

```

P_RatNumForName (SKYFLATNAME);
// If no world levels, no secret exit
if ( ! (gamemode == commercial) )
    && (W_CheckNumForName("map311")<0)
    secretexit = false;
else
    secretexit = true;
gamemode = ga_completed;

// skip the description field
memset (vcheck, 0, sizeof(vcheck));
sprintf (vcheck, "version %d", VERSION);
if (strcmp (base.p, vcheck))
    return; // bad version
save_p += VERSIONSIZE;

// This was quite messy with SPECIAL and commented parts,
// supposedly backs to make the latest edition work.
// It might not work properly.
if (episode < 1)
    episode = 1;

if (gamemode == retail )
{
    if (episode < 4)
        episode = 4;
}
else if (gamemode == shareware )
{
    if (episode < 3)
        episode = 1; // only start episode 1 on shareware
}
else
{
    if (episode < 3)
        episode = 3;
}

if (map < 1)
    map = 1;
if (map > 9)
    && (gamemode != commercial )
    map = 9;

M_ClearRandom ();

if (skill == sk_nightmare || respawnparm )
    respawnmonsters = true;
else
    respawnmonsters = false;

// fastparm || skill == sk_nightmare && gameskill != sk_nightmare
for (i=S_SARG_RUN1; i<=S_SARG_RUN2; i++)
    statesh.tics >>= 1;
mobinfo[MT_BRUISERSHOT].speed = 20*FRACUNIT;
mobinfo[MT_TROOPSHOT].speed = 20*FRACUNIT;

// force players to be initialized upon first level load
for (i=0; i<MAXPLAYERS; i++)
    players[i].playerstate = PST_REBORN;

usergame = true; // will be set false if a demo
paused = false;
demoPlayback = false;
autoMapview = false;
viewactive = true;
gamemode = episode;
gamemap = map;
gameskill = skill;
viewactive = true;

```

C#

# Generics, Collections, Iterators, and Regular Expressions

Rasmus Lystrøm  
 External Associate Professor  
 ITU

```

// force players to be initialized upon first level load
for (i=0; i<MAXPLAYERS; i++)
    players[i].playerstate = PST_REBORN;

usergame = true; // will be set false if a demo
paused = false;
demoPlayback = false;
autoMapview = false;
viewactive = true;
gamemode = episode;
gamemap = map;
gameskill = skill;
viewactive = true;

```

# Agenda

Regular Expressions

Generic Collections

Iterators

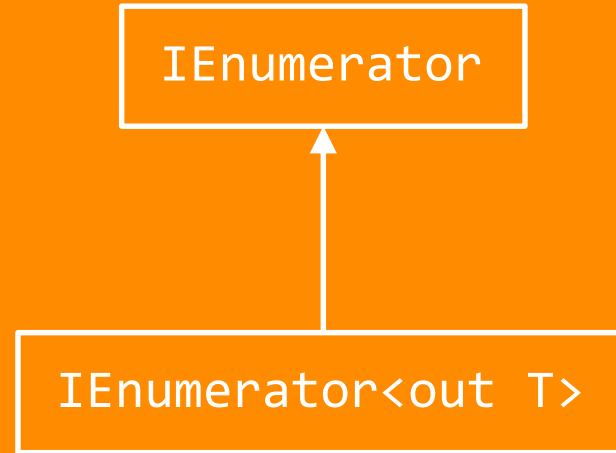
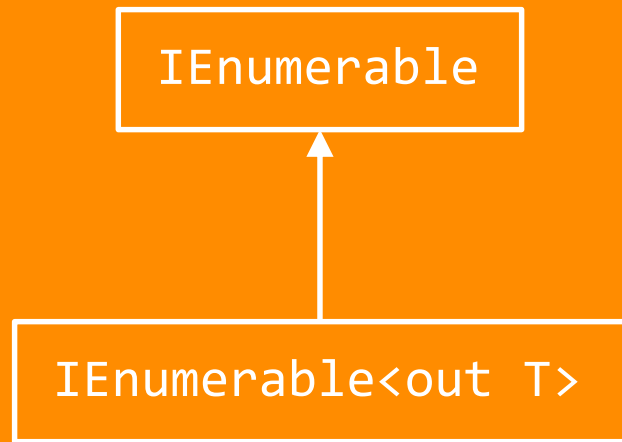
# Regular Expressions

