F.CS203 - Өгөгдлийн бүтэц ба алгоритм 2022-2023

Д.Золбоо

МТ-н салбарын багш

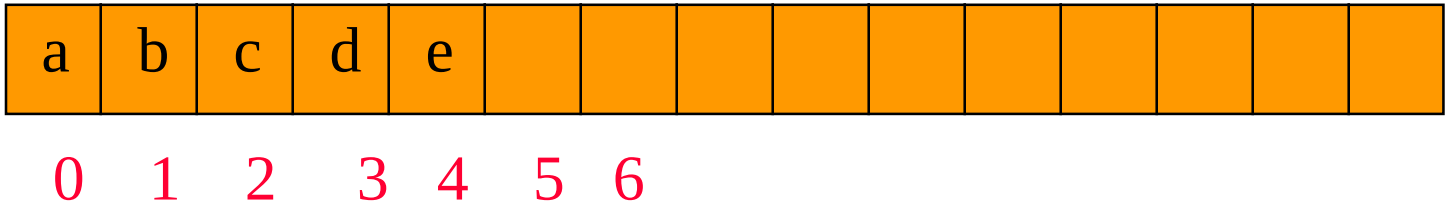
Стек

```
public interface Stack
{
    public boolean empty();
    public Object peek();
    public void push(Object theObject);
    public Object pop();
}
```

Linear List классаас уламжлах

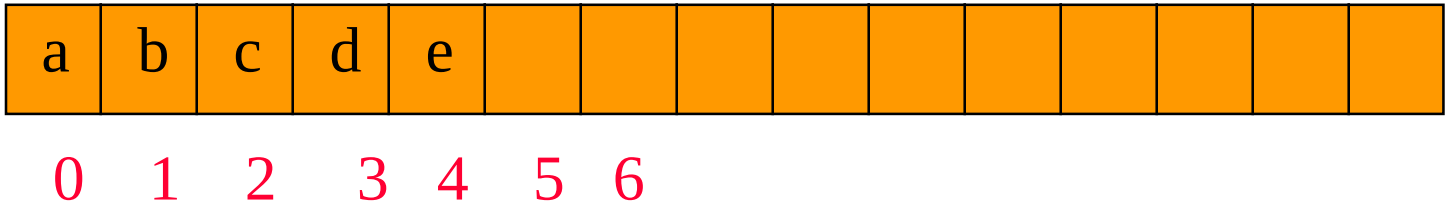
- ArrayLinearList
- Chain

ArrayLinearList –ээс уламжлах



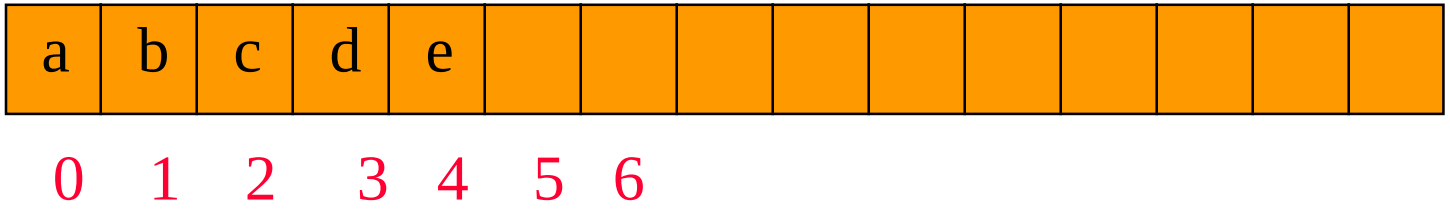
- стекын орой нь шугаман жагсаалтын зүүн, баруун төгсгөлийн нэг байна
- empty() => isEmpty()
- peek() => get(0) эсхүл get(size() - 1)

