CS 240

Data Structures and Algorithms I

Alex Vondrak

ajvondrak@csupomona.edu

October 17, 2011

Stacks

Abstract Data Type

```
interface Stack {
    public void push(int item);
    public int pop()
        throws StackUnderflowException;
    public int top()
        throws StackUnderflowException;
    public boolean isEmpty();
    public int size();
}
```

ArrayStack

```
class ArrayStack implements Stack {
  public void push(int item) { ... }

  public int pop()
     throws StackUnderflowException { ... }

  public int top()
     throws StackUnderflowException { ... }

  public boolean isEmpty() { ... }

  public int size() { ... }
}
```

ArrayStack

Constructor

```
class ArrayStack implements Stack {
  private int[] data;
  private int top;

  public ArrayStack() {
     final int CAPACITY = 10;
     top = -1;
     data = new int[CAPACITY];
  }

  // ...
}
```

ArrayStack

Auxiliary Methods

```
class ArrayStack implements Stack {
  // ...

  public int size() {
     return top + 1;
  }


  public boolean isEmpty() {
     return (size() == 0);
  }

  // ...
}
```

ArrayStack

top()

```
class ArrayStack implements Stack {
   // ...

   public int top() throws StackUnderflowException {
      if (isEmpty())
         throw new StackUnderflowException();
      return data[top];
   }

   // ...
}
```

ArrayStack

pop()

```
class ArrayStack implements Stack {
  // ...

  public int pop() throws StackUnderflowException {
      int result = top();
      top--;
      return result;
  }

  // ...
}
```

ArrayStack

push()

```
class ArrayStack implements Stack {
  // ...

  public void push(int value) {
    if (size() == data.length)
      grow();
    data[++top] = value;
  }

  // ...
}
```

ArrayStack

grow()

```
class ArrayStack implements Stack {
  // ...

  private void grow() {
    final int CAPACITY = 2 * data.length + 1;
    int[] biggerArray = new int[CAPACITY];

    for (int i = 0; i < data.length; i++)
      biggerArray[i] = data[i];

    data = biggerArray;
  }

  // ...
}
```