Hochschule Deggendorf Dr. Peter Jüttner	
Vorlesung: Objektorientierte Programmierung	SS 2019
Übung 8	Termin 16.5.19

Templateklassen - Musterlösung

1. Aufgabe: Templateklasse

a.) Ändern Sie die Klasse Liste aus der Übung 4 (s. Laufwerk) in eine Template Klasse. Ändern Sie dazu sowohl die Klasse Listenelement, als auch die Klasse Liste in eine Template Klasse um.

Die Definition beider Klassen schaut dann folgendermaßen aus:

```
template <class K> class Liste;
template <class K> class Listenelement
{ public:
K Element;
Listenelement* next;
};
template <class K> class Liste …
```

Versehen Sie alle Vorkommen der Klasse Listenelement in der Klasse Liste mit dem Template Parameter K, z.B. Listenelement<K> *l;

Entfernen Sie die Methode listenausgabe aus der Template Klasse Liste.

```
template <class K> class Liste;

template <class K> class Listenelement
{ public:
  K Element;
  Listenelement* next;
};

template <class K> class Liste
```

```
{ protected:
  Listenelement<K> *l; /* Pointer auf erstes Listenelement, NULL, falls
Liste leer */
  public:
  Liste()
  { l = NULL;
  };

  void Tail()
  { if (l!=NULL)
    { Listenelement<K> *hp = l->next;
      delete l;
      l=hp;
    };
  };

  void Append(int i)
  { Listenelement<K> *hp = new Listenelement<K>;
    hp->Element = i;
    hp->next = l;
    l=hp;
  };

  int isempty()
  { return (l==NULL);
  };

  void emptylist()
  { l=NULL;
  };

  K Head()
  { if (isempty())
    { printf("Fehler: Head mit leerer Liste aufgerufen \n");
      return (0);
    }
    else return l->Element;
  };

  ~Liste()
  { Listenelement<K> *hp1, *hp2;
    hp1 = l;
    while (hp1!=NULL)
    { hp2 = hp1->next;
      delete hp1;
      hp1 = hp2;
```

```
    };
  };
};
```

b.) Schreiben Sie eine Unterklasse I_Liste der Klasse Liste, wobei Sie diese mit dem Typ int ausprägen.

class I_Liste : public Liste<int>

Ergänzen Sie die Klasse I_Liste um die zuvor gelöschte Methode listenausgabe.

```
class I_Liste : public Liste<int>
{ public:
  void Listenausgabe()
  { if (isempty())
     printf("Liste ist leer\n");
    else
    { Listenelement<int> *hilfspointer = l;
      int i = 1;
      while (hilfspointer != NULL)
      { printf("%d-tes Element:%d\n",i,hilfspointer->Element);
        i++;
        hilfspointer = hilfspointer->next;
      };
    };
  };
};
```

c.) Probieren Sie die Klasse I_Liste aus.

```
int main()
{ int i;
  I_Liste list;
  list.emptylist();
  char c = 'x';


  while ((c != 'e') && (c != 'E'))
  { printf("Bitte Listenoperation eingeben: a=Append, h=Head, t=Tail,
p=Ausgabe, e = Ende\n");
    fflush(stdin);
    c = getchar();
    switch (c)
```

```
    { case 'e':
      case 'E': break;
      case 'a':
      case 'A': printf("Append gewaehlt\n");
                printf("Bitte int-Zahl eingeben\n");
                scanf("%d",&i);
                list = (const I_Liste&)list.Append(i);
                break;
      case 'h':
      case 'H': printf("Head gewaehlt\n");
                i = list.Head();
                printf("Headelement: %d\n",i);
                break;
      case 't':
      case 'T': printf("Tail gewaehlt\n");
                list = (const I_Liste&)list.Tail();
                break;
      case 'p':
      case 'P': printf("Ausgabe gewaehlt\n");
           list.Listenausgabe();
                break;
      default: break;
    }
  };

};
```