Alexander Guðmundsson
alg35@hi.is

```
// Höfundur spurningar:  Snorri Agnarsson, snorri@hi.is
// Permalink spurningar: https://rise4fun.com/Dafny/C223

// Höfundur lausnar:     Alexander Guðmundsson
// Permalink lausnar:    https://rise4fun.com/Dafny/4BSV

// Klárið að forrita klasann IntStackArray.

trait IntStack
{
    ghost var ghostseq: seq<int>;
    ghost var Repr: set<object>;

    predicate Valid()
        reads this, Repr;

    predicate method IsEmpty()
        reads this, Repr;
        requires Valid();
        ensures IsEmpty() <==> ghostseq==[];

    method Push( x: int )
        modifies this, Repr;
        requires Valid();
        ensures Valid() && fresh(Repr-old(Repr));
        ensures ghostseq == old(ghostseq)+[x];

    method Pop() returns ( x: int )
        modifies this, Repr;
        requires Valid();
        requires ghostseq != [];
        ensures Valid() && fresh(Repr-old(Repr));
        ensures ghostseq == old(ghostseq[..|ghostseq|-1]);
        ensures x == old(ghostseq[|ghostseq|-1]);
}

class IntStackArray extends IntStack
{
    var a: array<int>;
    var size: int;

    predicate Valid()
        reads this, Repr;
```

Alexander Guðmundsson
alg35@hi.is

```
{
    // Hér vantar skilgreiningu á fastayrðingu gagna.
    // Notið IntQueueArray til hliðsjónar.
    // Eðlilegt er að innihald hlaðans sé í sætum
    // a[0],a[1],...,a[size-1], frá botni til topps.
    Repr == {this, a} &&
    a.Length > 0 &&
    0 <= size <= a.Length &&
    |ghostseq| == size &&
    if size > 0 then
        ghostseq == a[..size]
    else
        ghostseq == []

}

constructor()
    ensures Valid() && fresh(Repr-{this});
    ensures ghostseq == [];
{
    a := new int[1];
    size := 0;
    Repr := {a};
    ghostseq := [];
}

predicate method IsEmpty()
    reads this, Repr;
    requires Valid();
    ensures IsEmpty() <==> ghostseq==[];
{
    size == 0
}

method Push( x: int )
    modifies this, Repr;
    requires Valid();
    ensures Valid() && fresh(Repr-old(Repr));
    ensures ghostseq == old(ghostseq)+[x];
{
    size := size + 1;
    ghostseq := ghostseq + [x];
}

method Pop() returns ( x: int )
```

Alexander Guðmundsson
alg35@hi.is

```
        modifies this, Repr;
        requires Valid();
        requires ghostseq != [];
        ensures Valid() && fresh(Repr-old(Repr));
        ensures size == old(size)-1;
        ensures ghostseq == old(ghostseq)[..size];
        ensures x == old(ghostseq[|ghostseq|-1]);
    {
        size := size-1;
        x := a[size];
        ghostseq := ghostseq[..size];

    }
}

method Factory() returns ( s: IntStack )
    ensures fresh(s);
    ensures fresh(s.Repr);
    ensures s.Valid();
    ensures s.IsEmpty();
{
    s := new IntStackArray();
}

method Main()
{
    var s := [1,2,3];
    var s1 := Factory();
    var s2 := Factory();
    while s != []
        decreases |s|;
        invariant s1.Valid();
        invariant s2.Valid();
        invariant ({s1}+s1.Repr) !! ({s2}+s2.Repr);
        invariant fresh(s1.Repr);
        invariant fresh(s2.Repr);
    {
        s1.Push(s[0]);
        s2.Push(s[0]);
        s := s[1..];
    }
    while !s1.IsEmpty()
        decreases |s1.ghostseq|
        invariant s1.Valid();
        invariant s2.Valid();
```

```
        invariant ({s1}+s1.Repr) !! ({s2}+s2.Repr);
        invariant fresh(s1.Repr);
        invariant fresh(s2.Repr);
    {
        var x := s1.Pop();
        print x;
        print " ";
    }
    while !s2.IsEmpty()
        invariant s2.Valid();
        decreases |s2.ghostseq|
        invariant fresh(s2.Repr);
    {
        var x := s2.Pop();
        print x;
        print " ";
    }
}
```