



Linking Structures

Part 03

Linking Structures

- Groups Together
 - Data
 - Link(s) / Reference(s) / Pointer(s)
 - “Node”
- Pros
 - Growable
 - Shrinkable
- Cons
 - No Random Access

Node



List of Nodes

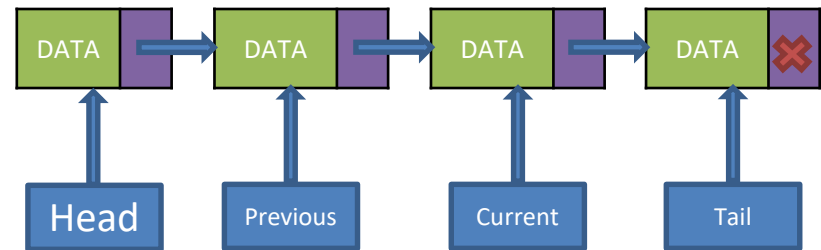


Linked Lists

Linked Lists

- Nodes Contain
 - Data
 - Link
- Special Nodes
 - Head: Always points to the first element of the list
 - Tail: Always points to the last element of the list
 - Current: Movable pointer used to Access and Modify Data in the List
 - Previous: Always stays on node behind Current
- Certain Linked Lists may omit some of these Nodes

Linked List



Generics

- Generics
 - “Variables for Types”
 - Spoken: “This is a class of <<types>”
- In Java the Generic type must be an Object-Type
 - Everything in Java is assumed to inherit from type “Object”

Syntax

```
public class <<class identifier>> < <<Generic Type>> >
{
}
}
```

Example

```
public class GenLL <T>
{
}
}
```

Generics

- Change the type for the data to “T”
 - All functionality previously described works in the exact same way
 - Only difference being the type
- “T” is always an Object-Type in Java
 - The “==” and “!=” should only be used to refer to memory addresses
 - All Objects are assumed to have a “.equals(Object)” method in Java
 - All Objects are assumed to have a “.toString()” method

Example

```
public class GenLL <T>
{
    private class ListNode
    {
        T data;
        ListNode link;
        public ListNode(T aData, ListNode aLink)
        {
            data = aData;
            link = aLink;
        }
    }
}
```

