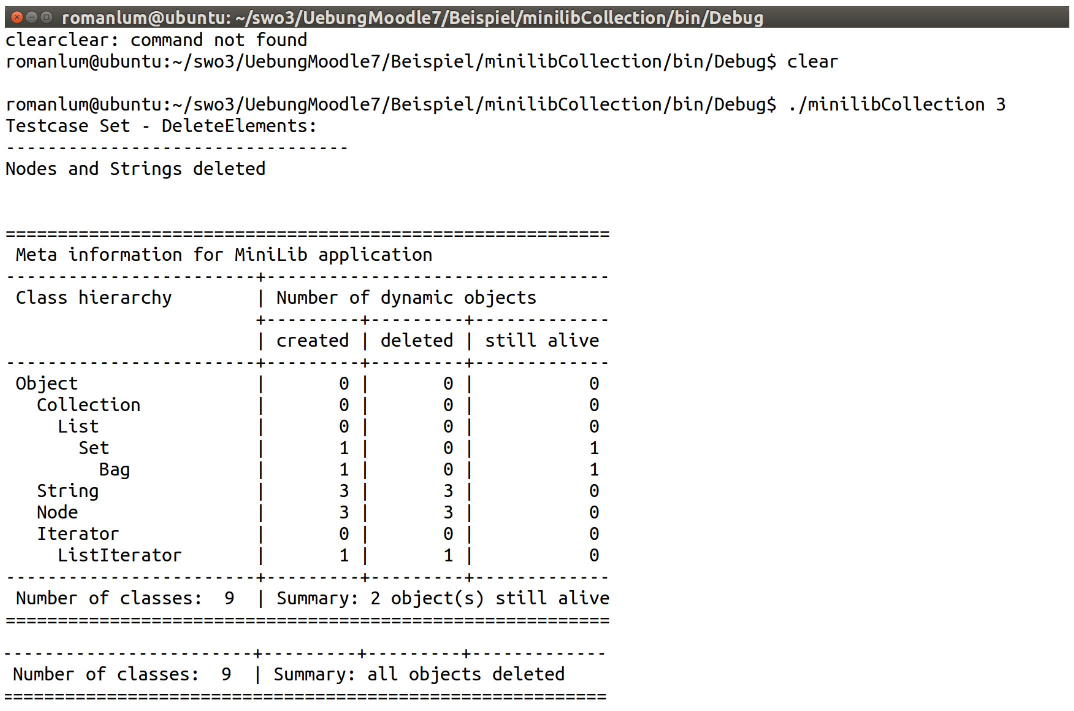
```
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 1
Testcase Set - Operations:
--------------------------------
entry1 already in this set!

Add Method: Set with 2 Elements: { entry1, entry2 }
Removed 'entry1': Set with 1 Elements: { entry2 }

Contains 'entry1': false
Contains 'entry1' after adding again': true


=========================================================
 Meta information for MiniLib application
------------------------+--------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       1 |       1 |           0
         Bag            |       1 |       1 |           0
   String               |       6 |       6 |           0
   Node                 |       3 |       3 |           0
   Iterator             |       0 |       0 |           0
     ListIterator       |       3 |       3 |           0
------------------------+---------+---------+------------
 Number of classes:  9  | Summary: all objects deleted
=========================================================

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ █
```

