Stacks

- Reading: Lewis and Loftus, JAVA: Software Solutions, (3rd ed), Chapter 12.2
- Savitch Chapter 10.2

Objectives

- To revise a couple of key points from the last lecture
- To learn how to implement a stack using a linked list
- To develop an alternative implementation of a stack using an array

Class exercise: stacks

- Problem
  - Given a stack s, what is the result of the following operations?

```
s.clear_stack()
s.push(8)
s.push(9)
output s.pop()
output s.peek()
s.push(22)
output s.pop()
if s.isEmpty()  is not true, output s.pop()
```

Linked list implementation

- JAVA declaration

```
class StackNode {
    Object data;
    StackNode next;
    StackNode(Object _d) {... ...}
}

class StackList {
    private StackNode top = null;
    public void push(Object elem) {... ...}
    public Object pop() {... ... }
    public Object peek() {... ...}
    public boolean isEmpty() {... ...}
}
```

Variables are used with package access for this teaching session, normally they should have private access

- The implementation of StackNode

```
public StackNode(Object _d) {
        data = _d;
        next = null;
}
```

Stack operations

- push

```
public void push(Object x) {
        StackNode p = new StackNode(x);

        p.next = top;
        top = p;
}
```

Stack operations (ctd)

- pop

```
public Object pop() {
    if (isEmpty()) { … }
    Object answer = top.data;
    top = top.next;
    return answer;
}
```

Class exercise

- Write JAVA code for the peek method

```
public Object peek() {
    …
    …
}
```

Stack operations (ctd)

- isEmpty

```
public boolean isEmpty() {
    return top == null;
}
```

Conversion to octal - pseudocode

```
method convertToOctal (int n)
   // n  a decimal number to be converted
   Initialize stack
   WHILE n != 0
      divide n by 8 giving a quotient and a remainder
      push the remainder onto the stack
      set n to the quotient
   ENDWHILE
   // Answer is the remainders, in reverse order, so…
   WHILE stack is not empty
      pop a digit from the stack and print it
   ENDWHILE
END method
```

Implementation of decimal to octal conversion algorithm

```
//ConvertToOctal.java
//Assume StackList is a stack of int (ie. substitute Object with int)

public class ConvertToOctal {
  public static void main(String[] args) {
          final  int OCTAL_BASE = 8;
          StackList s = new StackList();
          int n = 427;
          while (n != 0) {
                    s.push (n%OCTAL_BASE);
                    n = n/ OCTAL_BASE;
          }

          System.out.print(“ 427 is equal to ”);
          while (! s.isEmpty ())
                    System.out.print(s.pop ());

          System.out.println(“ in octal”);
  }
}
```

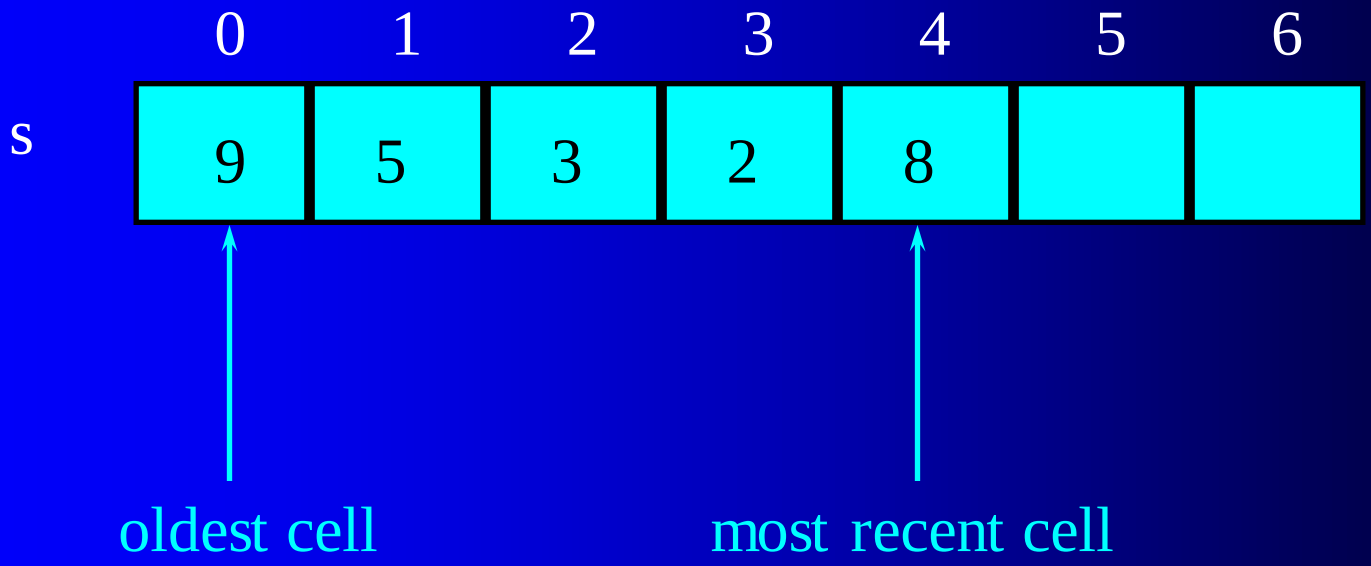
Program Execution

```
%java ConvertToOctal
 427 is equal to 653 in octal
```