Example

Problem



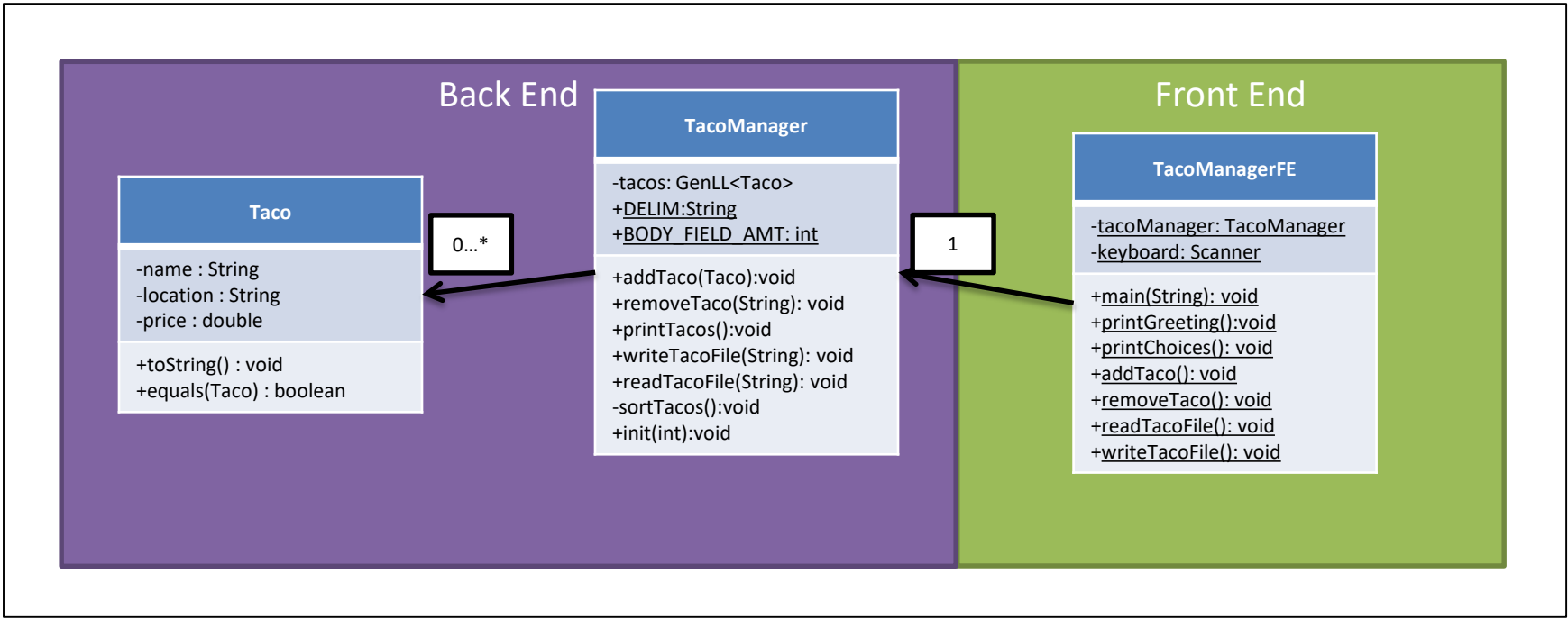
Requirements

- Keep Track of important Taco Information
- Taco's Information
 - Name
 - Location
 - Price

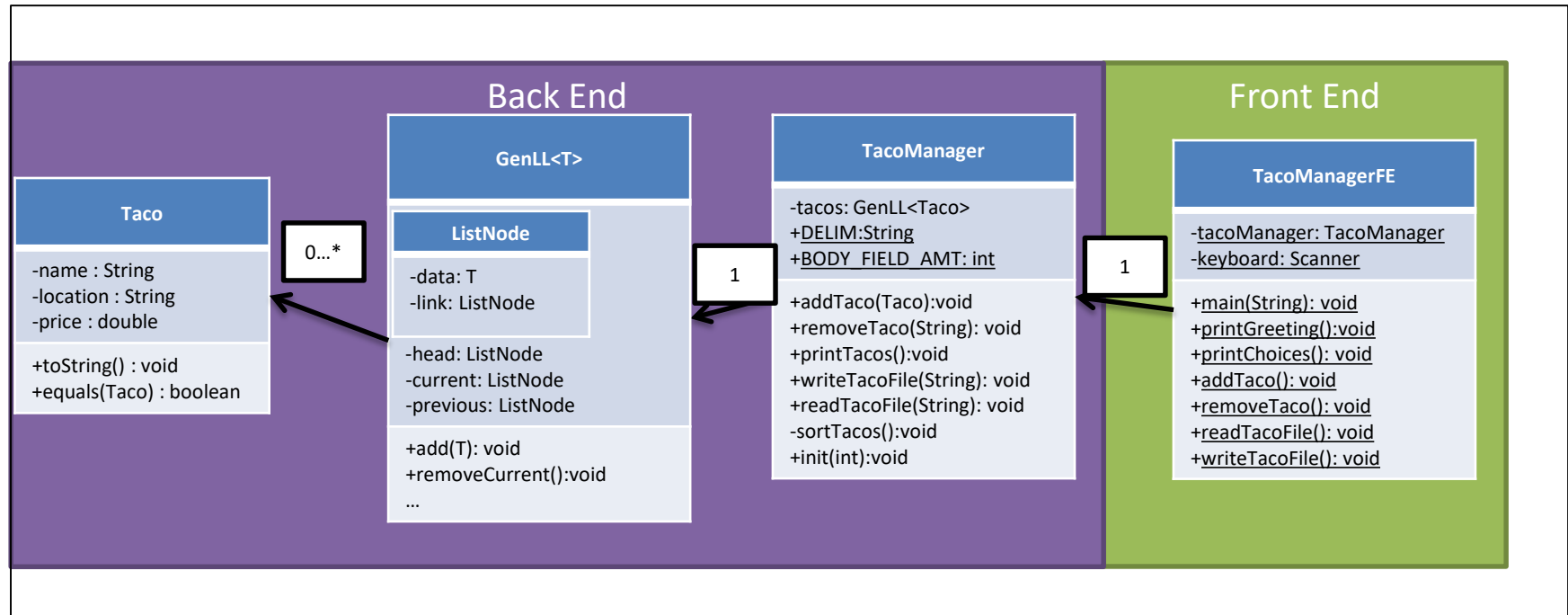


- Should be able to
 - Add a Taco
 - Remove a Taco by Name
 - Sort by Price
 - Display all Taco information
 - Store in a Taco File
 - Read from Stored Taco Files
- Clear and Simple Front End

Design



Design



Linking Structures