### 2.5.2 Testfall 2 - Set Clear


romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug

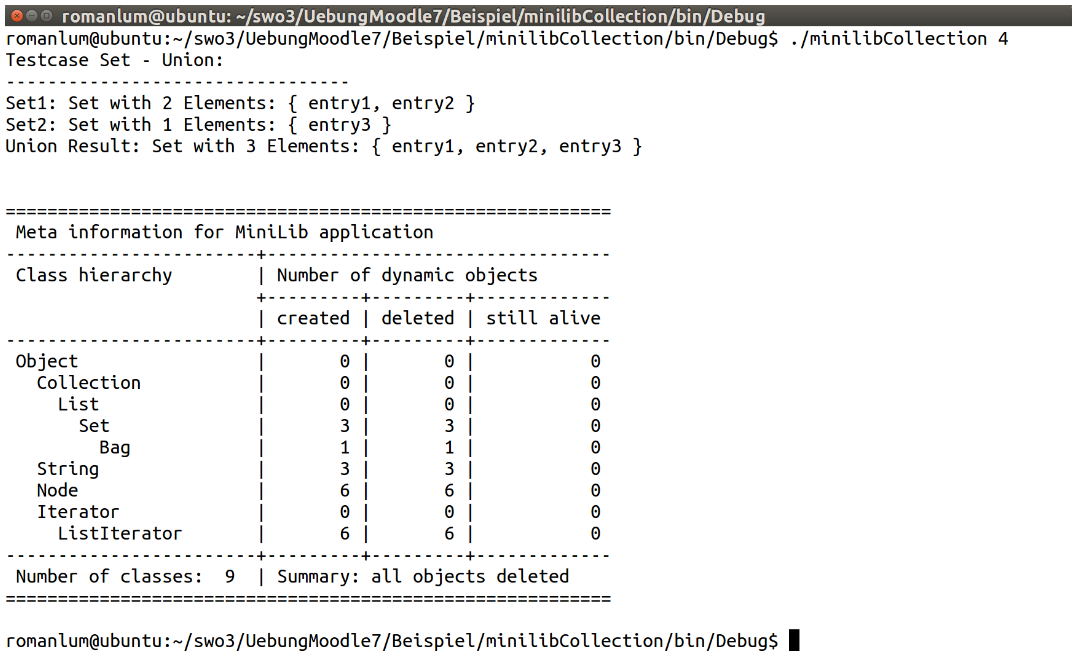
```
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 2
Testcase Set - Clear:
--------------------------------
Nodes deleted but Strings not


=========================================================
 Meta information for MiniLib application
------------------------+--------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       1 |       0 |           1
         Bag            |       1 |       0 |           1
   String               |       3 |       0 |           3
   Node                 |       3 |       3 |           0
------------------------+---------+---------+------------
 Number of classes:  7  | Summary: 5 object(s) still alive
=========================================================


------------------------+---------+---------+------------
 Number of classes:  9  | Summary: all objects deleted
=========================================================
```

### 2.5.3 Testfall 3 - Set DeleteElements



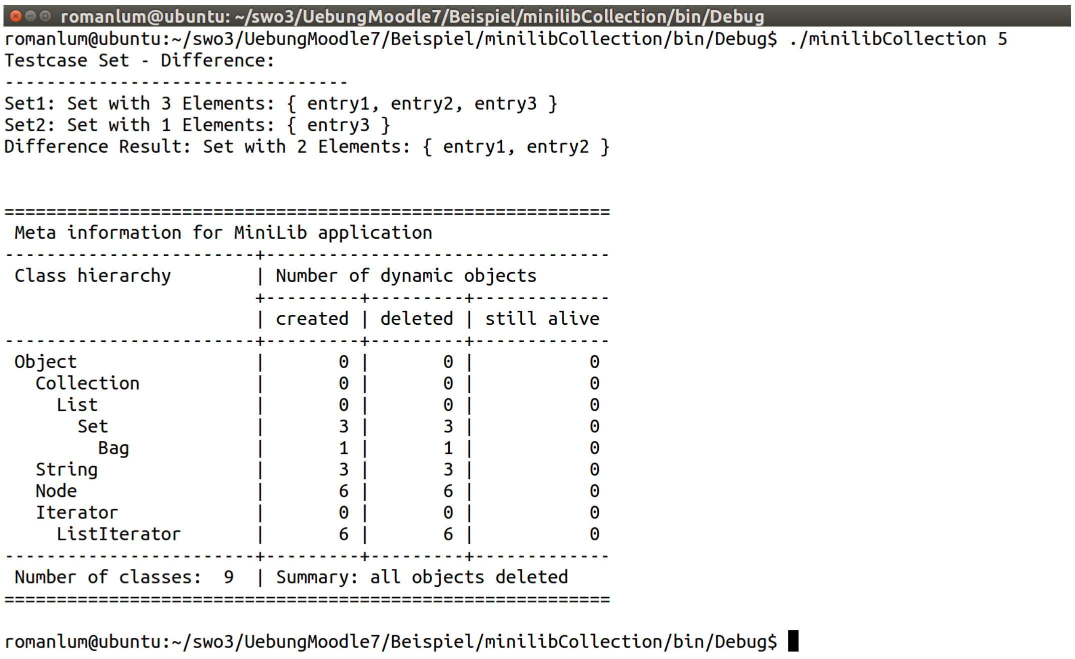
```
romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug
clearclear: command not found
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ clear

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 3
Testcase Set - DeleteElements:
--------------------------------
Nodes and Strings deleted


=========================================================
 Meta information for MiniLib application
------------------------+--------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       1 |       0 |           1
         Bag            |       1 |       0 |           1
   String               |       3 |       3 |           0
   Node                 |       3 |       3 |           0
   Iterator             |       0 |       0 |           0
     ListIterator       |       1 |       1 |           0
------------------------+---------+---------+------------
 Number of classes:  9  | Summary: 2 object(s) still alive
=========================================================

------------------------+---------+---------+------------
 Number of classes:  9  | Summary: all objects deleted
=========================================================
```