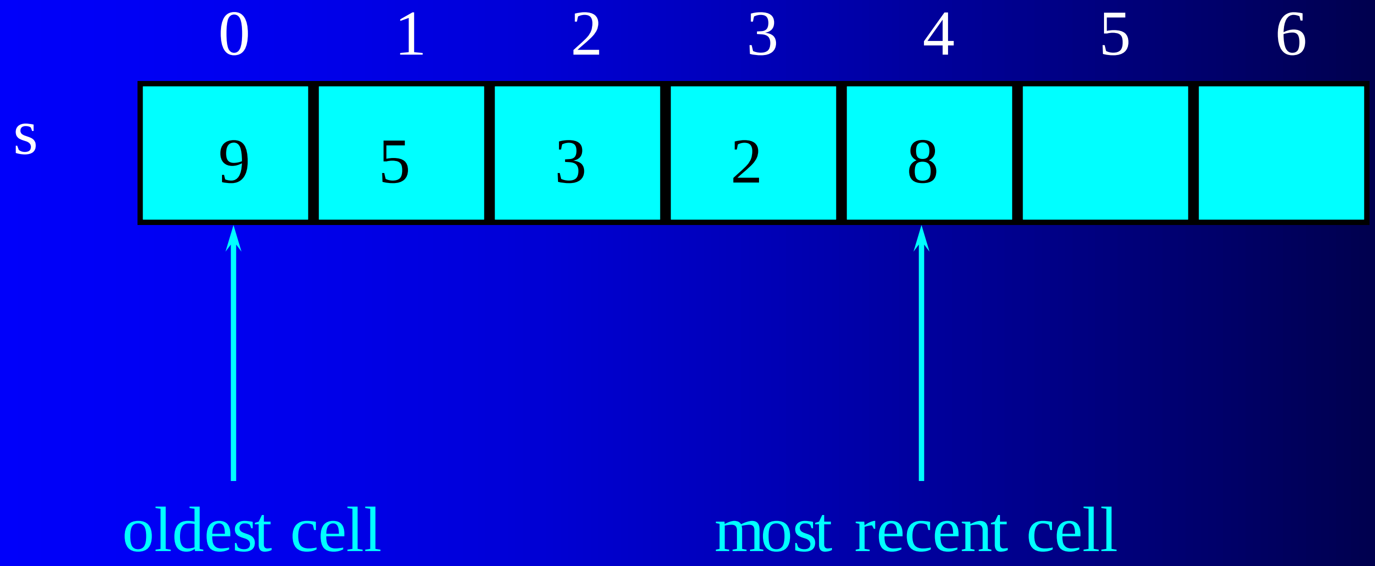
Other stack applications

- Checking for balanced parentheses
- Evaluating postfix expressions
- Infix to postfix conversion
- Recursive methods

