PA2 – Adaptor,Iterator,Singleton,Factory

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: 2/3/2022

Submission Details

Final ***Changelist*** number: 130325

Verified build: (•) Yes ( ) No

Required Configurations: DEBUG only

Test Passed: 10

Discussion (What did you learn):

I learned:
- Basic factory systems
- Refresher on singletons
- Basic iterators
- Basic adapters

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
  - Design Patterns
    - Adapter Pattern
    - Iterator Pattern
    - Singleton Pattern
    - Factory Pattern

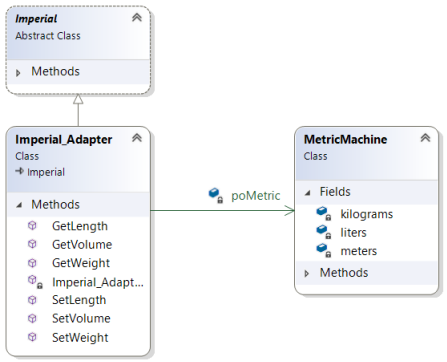
Assignments

General:

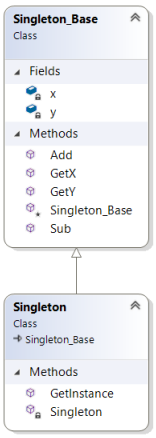
- Look at notes / lecture for Design Patterns
- Additional useful links
  - https://www.oodesign.com/
  - https://www.dofactory.com/net/design-patterns
  - https://sourcemaking.com/design_patterns
  - https://en.wikipedia.org/wiki/Design_Patterns
  - https://en.wikipedia.org/wiki/Software_design_pattern
  - https://refactoring.guru/design-patterns
- Books
  - Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software
  - Design Patterns: Elements of Reusable Object-Oriented Software

Problems:

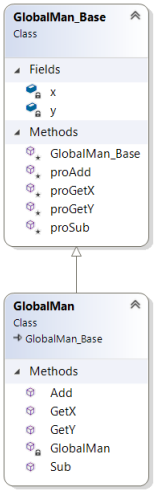
- Adapter Pattern
  - You are given a metric machine(all the units are in metric)
  - Create an Adapter class to interface an Imperial system (feet, gallons, pounds)



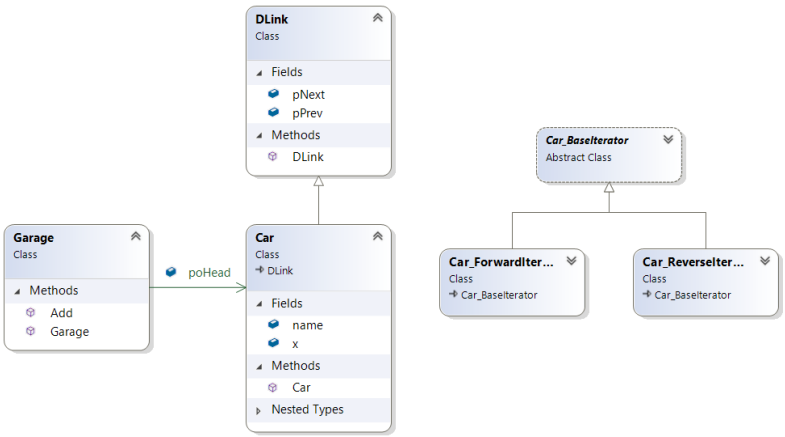
- Singleton Pattern
  - Classic textbook singleton that uses *GetInstance()* method



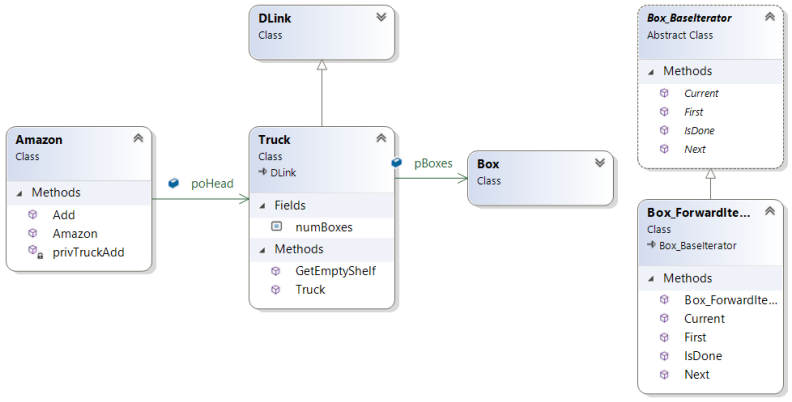
  - Modified Singleton (instance is private) only uses static methods



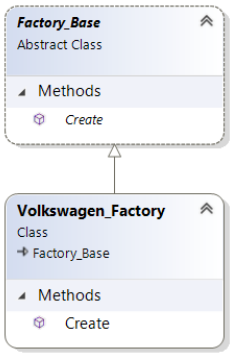
- Iterator Pattern
  - Garage - Given a linked list of Cars..
    - Create a forward iterator (start from the first car and iterate to the last)
    - Create a reverse iterator (start from the last car and iterate to the first)