### 2.5.4 Testfall 4 - Set Union



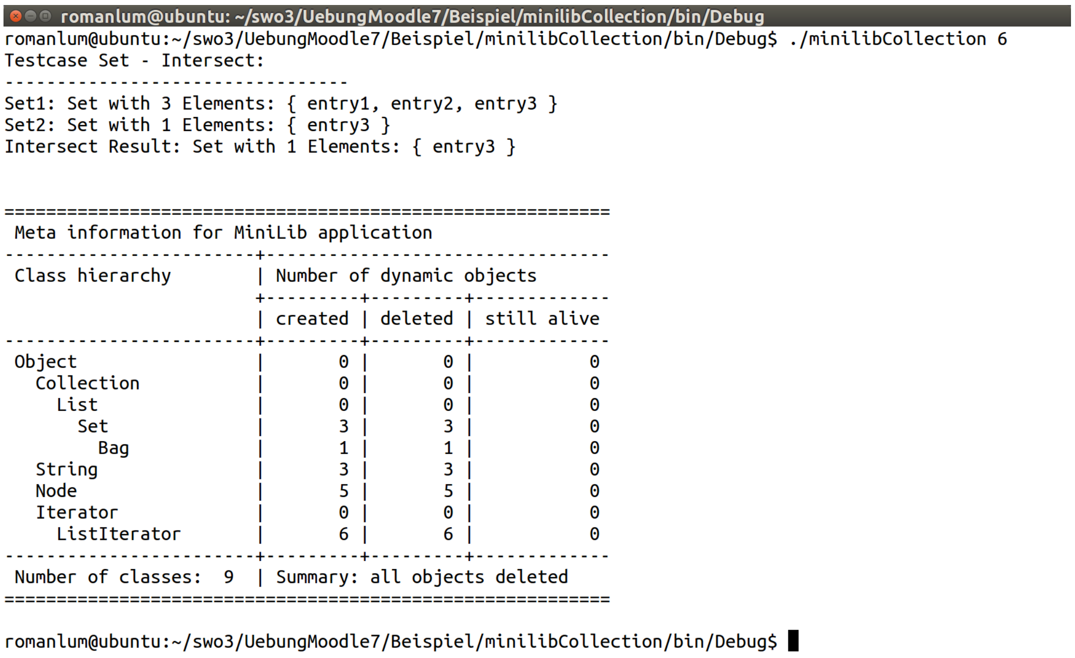
```
romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 4
Testcase Set - Union:
--------------------------------
Set1: Set with 2 Elements: { entry1, entry2 }
Set2: Set with 1 Elements: { entry3 }
Union Result: Set with 3 Elements: { entry1, entry2, entry3 }


=========================================================
 Meta information for MiniLib application
------------------------+--------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       3 |       3 |           0
         Bag            |       1 |       1 |           0
   String               |       3 |       3 |           0
   Node                 |       6 |       6 |           0
   Iterator             |       0 |       0 |           0
     ListIterator       |       6 |       6 |           0
------------------------+---------+---------+------------
 Number of classes:  9  | Summary: all objects deleted
=========================================================

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ █
```

