# Overview of ArrayStack

Pearson

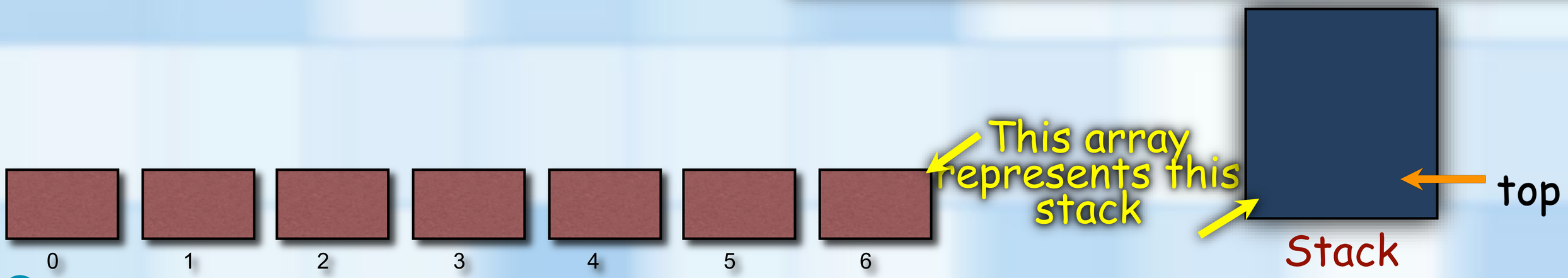
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class ArrayStack

- Implementing the ADT Stack
  - Using an array
- Which end is the top of the stack?
  - Last occupied location in the array

```
/* @file ArrayStack.h */
template<class ItemType>
class ArrayStack : public StackInterface<ItemType>
{
private:
   static const int DEFAULT_CAPACITY = 5;
      ItemType items[DEFAULT_CAPACITY]; // Stack items
      int      top;                     // Top of stack
public:
  // method headers here
};

/* @file ArrayStack.cpp */
template<class ItemType>
ArrayStack<ItemType>::ArrayStack() : top(-1)
{
}  // end default constructor
```




Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class ArrayStack isEmpty

- Implementing the ADT Stack
  - Using an array
- Which end is the top of the stack?
  - Last occupied location in the array

```
template<class ItemType>
bool ArrayStack<ItemType>::isEmpty() const exempt
{
   return top < 0;
}  // end isEmpty
```



Pearson

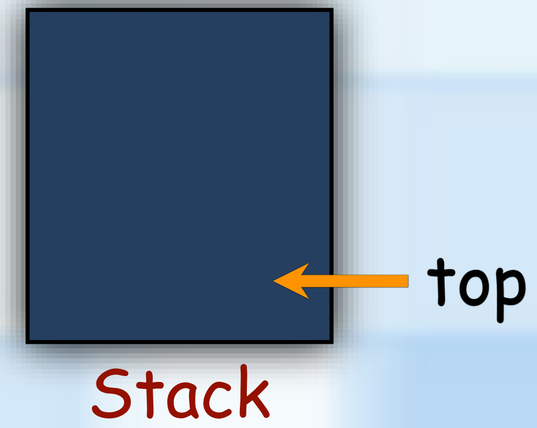
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class ArrayStack push

- Implementing the ADT Stack
  - Using an array
- Which end is the top of the stack?
  - Last occupied location in the array

```
template<class ItemType>
bool ArrayStack<ItemType>::push(const ItemType& someItem) noexcept
{
    bool result = false;
  // Does stack have room for someItem?
    if (top < (DEFAULT_CAPACITY - 1))
    {
       top++;
       items[top] = someItem;
       result = true;
    }  // end if

    return result;
}  // end push
```