ArrayLinearList –ээс уламжлах



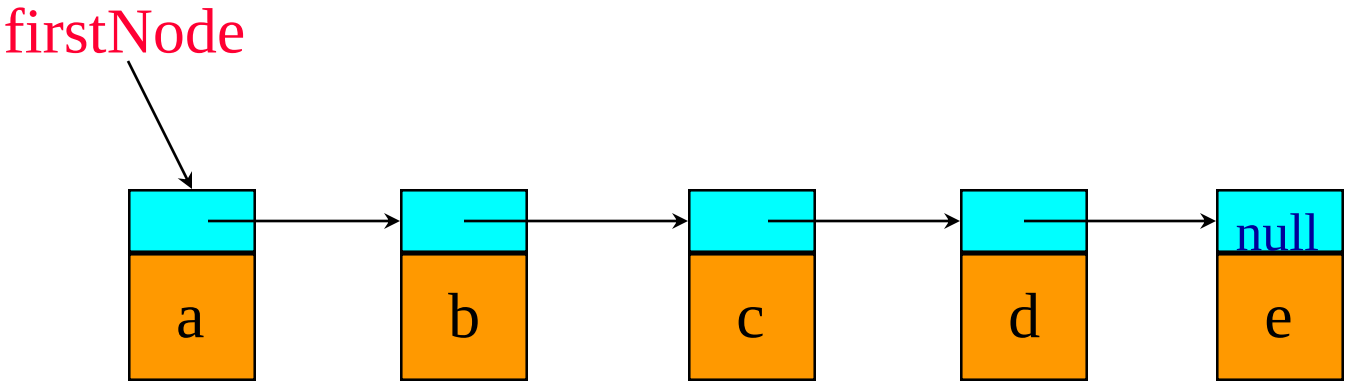
- шугаман жагсаалтын зүүн төгсгөл нь орой бол
  - push(theObject) => add(0, theObject)
  - pop() => remove(0)

ArrayLinearList –ээс уламжлах



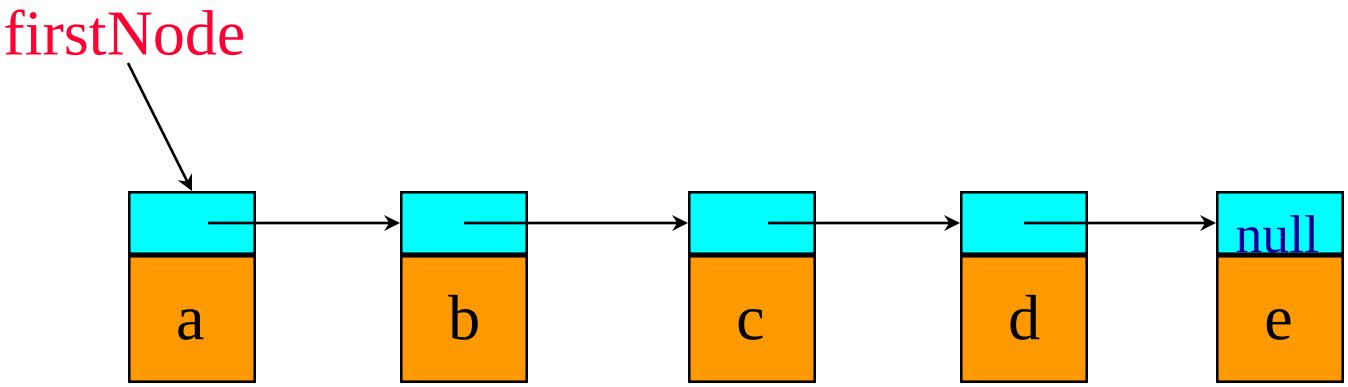
- шугаман жагсаалтын баруун төгсгөл орой бол
  - push(theObject) => add(size(), theObject)
  - pop() => remove(size()-1)
- жагсаалтын баруун төгсгөлийг стекын орой болгож ашиглая

Chain –ээс уламжлах



- Шугаман жагсаалтын зүүн, баруун төгсгөлийн нэг стекын орой
- empty() => isEmpty()