### 2.5.5 Testfall 5 - Set Difference



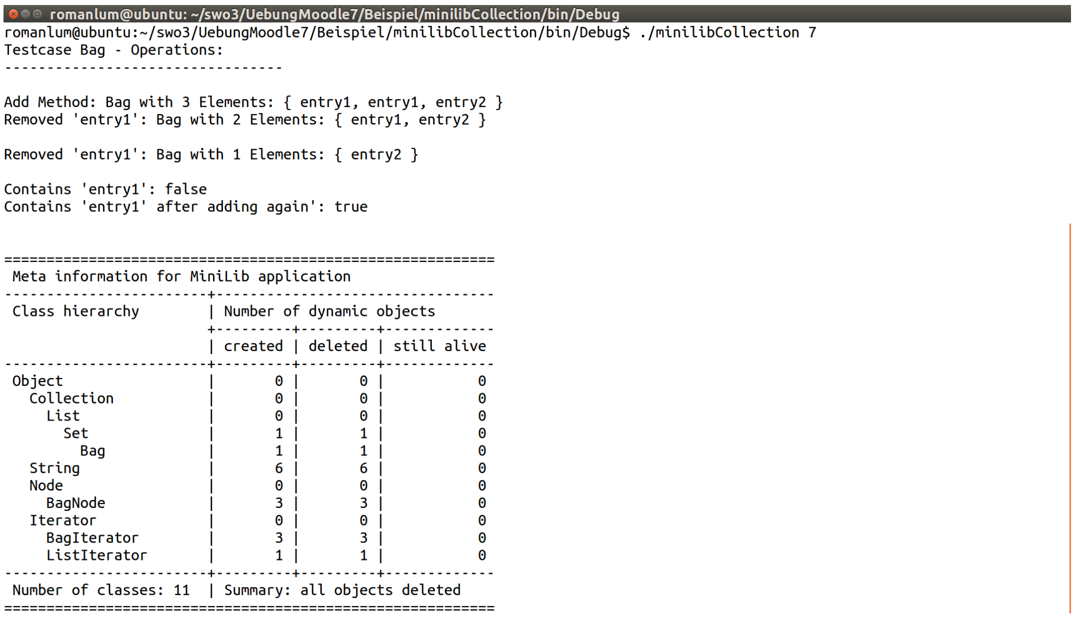
```
romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 5
Testcase Set - Difference:
--------------------------------
Set1: Set with 3 Elements: { entry1, entry2, entry3 }
Set2: Set with 1 Elements: { entry3 }
Difference Result: Set with 2 Elements: { entry1, entry2 }


==========================================================
 Meta information for MiniLib application
------------------------+---------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       3 |       3 |           0
         Bag            |       1 |       1 |           0
   String               |       3 |       3 |           0
   Node                 |       6 |       6 |           0
   Iterator             |       0 |       0 |           0
     ListIterator       |       6 |       6 |           0
------------------------+---------+---------+------------
 Number of classes:  9  | Summary: all objects deleted
==========================================================

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ █
```

### 2.5.6 Testfall 6 - Set Intersect



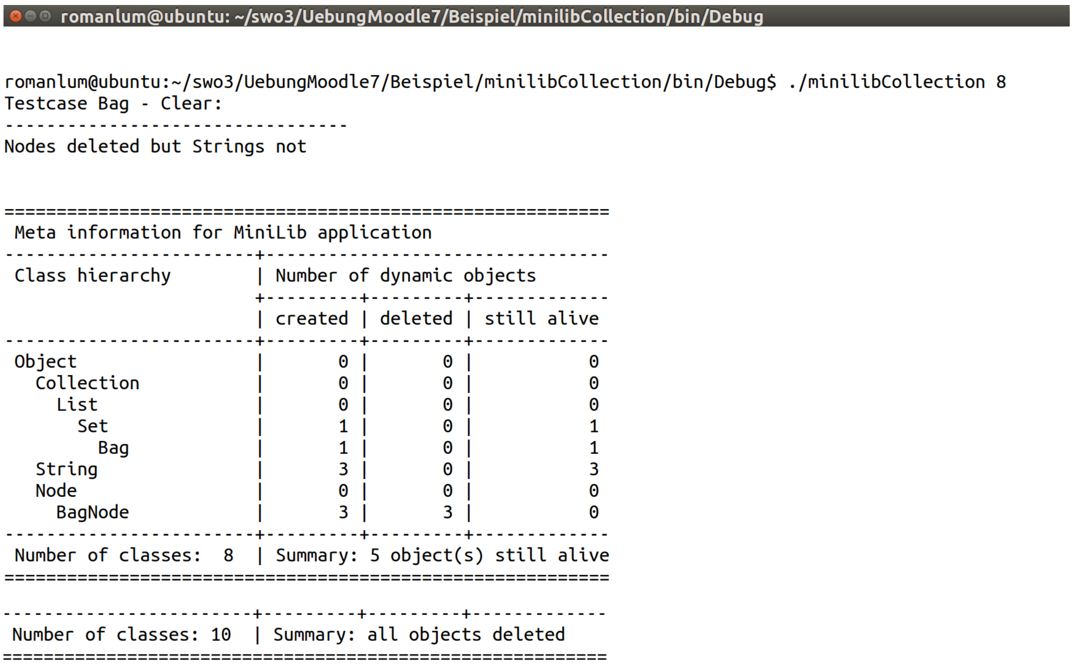
```
romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 6
Testcase Set - Intersect:
--------------------------------
Set1: Set with 3 Elements: { entry1, entry2, entry3 }
Set2: Set with 1 Elements: { entry3 }
Intersect Result: Set with 1 Elements: { entry3 }


==========================================================
 Meta information for MiniLib application
------------------------+---------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       3 |       3 |           0
         Bag            |       1 |       1 |           0
   String               |       3 |       3 |           0
   Node                 |       5 |       5 |           0
   Iterator             |       0 |       0 |           0
     ListIterator       |       6 |       6 |           0
------------------------+---------+---------+------------
 Number of classes:  9  | Summary: all objects deleted
==========================================================

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ █
```