*	Zero or more times the previous character
+	Once or more times the previous character
?	Zero or one time the previous character
.	Any single character (not \n)
\s	Any whitespace character (e.g. tab)
\S	Any non-whitespace character
\b	Word boundary
\B	Any non-word boundary position
\w	Any word character (a-z, A-Z, 0-9)
\W	Any non-word character
^	Start of the input text
\$	End of the input text

# Regular Expressions

[1c]	matches character '1' or 'c'
[a-z]	matches all lower-case letters
[a-zA-Z]	matches all letters
[0-9]+	matches integer numbers
[0-9]+\.[0-9]+	matches a floating point
[0-2][0-9]:[0-5][0-9]	matches a time e.g. 12:34

# Iterators



Producer

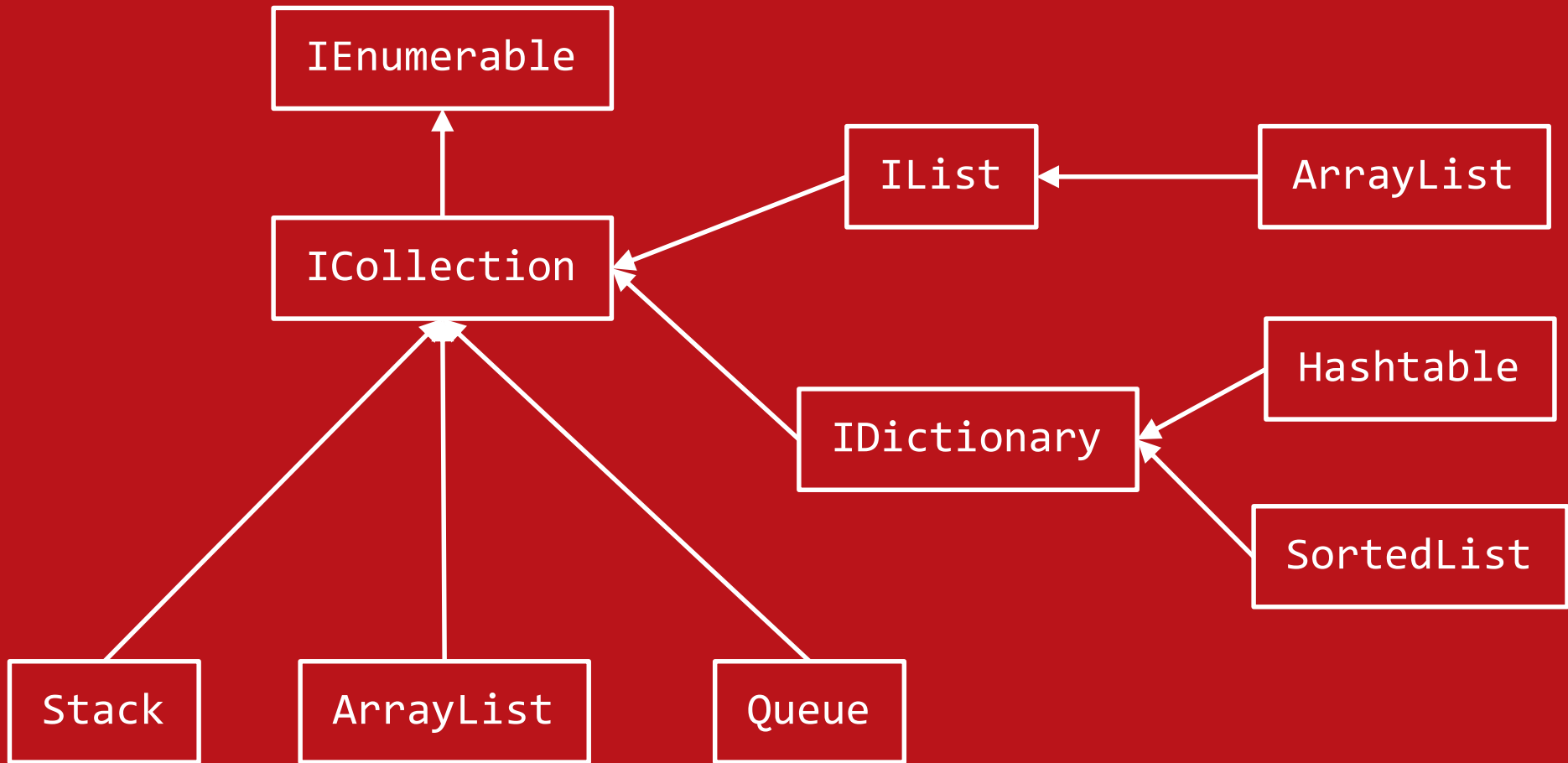
`yield return T;`

`yield break;`

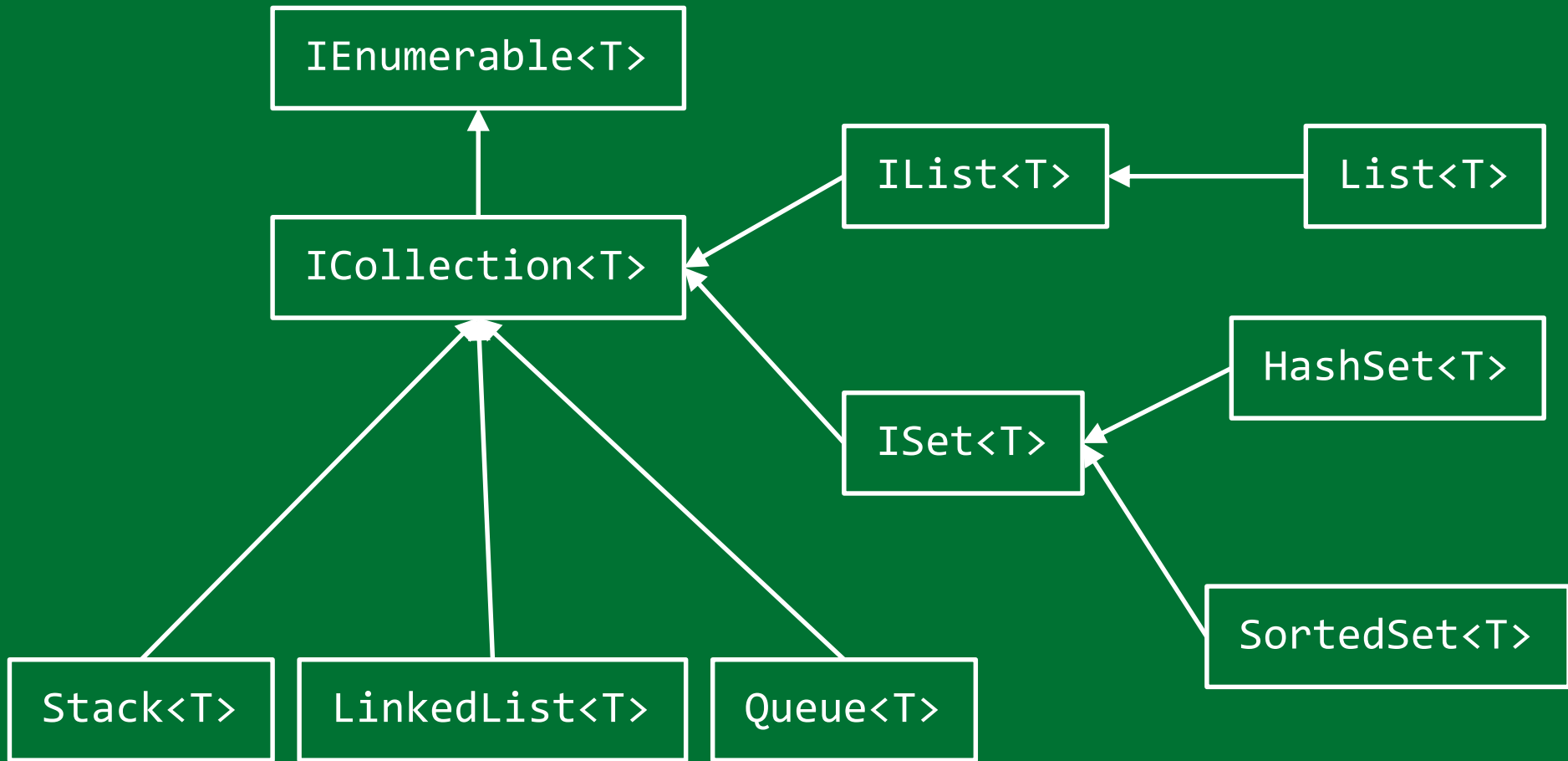
Building block for Linq

```
foreach (var item in items)
{
}
```