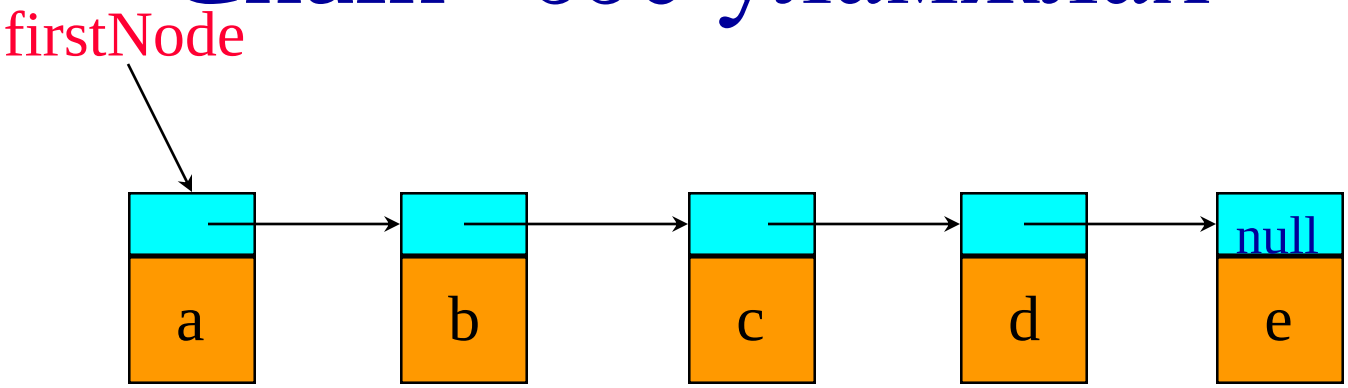
Chain –ээс уламжлах



– шугаман жагсаалтын зүүн төгсгөл орой бол

- peek() => get(0)
- push(theObject) => add(0, theObject)
- pop() => remove(0)

Chain –ээс уламжлах



– шугаман жагсаалтын баруун төгсгөл орой бол

- peek() => get(size() - 1)
- push(theObject) => add(size(), theObject)
- pop() => remove(size()-1)
- жагсаалтын зүүн төгсгөлийг стекын орой болгож ашиглая

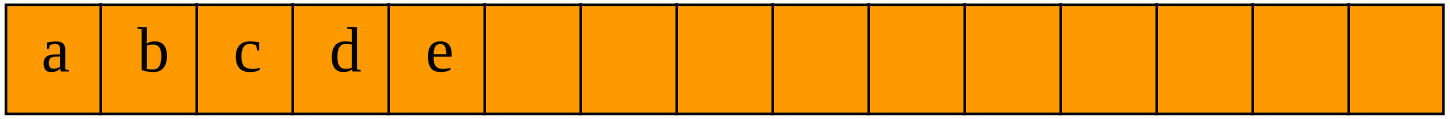
ArrayLinearList –ээс уламжлах

```
package dataStructures;
import java.util.*;  // онцгой тохиолдолд

public class DerivedArrayStack
                extends ArrayLinearList
                implements Stack
{
   // байгуулагчид
   // Stack интерфейсийн аргууд
}
```

Байгуулагчид

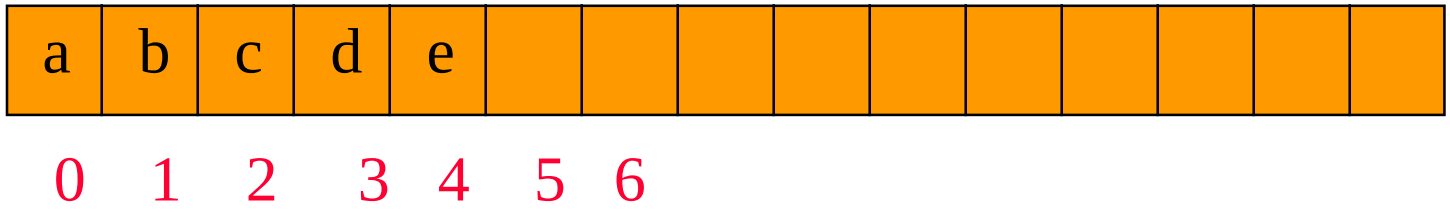
```
/** тодорхой багтаамжтай стек
    байгуулах */
public DerivedArrayStack(int initialCapacity)
  {super(initialCapacity);}

/** 10 –ын багтаамжтай стек байгуулах*/
public DerivedArrayStack()
  {this(10);}
```

empty() ба peek()




```
public boolean empty()
  {return isEmpty();}


public Object peek()
{
  if (empty())
     throw new EmptyStackException();
  return get(size() - 1)
}
```

push(theObject) ба pop()




```
public void push(Object theElement)
  {add(size(), theElement);}


public Object pop()
{
  if (empty())
     throw new EmptyStackException();
  return remove(size() - 1);
}
```

