12. Óra

Miről lesz szó

Generics

Általános, nem specifikus egy adott adattípusra.

- Műveletek, (pl: az elemek hozzáadása és eltávolítása a gyűjteményből) alapvetően ugyanúgy történnek, *függetlenül* a tárolt adatok *típusától*.

Lehetnek

Osztályok

```
public class List<T> {
    ...
}
```

Mezők

```
public class ZsakbaMacska<T> {
    T Present { get; set; };
    ...
}
```

Interface-k

```
public interface IList<T> : ICollection<T>, IEnumerable<T>, IEnumerable {
    ...
}
```

Metódusok

```
static void Swap<T>(ref T thingA, ref T thingB) {
    T temp;
    temp = thingA;
    thingA = thingB;
    thingB = temp;
}
```

stb...

```
public class List<T> : IList<T>, IList, IReadOnlyList<T> {
    ... //                   ↑ Generikus interface-k  ↑
}
```

List implementációja

Generikus típus jelölés konvenció: `T` (`T` prefix ha több generikus típus van)

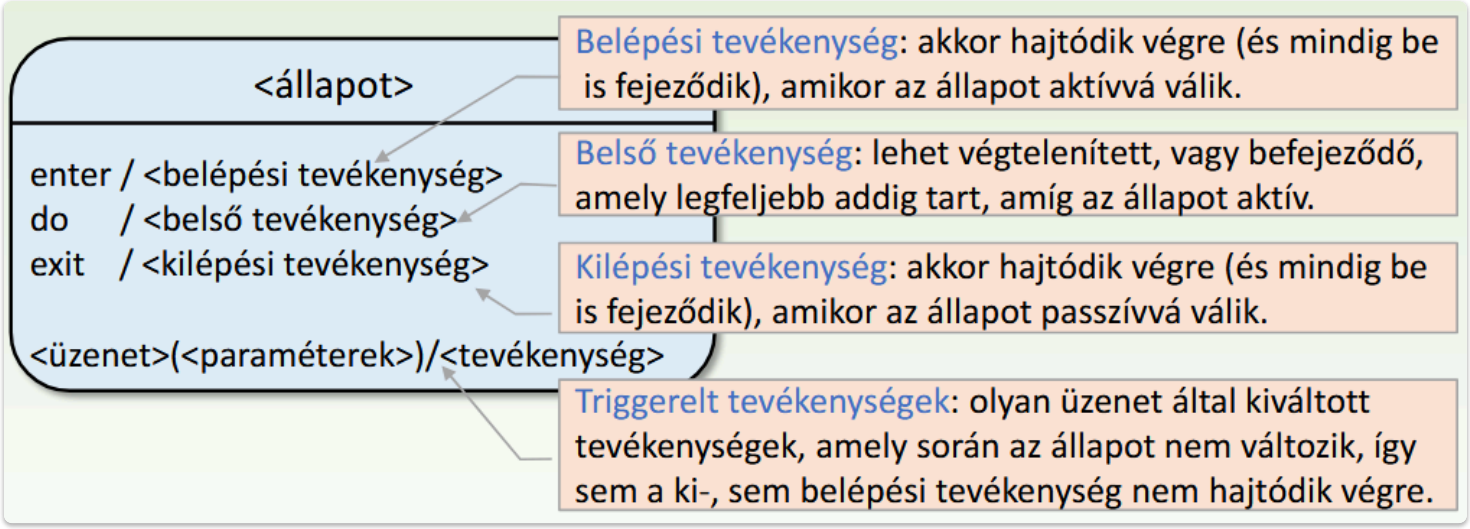
Megkötések a típus paraméterre

```
public class MyClass1<T> where T : notnull {
    // T nem lehet nullable
}

// kicsit komplikáltabbak is:

public interface IEnemy {
    public void Attack();
}

// IEnemy interface-t meg kell valósítania
public class Dungeon<T> where T : IEnemy {
    public Dungeon(T enemy) {
        enemy.Attack();
    }
}
```

Az összes megkötés: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/constraints-on-type-parameters