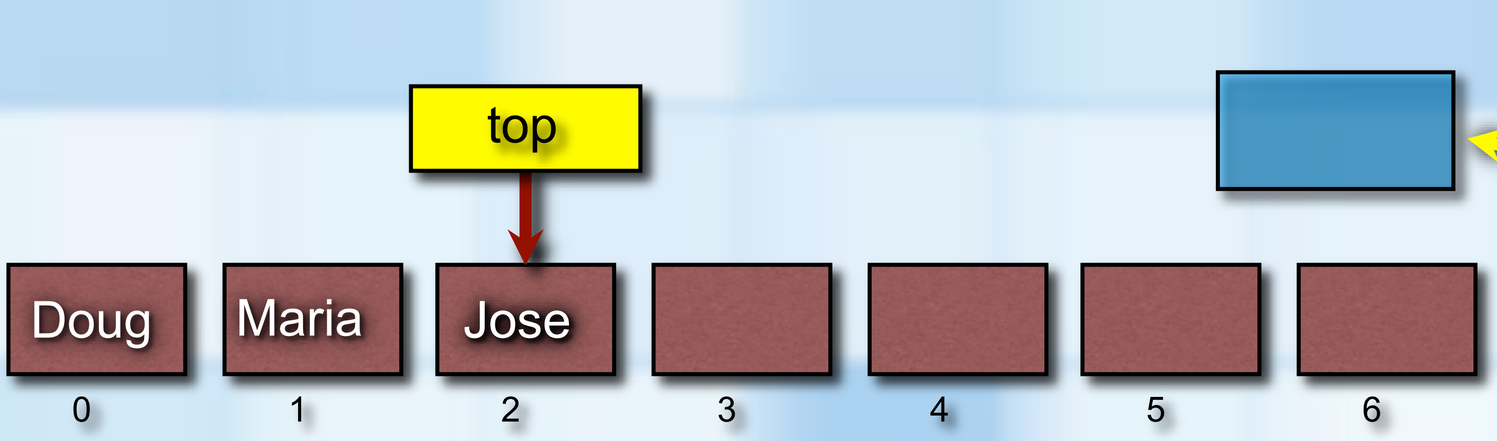

Pearson

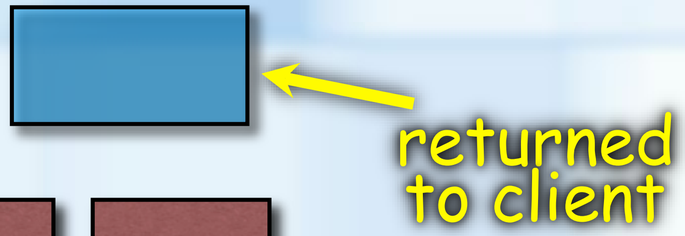
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

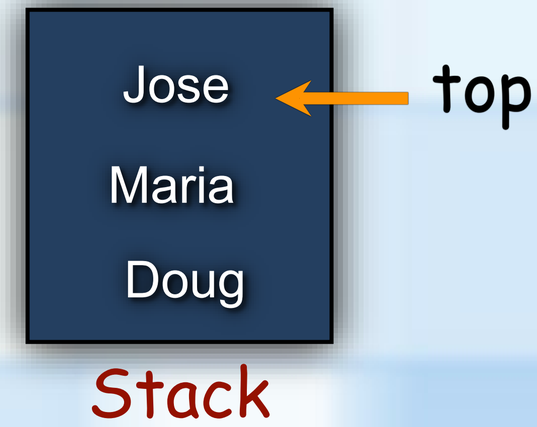
# The Class ArrayStack peek

- Implementing the ADT Stack
  - Using an array
- Which end is the top of the stack?
  - Last occupied location in the array

```
template<class ItemType>
ItemType ArrayStack<ItemType>::peek() const
{
   if (isEmpty()) // Enforce precondition
  {
            throw PrecondViolatedExcept("Stack is empty!")
  }
   return items[top];
}  // end peek
```