Дүгнэлт

- ArrayLinearList –ээс удамшихын давуу тал
  - Удамшсан классын код маш энгийн, хөгжүүлэхэд амар.
  - Кодыг бага зэрэг зүгшрүүлэх.
  - Стекыг холбоосоор хэрэгжүүэх кодыг амархан гарган авч болно.
    - extends ArrayLinearList -г extends Chain –ээр солино
    - Бүтээмжийг дээшлүүлэх үүднээс жагсаалтын зүүн төгсгөлийг орой болгохоор өөрчлөлт оруулах хэрэгтэй.

Дутагдал

- ArrayLinearList -ийн бүх public аргуудыг стект хэрэглэж болно.
  - get(0) … ёроолын элементийг авах
  - remove(5)
  - add(3, x)
  - Тэгэхээр жинхэнэ стекын хэрэгжилт биш болж байна.
  - Хэрэглэгдэхгүй аргуудыг давхар ачаалах ёстой.

```
public Object get(int theIndex)
  {throw new UnsupportedOperationException();}
```

Өмнө ашигласан get(i) -г super.get(i) болгож өөрчилнө

Дутагдал

- Код шаардлагагүй ажлыг хийдэг.
  - peek() стек хоосон эсэхийг get дуудахаас өмнө шалгадаг. Тэгэхээр индексийг шалгах get шаардлагагүй болж байна.
  - add(size(), theElement) индексийг шалгадаг, орохгүй for давталт орсон. Аль нь ч хэрэггүй.
  - pop() болохоор remove –г дуудахаас өмнө стек хоосон эсэхийг шалгадаг. remove индекс шалгаж, орохгүй for давталттай. Аль нь ч хэрэггүй.
  - Иймд код хэрэгцээгүй удаан ажиллана.

Дүгнэлт

- Кодыг шинээр бичсэнээр хурдан ажиллах боловч хөгжүүлэх гэж хугацаа зарна.
- Програм хангамж хөгжүүлэх зардал, чанарын үзүүлэлт хоёроос сонгох хэрэгтэй.
- Зах зээл гарах хугацаа, чанарын үзүүлэлт хоёроос сонгох хэрэгтэй.
- Анхны кодыг амархан хийгээд, дараа нь чанарын үзүүлэлтийг сайжруулах.

Хурдан pop()

```
if (empty())
     throw new EmptyStackException();
  return remove(size() - 1);
```

оронд нь

```
try {return remove(size() - 1);}
catch(IndexOutOfBoundsException e)
   {throw new EmptyStackException();}
```

Эхнээс нь кодчилох

- 1D stack массив ашиглах, төрөл нь Object.
  - ArrayLinearList –д массив element ашигласан шиг
- int top хувьсагч ашиглах
  - Стекын элементүүд stack[0:top] -д
  - Оройн элемент нь stack[top].
  - Ёроолын элемент нь stack[0].
  - top = -1 бол стек хоосон
  - Стек дэх элементийн тоо top+1.