Állapotgépek

Előadás diái: https://people.inf.elte.hu/gt/oep/10.%C3%81llapotg%C3%A9p.pdf



Állapotgép megvalósítása

állapot / esemény	állapot1 / start: inic()	állapot2	állapot3
esemény1	állapot2	állapot3	
esemény2		állapot1 / akció()	állapot1

```
inic()
állapot := állapot1
while állapot≠stop loop
  switch esemény
    case esemény1:
      switch állapot
        case állapot1 : állapot := állapot2
        case állapot2 : állapot := állapot3
        case állapot3 :
      endswitch
    case esemény2:
      switch állapot
        case állapot1 :
        case állapot2 : akció()
                        állapot := állapot1
        case állapot3 : állapot := állapot1
      endswitch
  endswitch
endloop
```

az elágazások kiküszöbölésére állapot-, illetve látogató tervezési mintát alkalmazzunk

```
inic()
állapot := állapot1
while állapot≠stop loop
  switch állapot
    case állapot1:
      switch esemény
        case esemény1 : állapot := állapot2
      endswitch
    case állapot2:
      switch esemény
        case esemény1 : állapot := állapot3
        case esemény2 : akció()
                        állapot := állapot1
      endswitch
    case állapot3:
      switch esemény
        case esemény2 : állapot := állapot1
      endswitch
  endswitch
endloop
```

Gregorics Tibor: Objektumelvű programozás 11

Feladatok

```
+-------------------------------------------------------------------------------------------------------+
|                                  BasicStateMachine<TState, TSignal>                                   |
|-------------------------------------------------------------------------------------------------------|
| - _transitionMap: Dictionary<(TState, TSignal), TransitionInfo>                                       |
| - _currentState: TState                                                                               |
|-------------------------------------------------------------------------------------------------------|
| + CurrentState: TState { return _currentState }                                                       |
|-------------------------------------------------------------------------------------------------------|
| # BasicStateMachine(startState: TState)                                                               |
| # AddTransition(fromState: TState, signal: TSignal, toState: TState, onTransition: Action? = null): void|
| # ProcessSignal(signal: TSignal): void                                                                |
+-------------------------------------------------------------------------------------------------------+

Public Nested:
+-----------------------------------------------------------+
| InvalidTransitionException                                |
|-----------------------------------------------------------|
| + InvalidTransitionException(state: TState, signal: TSignal) |
```

```
+-----------------------------------------------------------+

Private Nested:
+-------------------------------------------+
| TransitionInfo                            |
|-------------------------------------------|
| + ToState: TState                         |
| + OnTransition: Action?                   |
+-------------------------------------------+
```

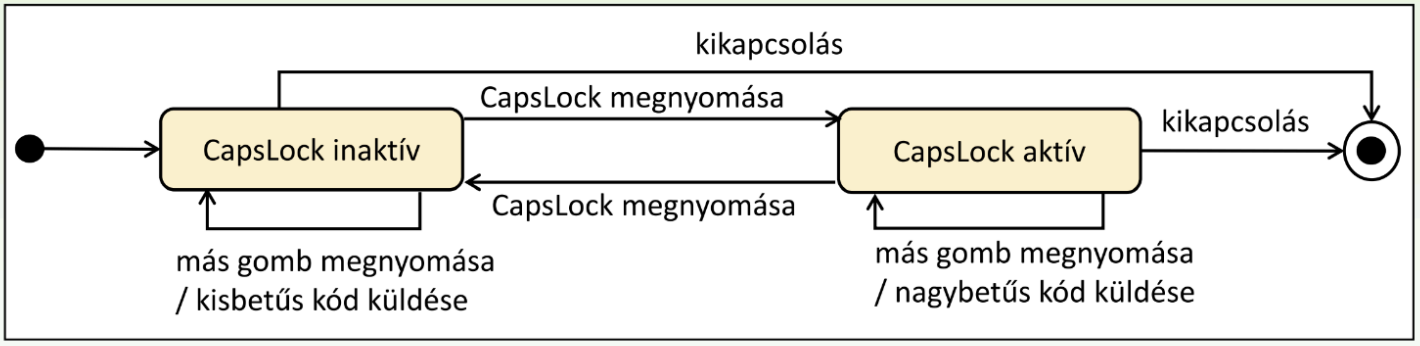
Billentyűzet

Billentyűzet

Egyszerűsített billentyűzet modellezése:
ha a CapsLock aktivált, akkor minden más billentyű lenyomásra nagybetűs karaktereket, különben kisbetűs karaktereket kapunk.

Legyen két állapot: CapsLock aktív, illetve inaktív.

Legyen háromféle művelet: CapsLock lenyomása, más gomb lenyomása, kikapcsolás.



	inaktív	**aktív**
CapsLock	aktív	inaktív
más nyomógomb	inaktív **/** kisbetűs kód küldése	aktív **/** nagybetűs kód küldése
kikapcsolás	exit	exit

Közlekedési lámpa

Közlekedési lámpa

Egy közlekedési lámpán piros, piros-sárga, zöld, sárga fények vannak. A lámpa 60 másodpercig piros és 90 másodpercig zöld színű. Az átmeneti állapotok 5 másodpercig tartanak: pirosról a zöldre a piros-sárgán keresztül, zöldről a pirosra a sárgán keresztül. Kezdetben a lámpa piros.