### 2.5.7 Testfall 7 - Bag Operationen


romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug

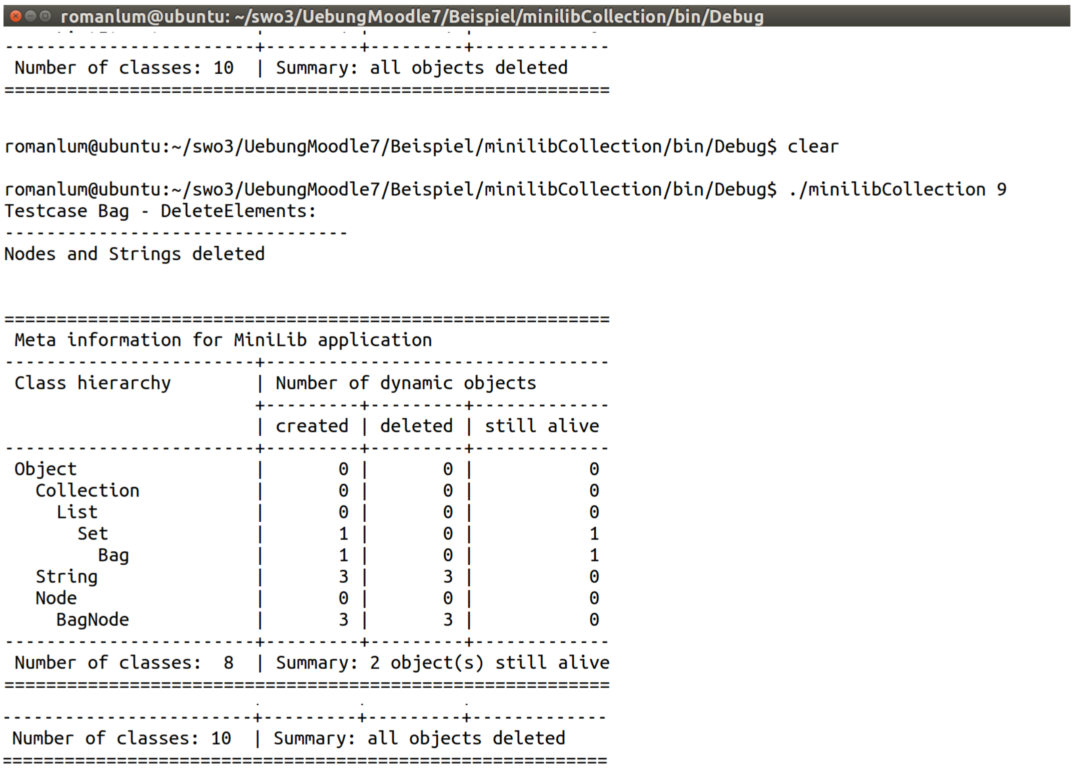
```
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 7
Testcase Bag - Operations:
--------------------------------

Add Method: Bag with 3 Elements: { entry1, entry1, entry2 }
Removed 'entry1': Bag with 2 Elements: { entry1, entry2 }

Removed 'entry1': Bag with 1 Elements: { entry2 }

Contains 'entry1': false
Contains 'entry1' after adding again': true


=========================================================
 Meta information for MiniLib application
-----------------------+---------------------------------
 Class hierarchy       | Number of dynamic objects
                       +---------+---------+------------
                       | created | deleted | still alive
-----------------------+---------+---------+------------
 Object                |       0 |       0 |           0
   Collection          |       0 |       0 |           0
     List              |       0 |       0 |           0
       Set             |       1 |       1 |           0
         Bag           |       1 |       1 |           0
   String              |       6 |       6 |           0
   Node                |       0 |       0 |           0
     BagNode           |       3 |       3 |           0
   Iterator            |       0 |       0 |           0
     BagIterator       |       3 |       3 |           0
     ListIterator      |       1 |       1 |           0
-----------------------+---------+---------+------------
 Number of classes: 11 | Summary: all objects deleted
=========================================================
```