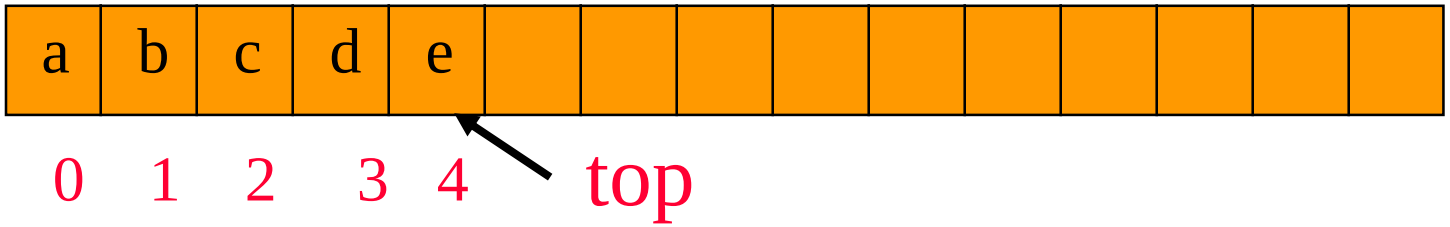
Эхнээс нь кодчилох

```
package dataStructures;
import java.util.EmptyStackException;
import utilities.*;  // ChangeArrayLength
public class ArrayStack implements Stack
{
   // өгөгдөл гишүүд
   int top;                  // Стекын орой
   Object [] stack;  // Элементийн массив
   // байгуулагчид
   // Stack интерфейсийн аргууд
}
```

Байгуулагчид

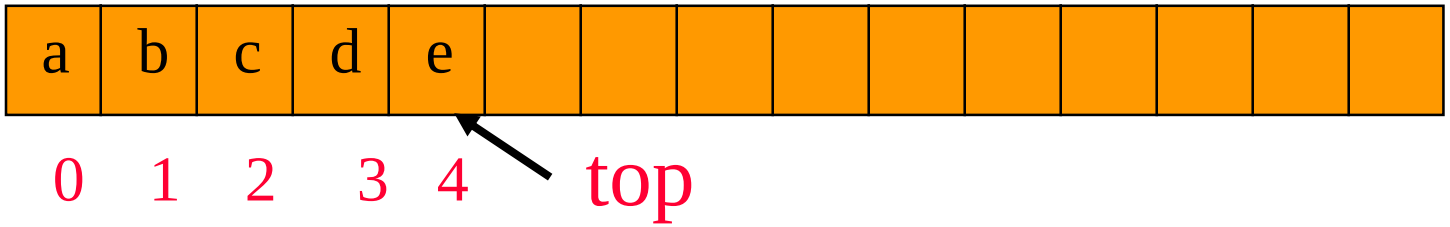
```
public ArrayStack(int initialCapacity)
{
  if (initialCapacity < 1)
     throw new IllegalArgumentException
      ("initialCapacity must be >= 1");
  stack = new Object [initialCapacity];
  top = -1;
}
public ArrayStack()
  {this(10);}
```

push(…)




```
public void push(Object theElement)
{
  // хэрэгтэй бол массивын хэмжээг ихэсгэх
   if  (top == stack.length - 1)
       stack = ChangeArrayLength.changeLength1D
               (stack, 2  * stack.length);
  // theElement стекын оройд хийх
  stack[++top] = theElement;
}
```

pop()




```
public Object pop()
{
  if (empty())
     throw new EmptyStackException();
  Object topElement = stack[top];
  stack[top--] = null;   // хаягдалд өгөх
  return topElement;
}
```

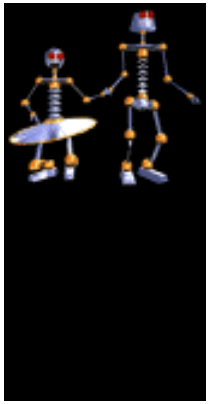
Холбоосон стекыг эхнээс нь кодчилох

- Сурах бичгээс харна уу.

java.util.Stack

- java.util.Vector -аас уламжлагдсан
- java.util.Vector бол шугаман жагсаалтын массив хэрэгжүүлэлт.

Чанарын үзүүлэлт



500,000 pop, push, peek үйлдэл

	анхны багтаамж	
Класс	10	500,000
ArrayStack	0.44s	0.22s
DerivedArrayStack	0.60s	0.38s
DerivedArrayStackWithCatch	0.55s	0.33s
java.util.Stack	1.15s	-
DerivedLinkedStack	3.20s	3.20s
LinkedStack	2.96s	2.96s