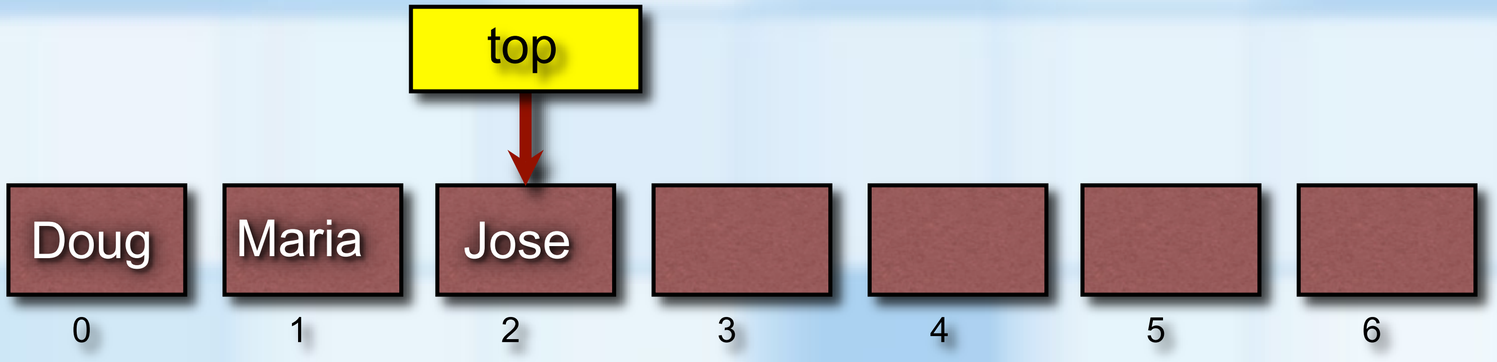

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class ArrayStack pop

- Implementing the ADT Stack
  - Using an array
- Which end is the top of the stack?
  - Last occupied location in the array

```
template<class ItemType>
bool ArrayStack<ItemType>::pop()
{
    bool result = false;
    if (isEmpty())
  {
    throw PrecondViolatedExcept("Stack is empty!")
  }
  else
    {
        result = true;
        top--;
     }  // end if
     return result;
}  // end pop
```





Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# Overview of LinkedStack

Pearson

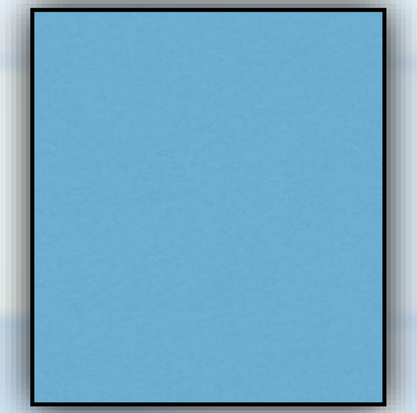
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class LinkedStack

- Implementing the ADT Stack
  - Using a linked chain
- Which end is the top of the stack?
  - Using the head is fast

LinkedStack.h

```
/* @file LinkedStack.h */
template<class ItemType>
class LinkedStack : public StackInterface<ItemType>
{
private:
   Node<ItemType>* topPtr;
public:
  // constructors
  // destructor
  // method headers from StackInterface
};
```

Stack

Pearson

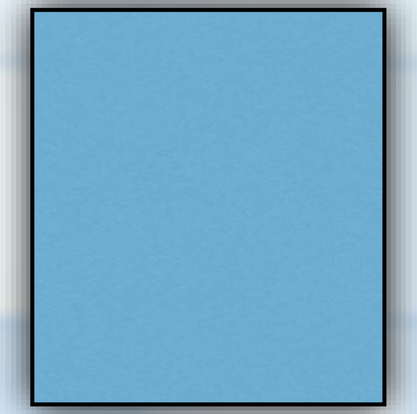
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class LinkedStack

- Implementing the ADT Stack
  - Using a linked chain
- Which end is the top of the stack?
  - Using the head is fast

LinkedBag.cpp

```
/* @file LinkedStack.cpp */
template<class ItemType>
LinkedStack<ItemType>::LinkedStack() : topPtr(nullptr)
{ }  // end default constructor
```