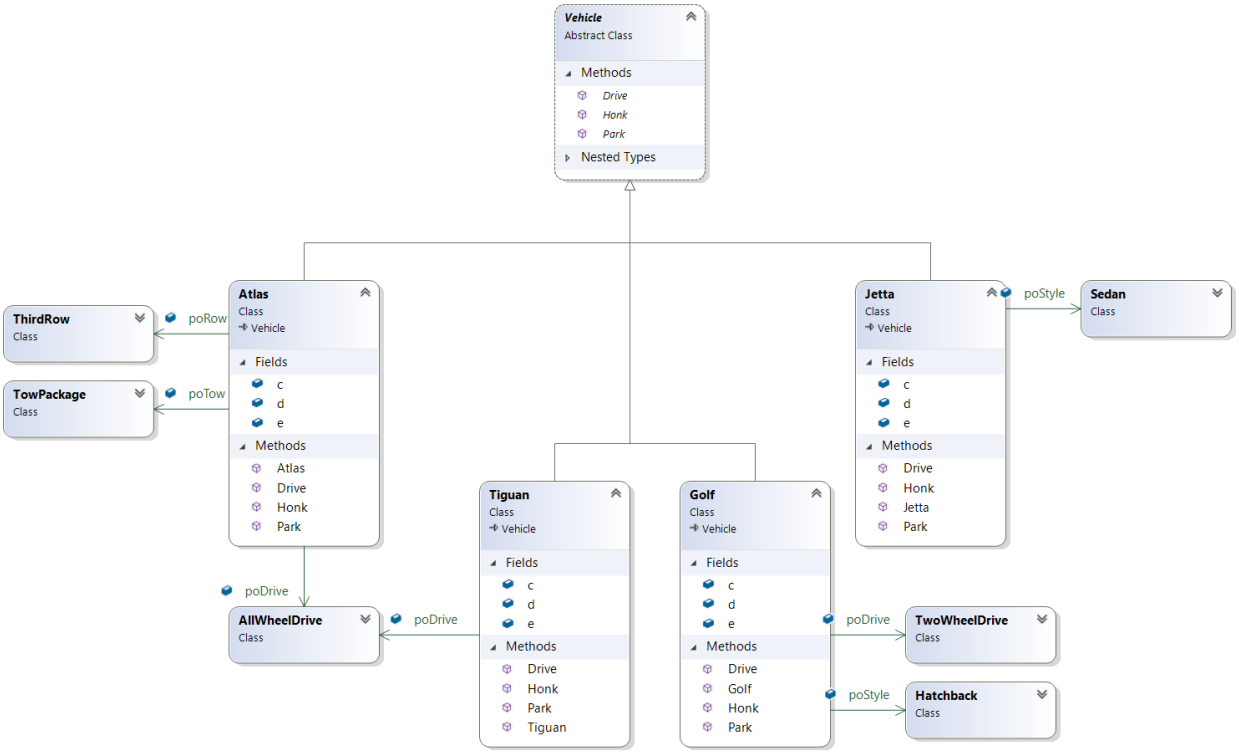


- Amazon - More complicated collection
  - Amazon has delivery trucks
    - Each truck has 1-4 boxes
      - There is a linked list of Trucks
      - Each Truck has 0-4 boxes
  - Iterate through the fleet…
    - Go to each Truck sequentially through linked list
    - Iterate through each box
    - Then advance to the next truck
  - Sometimes the Box is empty (Skip those)
  - Only return NON-EMPTY boxes in the iterator



- Factory Pattern
  - Create a vehicle factory (Volkswagen)
    - The factory builds and returns a new vehicle base on the input parameters
    - Some features are switchable (input parameters to the Create())
      - Color and Model
    - Some features are bound to the specific car model
      - Engine and Doors
    - Each vehicle has unique data
      - Golf – 2 Door, Petrol
      - Jetta – 4 Door, Diesel
      - Tiguan – 4 Door, Electric
      - Atlas – 4 Door, Petrol





General guidelines:

- Idea is to get you comfortable with these patterns
  - You will include these concepts into the Space Invaders project
- Create UML diagrams to help
  - Post on Piazza questions and clarifications
- No need to add any files… the unit tests are fully stubbed out

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;`

Development

- Store project in student directory in perforce
- Do your work in the supplied PA2 project

Submission

- Submit your PA2 directory into perforce:
  - /student/<yourname>/PA2/…
    - 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 one design pattern at a time
    - Look up the pattern
    - See some reference code
        - I like *oodesign* and *dofactory* reference
- You code might be very small…
    - You might think "that's it".
        - Understand what the pattern is doing… why its doing x behavior
    - I created semi-real examples… so there is a lot of code to give the environment
        - But in some cases, you just fill in one or two methods

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.
    - Its not hard… just different way of solving problems
        - Embrace the pattern concept