Array implementation of a stack

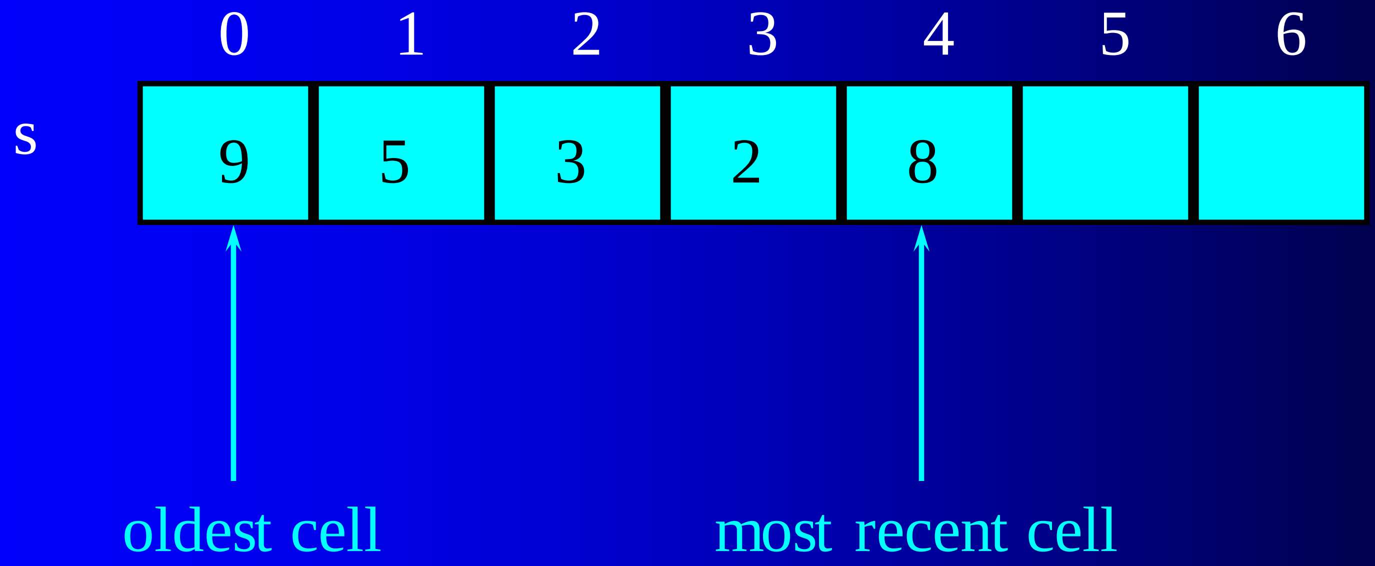


Adding an element



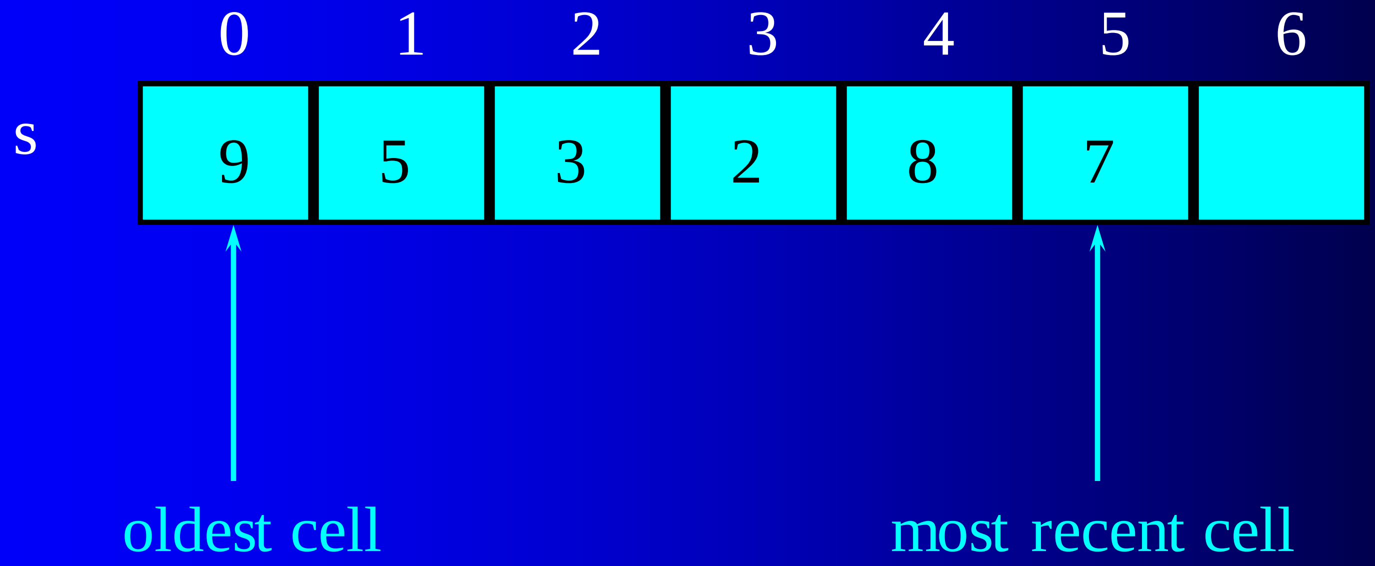
Adding an element

Push 7 on the stack

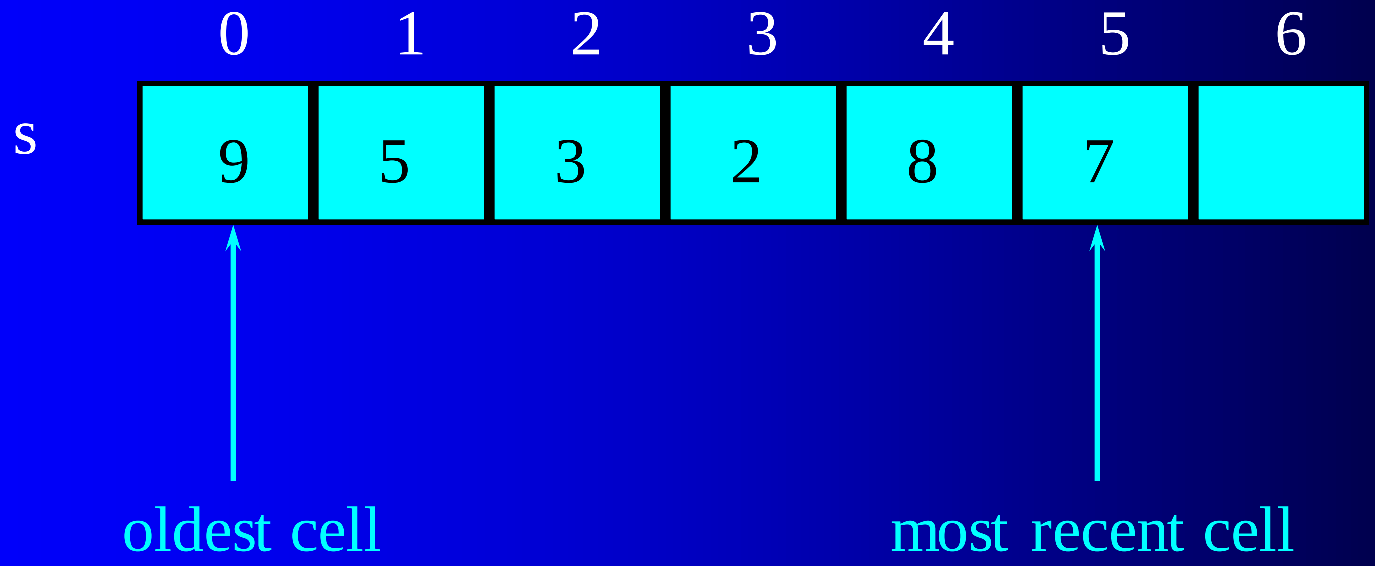


Adding an element

Push 7 on the stack

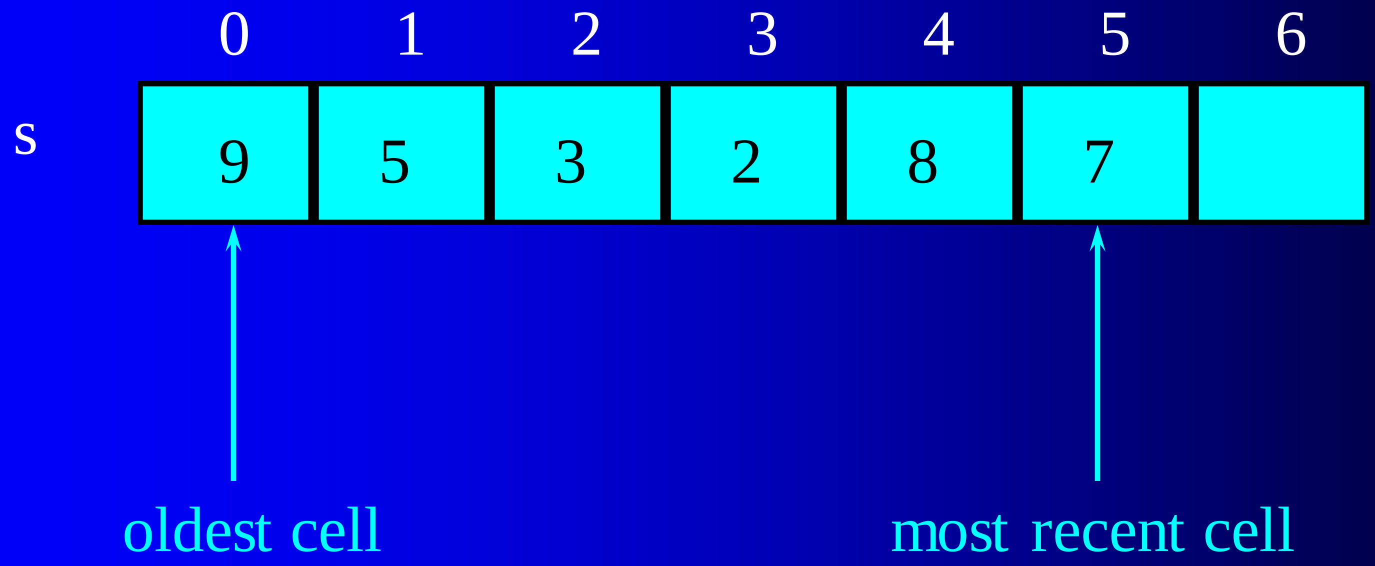


Removing an element

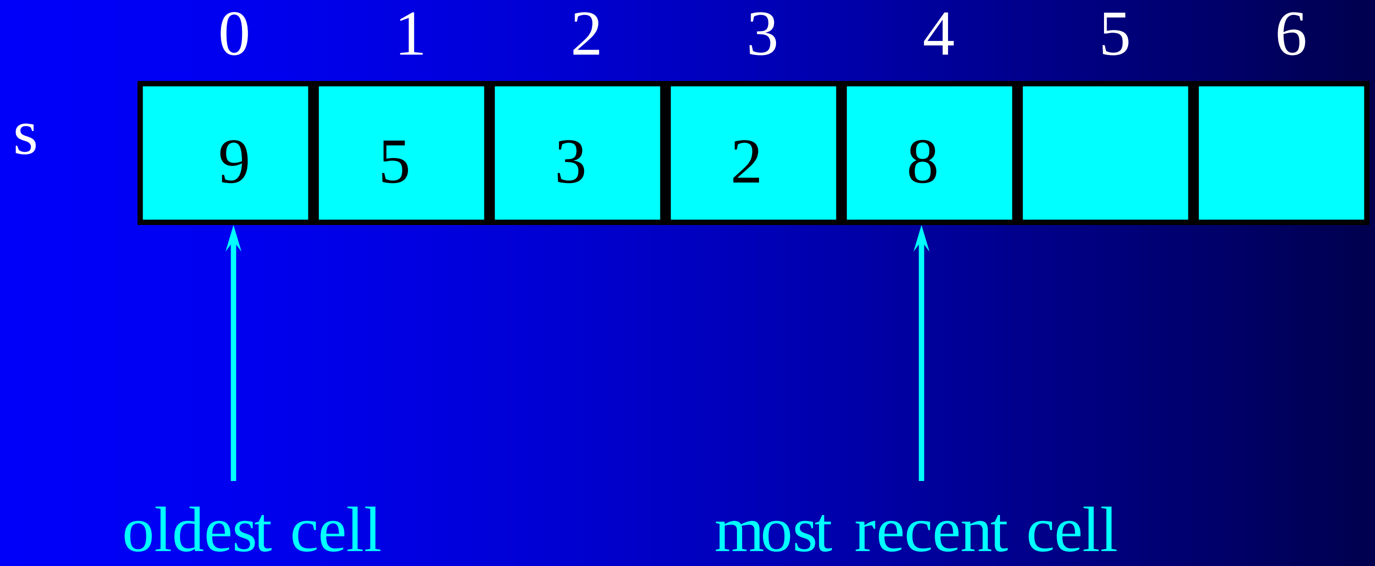


Removing an element

pop the stack



Removing an element



Array implementation

- JAVA declaration

```
public class StackArray {
    private final static int MAX_STACK_SIZE = 50;
    private Object[] storage;
    private int stackSize = 0;
    private int top= -1;
    public StackArray() {...}
    public void push(Object x) { …  }
    public Object pop() { …  }
    public Object peek() { …  }
    public boolean isFull() { …  }
    public boolean isEmpty() { …  }
    public void clearStack() {...}
}
```

Stack operations

- Constructor

