PA1 – Manager Prototype

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: (•) Yes ( ) No

Name: Joey Domino

Date: 1/22/2022

Submission Details

Final ***Changelist*** number: 128050

Verified build: (•) Yes ( ) No

Required Configurations: DEBUG only

Tests Passed: 12

Discussion (What did you learn):

I learned:
- How to set up DLinks
- Rehash on DLL's

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.csproj, *.cs,
  - App.config, AssemblyInfo.cs, CleanMe.bat
  - Resources Directory:
    - *.tga, *.dll, *.wav, *.glsl, *.azul

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Points will be deducted if minimum is not reached

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level 4
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work “as-is”.
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
  - It will not be graded and you get a 0

No Containers

- Containers (No automatic containers or arrays
- Template or generic parameters
- No arrays
  - You need to do this the old fashion way - ***YOU EARNED IT***

Leave Project Settings

- Do NOT change the project or warning level
  - Any changing of level or suppression of warnings is an integrity issue

Simple C#

- No .Net
- We are using the basics
  - Types:
    - Class, Structs, intrinsic types (int, float, bool, etc…)
    - NO arrays allowed!
  - Basics language features
    - Inheritance, methods, abstract, virtual, etc…

No Debug code or files disabled

- Make sure the program has only active code
  - If you added debug code or commented out code,
    - please return to code to active state or remove it

Adding files to this project

- Make sure you add the files in the appropriate sub-directories
- Make sure any new files are successfully integrated into the project
- Make sure your new files are submitted to Perforce

Due Dates

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
  - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
  - Fill out the form and discussion for full credit.

Goals

- Learn
  - Understand the class environment
    - Practice using C# language
    - Practice using Microsoft IDE
    - Practice using Perforce
  - Linked List
    - Starting creating a prototype of our manager

Assignments

Grading:

- Linked List Demo
  - Add to Front of list
  - Add to End of list
  - Add Node pooling
    - Active / Reserve list to hold nodes
    - Keeping track of stats (current Active, current Reserve, Growth, total)
  - Add searching by Name

- Memory Pooling
  - Manager has both an Active and Reserve List
  - Nodes are initially added to reserve (new() create nodes)
  - Add a node is removed from Reserve and transferred to Active list (no new() allowed)
  - Remove node, node is removed from Active list and added to Reserve list

General guidelines:

- You can add more methods and functionality
  - There is the minimum for the 1st pass of this manager
- Start to think about what features you need
  - It's a prototype
- Write code to test and run your manager
- Create UML diagrams to help
- Post on Piazza questions and clarifications

Required Features

Manager

- Create a Manager
  - Manager actually has 2 linked list attached an Active List and a Reserved List
    - essentially 2 head pointers
  - All nodes are double linked lists
- Manager(…) - Initialized you manager with a reserved size and growth size
  - Example, say you initialize the reserve size with 5 nodes, growth size is 2 nodes
    - You will create on construction of the manager
      - 5 linked nodes attached to the reserved list.
      - You set the growth size member variable to 2
      - The Active list will have nothing attached only null pointer.
- Generally
  - Adding a node
    - You move a node from the reserved list and add it to the active list
      - Insert the node to the front of the list
        - In front for speed…. Think about it O(1)
      - Likewise for adding to End of list
    - If the reserved list is empty
      - Refill the reserve list with the growth size
      - The reserved list will be filled with that many nodes, then you can move one from the reserve and transfer it to the active list
        - (might seem unnecessary to move to reserve then immediately transfer to active… but it reduces complexity)
  - Remove a node
    - Remove any given node by address
      - Its double link so it should be O(1) in complexity
    - Add it back to the Reserve List
      - To the front of the list O(1) complexity
- Add(...) - creates and return node
  - moves one from the reserved list and inserts the node to the active list
- Remove(...) - recycles the node
  - removes the node from active list and transfers it to the reserved list
- Ability to report stats – data we can use
  - number of active, reserved, growth number, total number, etc...
- Find(...) - searches through active list and returns node
  - Search by Name
    - Hint – you'll need to have casting to make this possible

