TÖL212M Rökstudd Forritun - Hópverkefni 7

Andri Fannar Kristjánsson

2. mars 2025

Hópverkefni 7

1

Sækið skrána `H7-skeleton.java` og vistið hana hjá ykkur en breytið nafni hennar í H7.java. Klárið að forrita klasann í skránni.

1.1 Svar:

Hér fyrir neðan má sjá kóðann þar sem klasarnir hafa verið forritaðir. Dafny samþykkir þessa útgáfu, þótt `Tio.run` geri það ekki. Einnig er hægt að skoða kóðann á þessari slóð: https://tinyurl.com/3prs3fcn.

```
// Author of question:  Snorri Agnarsson, snorri@hi.is

// Author of solution:       Andri Fannar Kristjánsson, afk6@hi.is
// Permalink of solution:    https://tinyurl.com/3prs3fcn

// Klárið að forrita klasann IntStackArray.

// Finish programming the class IntStackArray.

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

  ghost predicate Valid()
    reads this, Repr;

  predicate 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;

  ghost predicate Valid()
    reads this, Repr;
  {
    // 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.

    // Here we need a definition of the data invariant.
    // Look at IntQueueArray as a precedent.
    // It is natural to have the contents of the stack
    // in a[0],[1],...,a[size-1], from bottom to top.
    Repr == {this, a} &&
    0 <= size <= a.Length &&
    1 <= a.Length &&
    |ghostseq| == size &&
    ghostseq == a[..size]
  }

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

  predicate 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];
  {
    // Hér vantar forritstexta.
    // Notið IntQueueArray til hliðsjónar.

    // Here code is missing.
    // Use IntQueueArray as a precedent.
    if size == a.Length
    {
      var newa := new int[2*a.Length];
      var i := 0;
      while i < size
        decreases size-i
        invariant 0 <= i <= |ghostseq|
            == size <= a.Length < newa.Length
        invariant newa[..i] == ghostseq[..i]
        invariant ghostseq == old(ghostseq)
        invariant Valid()
      {
        newa[i] := a[i];
        i := i+1;
        assert newa[..i] == ghostseq[..i];
      }
      a := newa;
      Repr := {this,a};
    }
    a[size] := x;
    size := size+1;
    ghostseq := ghostseq+[x];
    assert Valid();
  }

  method Pop() returns ( x: int )
    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]);
  {
    // Hér vantar forritstexta.

    // Here code is missing.
    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 "␣";
  }
}
```