Stack

Pearson

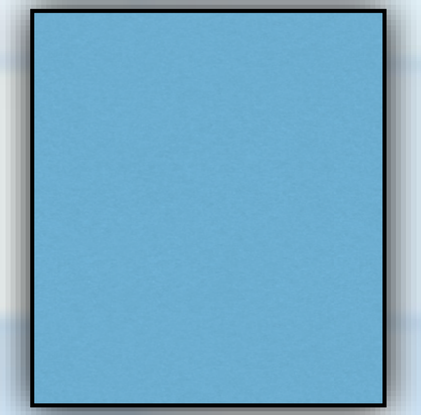
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class LinkedStack isEmpty

- Implementing the ADT Stack
  - Using a linked chain
- Which end is the top of the stack?
  - Using the head is fast

LinkedBag.cpp

```
template<class ItemType>
bool LinkedStack<ItemType>::isEmpty() const noexcept
{
   return topPtr == nullptr;
}  // end isEmpty
```

Stack

Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

# The Class LinkedStack push

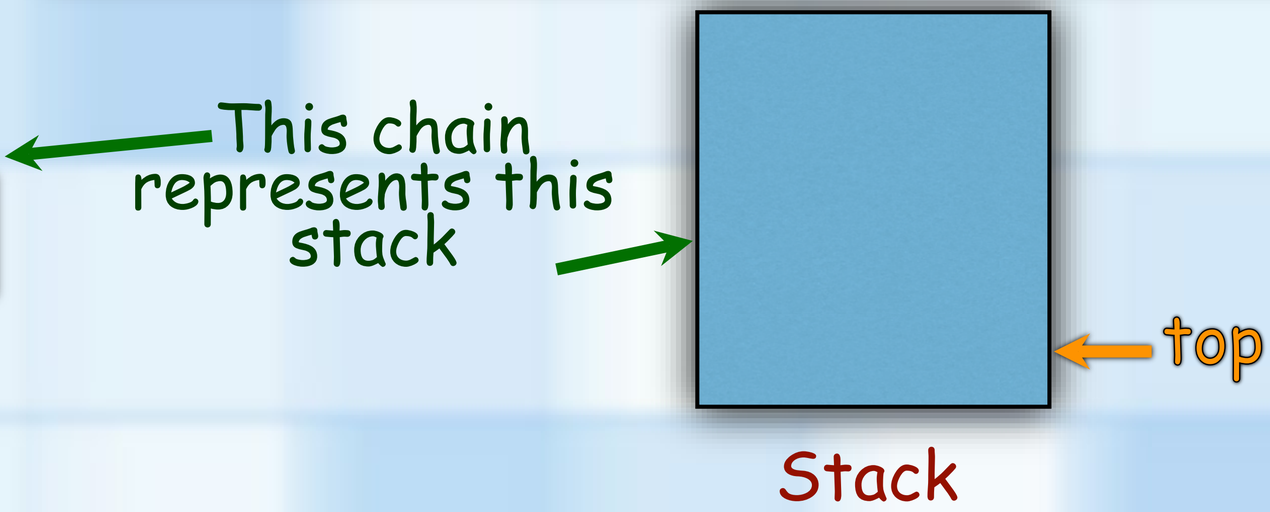
- Implementing the ADT Stack
  - Using a linked chain
- Which end is the top of the stack?
  - Using the head is fast

LinkedBag.cpp

```
/* @file LinkedStack.cpp */
template<class ItemType>
bool LinkedStack<ItemType>::push(const ItemType& someItem)
{
   auto itemNodePtr = new Node<ItemType>(someItem, topPtr);

   topPtr = itemNodePtr;

    return true;
}  // end push
```