DataNode

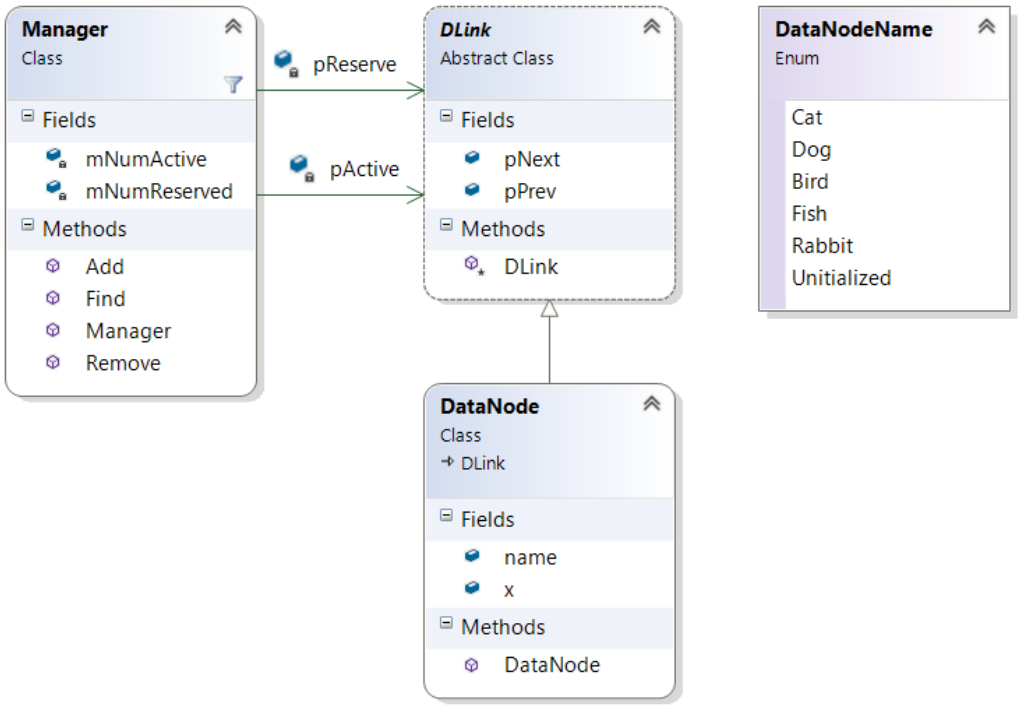
- Holds data (int x)
- Holds name (enumeration)

DLink

- Holds the next and prev references
- Data Node derives from DLink
- Used for implementing in terms of… paradigm

Super simple UML (just high level idea):

- Your solution will probably be more complicated:



Make sure you delete these using directives (we are not using them)

- `using System.Collections.Generic;`
- `using System.Linq;`
- `using System.Text;`
- `using System.Threading.Tasks;`

I created a Batch file to remove temporary files and temporary directory creation

- Ask on Piazza how to do that – if you don't understand it
- I call it, CleanMe.bat

Development

- Store project in student directory in perforce
- Do your work in the supplied PA1 project
    - Feel free to add new files and methods to the project

Submission

- Submit your PA1 directory into perforce:
    - /student/<yourname>/PA1/...
        - You need to submit a complete C# project
    - Solution, project and C# files (whatever it takes to build the project)
        - Do not submit anything that is auto generated
    - Run the supplied CleanMe.bat before submission
        - Should cleanup files
- Fill out the Submission report and submit that pdf to your student directory

Validation

Simple checklist to make sure that everything is submitted correctly

- Is the project compiling and running without any errors or warnings?
- Does the project runs ***ALL*** without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
    - Is all the code there and compiles “as-is”?
    - No extra files

Hints

Most assignments will have hints in a section like this.

- Do this assignment by iterating and slowly growing your project
    - Do a simple linked list – add/remove
    - Add printing to help you verify
    - Add asserts to verify in runtime the behavior
- Debugging is the key to this program
    - Learn how to debug, add functions to make it easy to debug and track
    - Get comfortable with Microsoft IDE – it does a lot of stuff
- Search Google for C# spec and language questions

Troubleshooting

- Print, print, print
    - Draw diagrams to help you understand
- Have fun… this shouldn’t be stressful

- Slow and steady discovery and development will get you there.

- I would create tests cases
  - Just on paper – some use case scenarios
  - For example.
    - A)
      - Initialize Manager(5,2)
      - Add – two nodes (one at a time)
      - Remove – two node
      - Validate the data
    - B)
      - Initialize Manager(5,2)
      - Add – 7 nodes (one at a time)
      - Remove – 3 nodes
      - Validate the data
    - Etc…