### 2.5.8 Testfall 8 - Bag Clear


romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug

```
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 8
Testcase Bag - Clear:
--------------------------------
Nodes deleted but Strings not


=========================================================
 Meta information for MiniLib application
-----------------------+---------------------------------
 Class hierarchy       | Number of dynamic objects
                       +---------+---------+------------
                       | created | deleted | still alive
-----------------------+---------+---------+------------
 Object                |       0 |       0 |           0
   Collection          |       0 |       0 |           0
     List              |       0 |       0 |           0
       Set             |       1 |       0 |           1
         Bag           |       1 |       0 |           1
   String              |       3 |       0 |           3
   Node                |       0 |       0 |           0
     BagNode           |       3 |       3 |           0
-----------------------+---------+---------+------------
 Number of classes:  8 | Summary: 5 object(s) still alive
=========================================================

-----------------------+---------+---------+------------
 Number of classes: 10 | Summary: all objects deleted
=========================================================
```

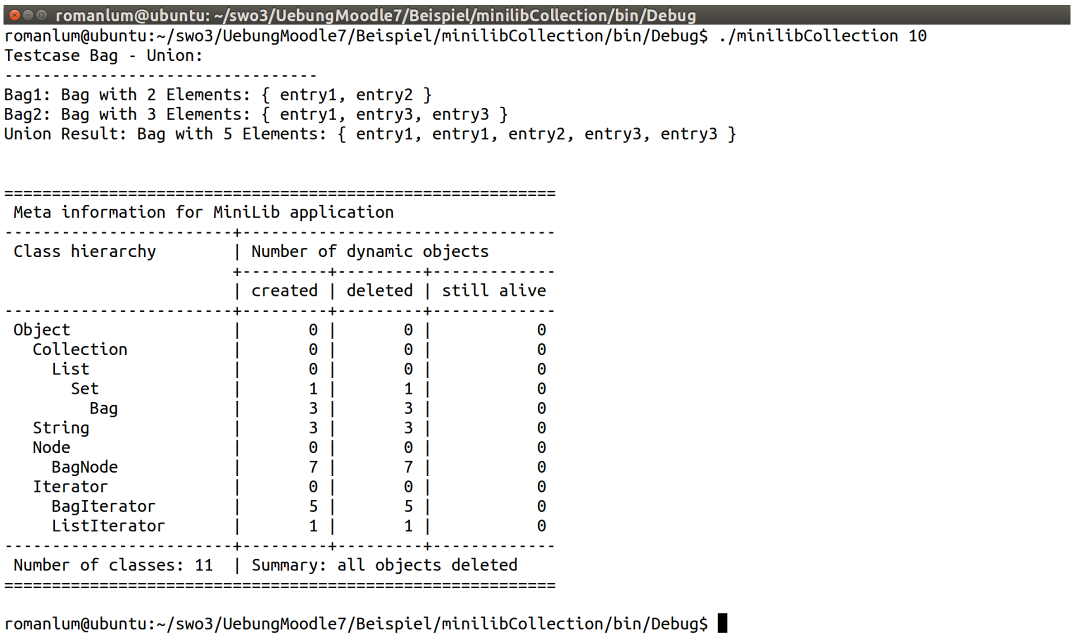
### 2.5.9 Testfall 9 - Bag DeleteElements



romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug

```
-----------------------+---------+---------+------------
 Number of classes: 10  | Summary: all objects deleted
=========================================================


romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ clear

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 9
Testcase Bag - DeleteElements:
--------------------------------
Nodes and Strings deleted


=========================================================
 Meta information for MiniLib application
-----------------------+---------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
-----------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       1 |       0 |           1
         Bag            |       1 |       0 |           1
   String               |       3 |       3 |           0
   Node                 |       0 |       0 |           0
     BagNode            |       3 |       3 |           0
-----------------------+---------+---------+------------
 Number of classes:  8  | Summary: 2 object(s) still alive
=========================================================

-----------------------+---------+---------+------------
 Number of classes: 10  | Summary: all objects deleted
=========================================================
```