Pearson

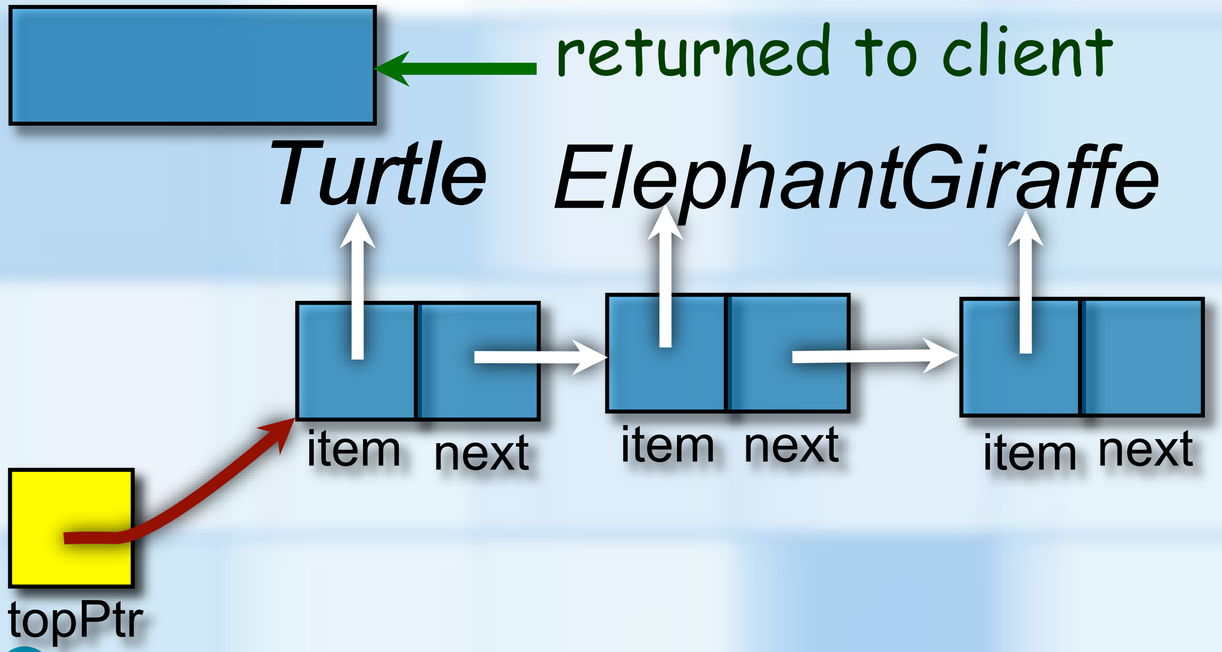
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

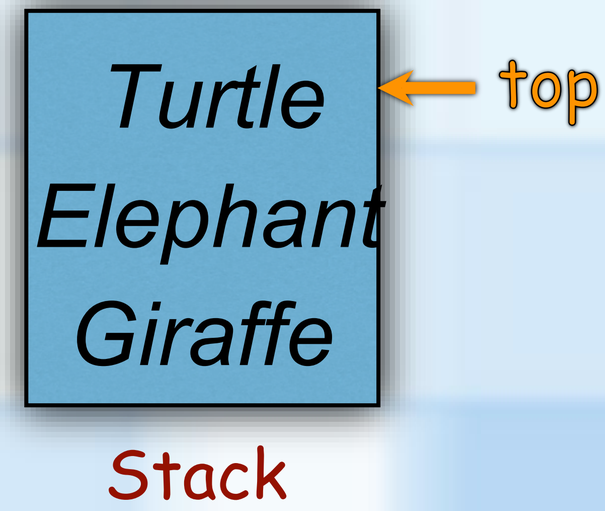
# The Class LinkedStack peek

- Implementing the ADT Stack
  - Using a linked chain
- Which end is the top of the stack?
  - Using the head is fast

LinkedBag.cpp

```
template<class ItemType>
ItemType LinkedStack<ItemType>::peek() const
{
   if (isEmpty()) // Enforce precondition
   {
            throw PrecondViolatedExcept("Stack is empty!")
   }
   return topPtr->getItem();
}  // end getTop
```