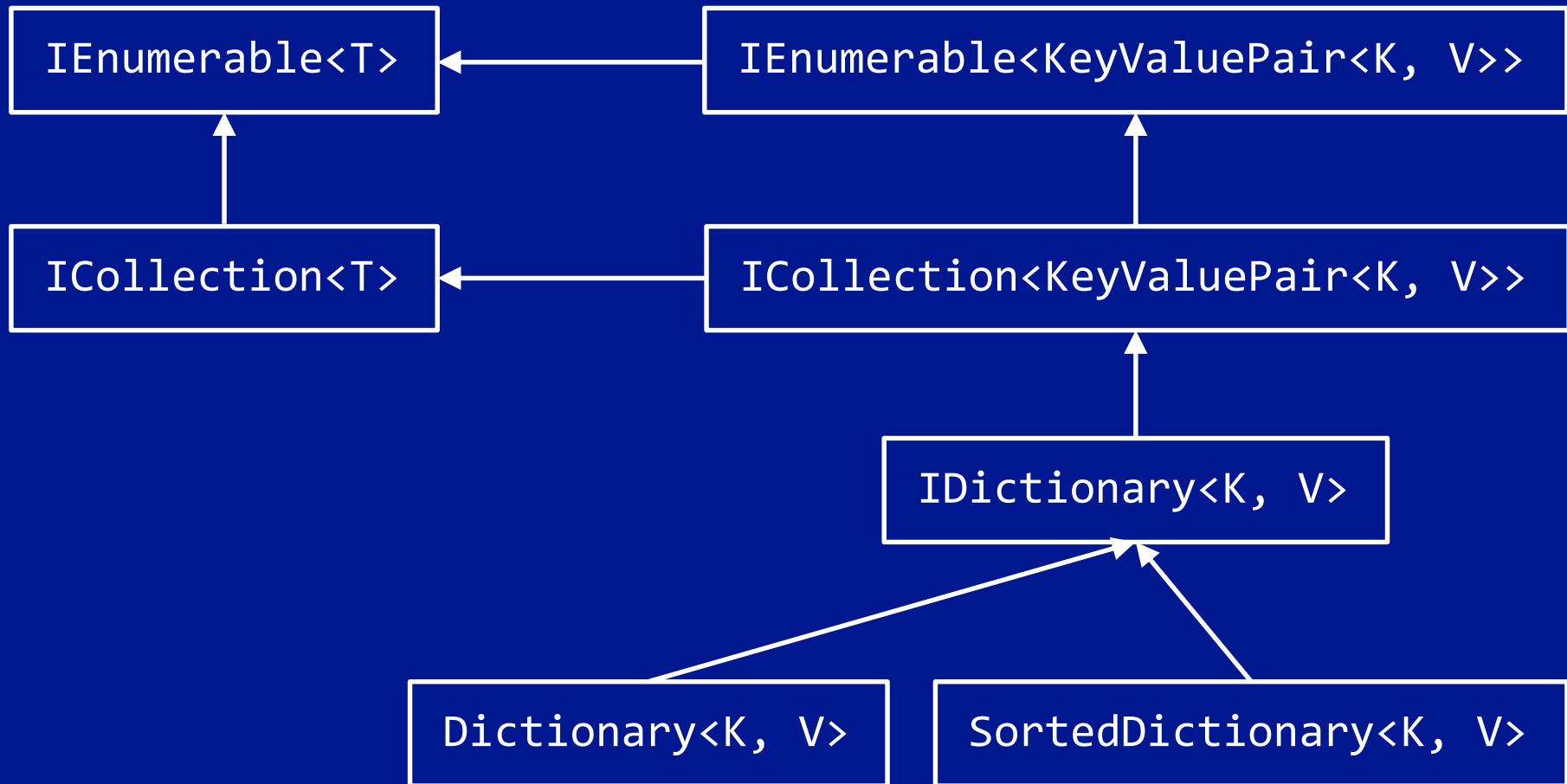
# System.Collections



# System.Collections.Generic



# System.Collections.Generic 2





# System.Collections.Concurrent