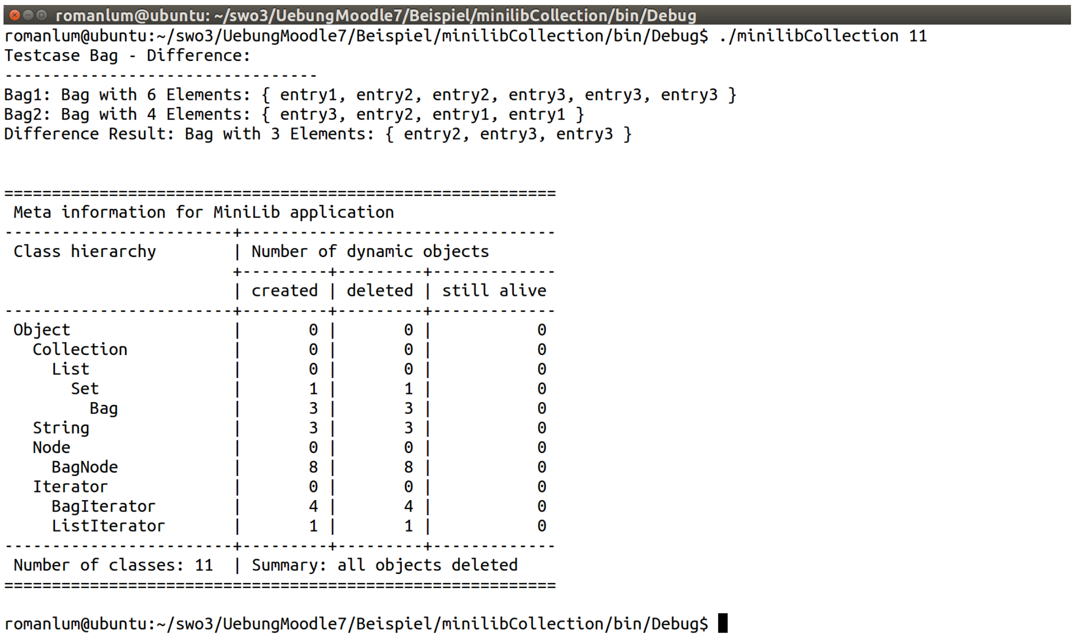
### 2.5.10 Testfall 10 - Bag Union



romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug

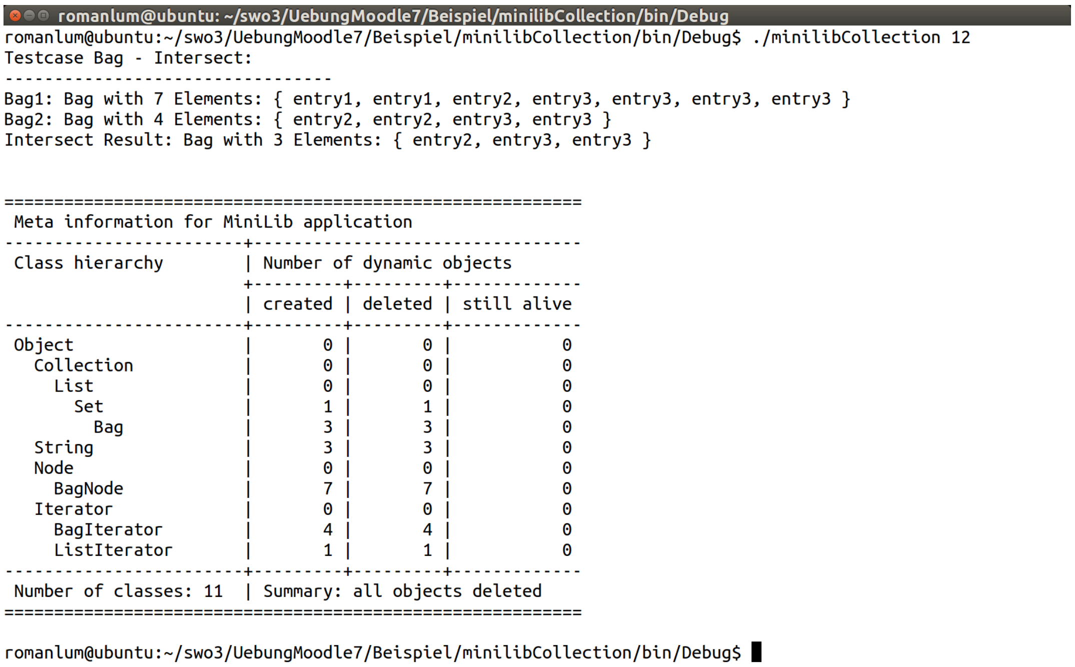
```
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 10
Testcase Bag - Union:
--------------------------------
Bag1: Bag with 2 Elements: { entry1, entry2 }
Bag2: Bag with 3 Elements: { entry1, entry3, entry3 }
Union Result: Bag with 5 Elements: { entry1, entry1, entry2, entry3, entry3 }


=========================================================
 Meta information for MiniLib application
-----------------------+---------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
-----------------------+---------+---------+------------
 Object                 |       0 |       0 |           0
   Collection           |       0 |       0 |           0
     List               |       0 |       0 |           0
       Set              |       1 |       1 |           0
         Bag            |       3 |       3 |           0
   String               |       3 |       3 |           0
   Node                 |       0 |       0 |           0
     BagNode            |       7 |       7 |           0
   Iterator             |       0 |       0 |           0
     BagIterator        |       5 |       5 |           0
     ListIterator       |       1 |       1 |           0
-----------------------+---------+---------+------------
 Number of classes: 11  | Summary: all objects deleted
=========================================================

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ █
```

### 2.5.11 Testfall 11 - Bag Difference



```
romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 11
Testcase Bag - Difference:
--------------------------------
Bag1: Bag with 6 Elements: { entry1, entry2, entry2, entry3, entry3, entry3 }
Bag2: Bag with 4 Elements: { entry3, entry2, entry1, entry1 }
Difference Result: Bag with 3 Elements: { entry2, entry3, entry3 }


=========================================================
 Meta information for MiniLib application
------------------------+--------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |          0
   Collection           |       0 |       0 |          0
     List               |       0 |       0 |          0
       Set              |       1 |       1 |          0