Pearson

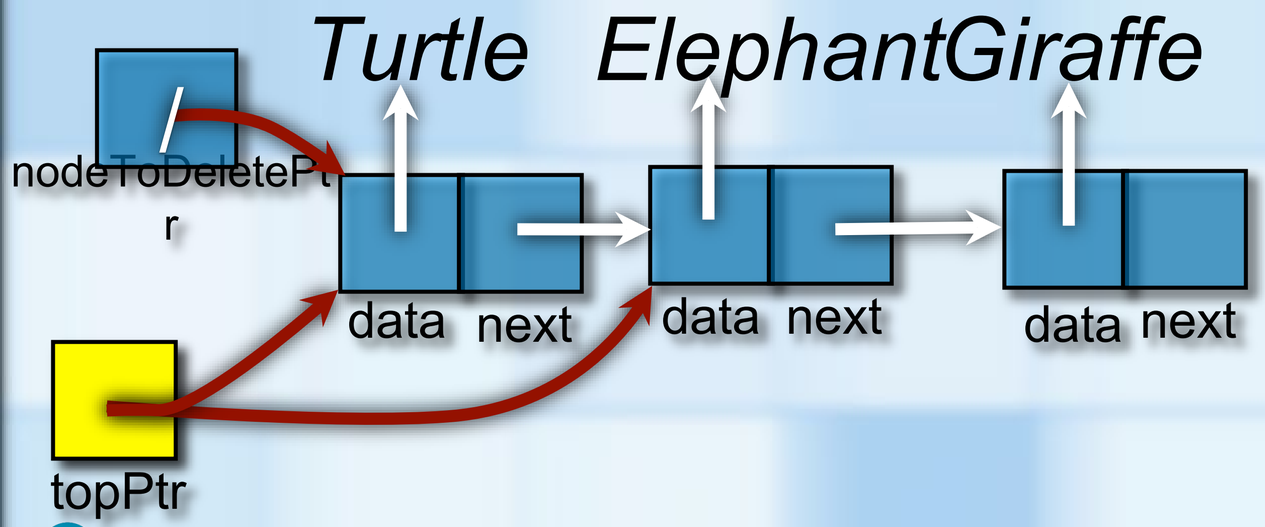
Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved

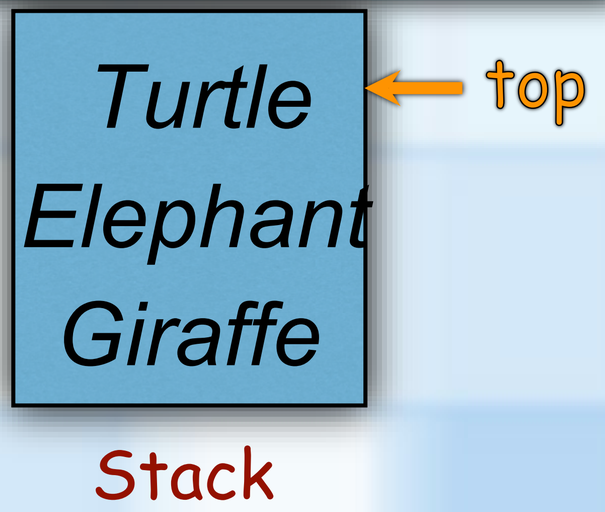
# The Class LinkedStack pop

- Implementing the ADT Stack
  - Using a linked chain
- Which end is the top of the stack?
  - Using the head is fast

```
template<class ItemType>
bool LinkedStack<ItemType>::pop()
{
    bool result = false;
    if (!isEmpty())
    {
     auto nodeToDeletePtr = topPtr;
     topPtr = topPtr->getNext();

     delete nodeToDeletePtr;
     nodeToDeletePtr = nullptr;
     result = true;
    }  // end if
    return result;
}  // end pop
```




Pearson

Copyright © 2025 Pearson Education, Hoboken, NJ. All rights reserved