Linked lists revisited

now in C#

- We know them from Pascal,
- in C# we may implement them similarly (except of deallocation):

-
  ```
  class element{
  public int value;
  public element next;
  }
  ```
- Initialization: `element a=null;`
- Deleting the list: `a=null;`

Universal list

and its implementation

```
abstract class listgen
{    public listgen next;
     public listgen Next
     {    set{next=value;}
          get{return next;}
     }
     public listgen() {    next=null;}
}
```

Example – son

of generic linked list

```
class intlist:listgen {    public int val;
     public intlist(int val)
     { this.val=val; }
} listgen l=new intlist(10);
...
```

Keyword `using`

for the last time

- We know how to create namespaces, we know how to use them (`using`), but there is yet one remarkable feature of this keyword:
- We may use it to create alias for a particular class.
- Syntax: `using <alias>=<class>;`
- Example: `using c=System.Console;`
  `c.WriteLine();`
- Now we know everything about the namespaces...
- ... except of their particular content (which you may learn yourself).

Files I/V

StreamReader and StreamWriter

- As in Pascal, it is very simple to work with text-files.
- Compared to Pascal, in C# it behaves completely differently.
- Standard input (output) is governed by class `Console` in namespace `System`.
- Text-files are represented by `StreamReader` and `StreamWriter`.
- These classes are in `System.IO`. We establish a variable of this type and associate it with a file. Then it behaves similarly to static class `Console`:
- `System.IO.StreamReader r = new System.IO.StreamReader(@"c:\temp\file.txt");`
- Now variable `r` is associated with `file.txt`. To work with the file, we will be calling its methods...

Files II/V

StreamReader and StreamWriter, their methods and attributes

- `void r.Close() //closes the Reader/Writer`
- `string r.ReadToEnd() //reads the file until the end`
- `w.Write(what) //writes into the file`
- `r.EndOfStream //attribute equivalent to EOF`
- `@"..."//fixed string, backslash (e.g., \r) won't be interpretted`
-

Files III/V – example

make a copy of a file

```
System.IO.StreamReader r=new
System.IO.StreamReader("file.txt");
System.IO.StreamWriter w=new
System.IO.StreamWriter("cheapcopy.txt");
w.Write(r.ReadToEnd());
w.Close(); r.Close();
```

Copying file by characters

i.e., the same example in a different way

```
System.IO.StreamReader r=new
System.IO.StreamReader("file.txt");
System.IO.StreamWriter w=new
System.IO.StreamWriter("cheapcopy.txt");
while(!r.EndOfStream) w.Write((char)r.Read());
w.Close();
```

Files V/V – remarks

- Methods `ReadLine()` and `WriteLine()`,
- character-encoding – we may pass as a second argument to the Reader/Writer constructor:
- Example:

```
Encoding e=Encoding.GetEncoding(1250);
Encoding f=Encoding.GetEncoding(852);
System.IO.StreamReader r=new
System.IO.StreamReader("file.txt",e);
System.IO.StreamWriter w=new
System.IO.StreamWriter("cheapcopy.txt",f);
w.Write(r.ReadToEnd());
w.Close();
```

Computer simulation I

is a very important topic

- Considering a problem complicated enough to get imagined, we should get an opinion of it.
- We may simulate many topics (e.g., injury – namely its healing, how the alcohol gets spreaded through the organism, how the elevators are serving people...)
- Computer simulation is a simulation where for modelling a computer-program is used.
- The aim is to decide how the simulated objects interact, i.e., how the whole simulated system works.
- Simulation should not optimize the processes!
- Results may differ, basic information is time of end (of a simulation).

Computer simulation II

- Continuous VS Discrete event simulation,
- problem of continuous simulation usually needs some (differential) equation to get solved,
- we restrict our attention to discrete event simulation,
- continuous simulation can be approximated making small steps and recalculating very often (which needs so called stability of the system).

Cars transporting a sand

very typical problem on our lectures

- We want to move a sand-heap to a building yard [site].
- We have a given number of workers and a given number of cars,
- along the path, critical sections may turn up (at the building year and at the heap, too).
- Number of workers is bounded, road may be narrow (for one car only driven by traffic-light or there are no overtaking zones).
- How to schedule workers and cars to move the heap as soon as possible?
- Discrete simulation simulates a given schedule (workers and cars) and determins when the heap will be moved completely.
- Usually we will consider one narrow zone (for one car only).