         Bag            |       3 |       3 |          0
   String               |       3 |       3 |          0
   Node                 |       0 |       0 |          0
     BagNode            |       8 |       8 |          0
   Iterator             |       0 |       0 |          0
     BagIterator        |       4 |       4 |          0
     ListIterator       |       1 |       1 |          0
------------------------+---------+---------+------------
 Number of classes: 11  | Summary: all objects deleted
=========================================================

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ █
```

### 2.5.12 Testfall 12 - Bag Intersect



```
romanlum@ubuntu: ~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug
romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ ./minilibCollection 12
Testcase Bag - Intersect:
--------------------------------
Bag1: Bag with 7 Elements: { entry1, entry1, entry2, entry3, entry3, entry3, entry3 }
Bag2: Bag with 4 Elements: { entry2, entry2, entry3, entry3 }
Intersect Result: Bag with 3 Elements: { entry2, entry3, entry3 }


=========================================================
 Meta information for MiniLib application
------------------------+--------------------------------
 Class hierarchy        | Number of dynamic objects
                        +---------+---------+------------
                        | created | deleted | still alive
------------------------+---------+---------+------------
 Object                 |       0 |       0 |          0
   Collection           |       0 |       0 |          0
     List               |       0 |       0 |          0
       Set              |       1 |       1 |          0
         Bag            |       3 |       3 |          0
   String               |       3 |       3 |          0
   Node                 |       0 |       0 |          0
     BagNode            |       7 |       7 |          0
   Iterator             |       0 |       0 |          0
     BagIterator        |       4 |       4 |          0
     ListIterator       |       1 |       1 |          0
------------------------+---------+---------+------------
 Number of classes: 11  | Summary: all objects deleted
=========================================================

romanlum@ubuntu:~/swo3/UebungMoodle7/Beispiel/minilibCollection/bin/Debug$ █
```