```
public StackArray () {
    storage = new Object[MAX_STACK_SIZE];
}
```

Stack operations

- push

```
public void push (Object x) {
     if (isFull()) {
        System.out.println("stack overflow");
        System.exit (1);
    }

    storage [++top] = x;
}
```

Stack operations

- pop

```
public Object pop ()
{
    if (isEmpty()) {
        System.out.println(“Attempt to pop from empty stack”);
        System.exit (1);
    }

    return storage[top--];
}
```

Class exercise

- Write JAVA code for the peek method

```
public Object peek ()
{
    ….
    ….
}
```

Stack operations (ctd)

- isFull

```
public boolean isFull () {
    return top == MAX_STACK_SIZE -1;


}
```

Stack operations (ctd)

- isEmpty

```
public boolean isEmpty () {
   return top == -1;
}
```

Stack operations (ctd)

- clearStack

```
public void clearStack () {
    while (! isEmpty () )
        pop ();

}

//any alternative implementation?
```

Implementation of decimal to octal conversion algorithm

```
//ConvertToOctal.java
//Assume StackArray is a stack of int (ie. Substitute Object with int)

public class ConvertToOctal {
   public static void main(String[] args) {
		final  int OCTAL_BASE  = 8;
		StackArray s = new StackArray();
		int n = 427;
		while  (n != 0) {
				s.push (n % OCTAL_BASE);
				n = n / OCTAL_BASE;
		}

		System.out.print(“ 427 is equal to ”);
		while  (! s.isEmpty ())
				System.out.print(s.pop  ());

		System.out.println(“  in octal”);
   }
}
```

Implementation of decimal to octal conversion algorithm

- Identical to the method for the linked list implementation!
- See slide 11

The Stack class

- The Stack class is defined in *java.util*.
- It contains methods such as

```
boolean empty()
Object peek()
Object pop()
Object push(Object item)
int search(Object o) //returns the position of the
//element. The top element is in position 1.
```

- Stack is a subclass of Vector, therefore we can push objects of different classes into a stack.

Example

```
import java.util.*;
public class StackApp
{
    public static void main (String[] args)
    {
        Stack s = new Stack();
        int i = 0;
        while (i++ < 10)
        {
            if (i % 2 == 0)
                s.push(i);
            else
                s.push(i+ 0.5);
        }
        System.out.println("The top element is: " + s.peek());
        System.out.println("The position of 4 is: " + s.search(4));
        System.out.print("The stack contains: ");

        while(!s.empty())
        {
            System.out.print(s.pop() + " ");
        }
        System.out.println();

    }
}
```

Program execution

```
% java StackApp
The top element is: 10
The position of 4 is: 7
The stack contains: 10 9.5 8 7.5 6 5.5 4 3.5 2 1.5
```