**Terms to review:**

identifier

variable

constant

declaration

initialization

assignment

argument

data type

**function**

**method**

**invoke / call**

**calling method**

**called method**

**method declaration**

**method body**

**return type**

**parameter**

**parameter list**

**argument**

**local variable**

**nested method call**

**return statement**

overloaded method

value type

reference type

**class**

**object/instance**

**instantiate**

**constructor**

**field (aka. member variable or instance variable)**

**property**

**access modifier**

**public**

**private**

**encapsulation**

**information hiding**

**implementation hiding**

**composition**

**aggregation**

**Homework & Labs**

*// Please name your projects LB1, LB2, LB3, etc*

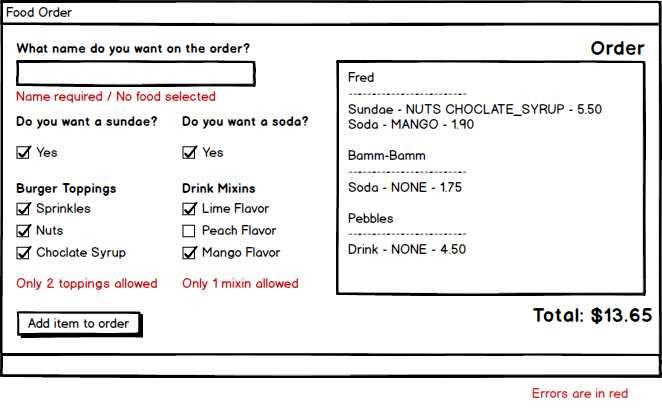
LBI. Complete Naming Conventions Handout

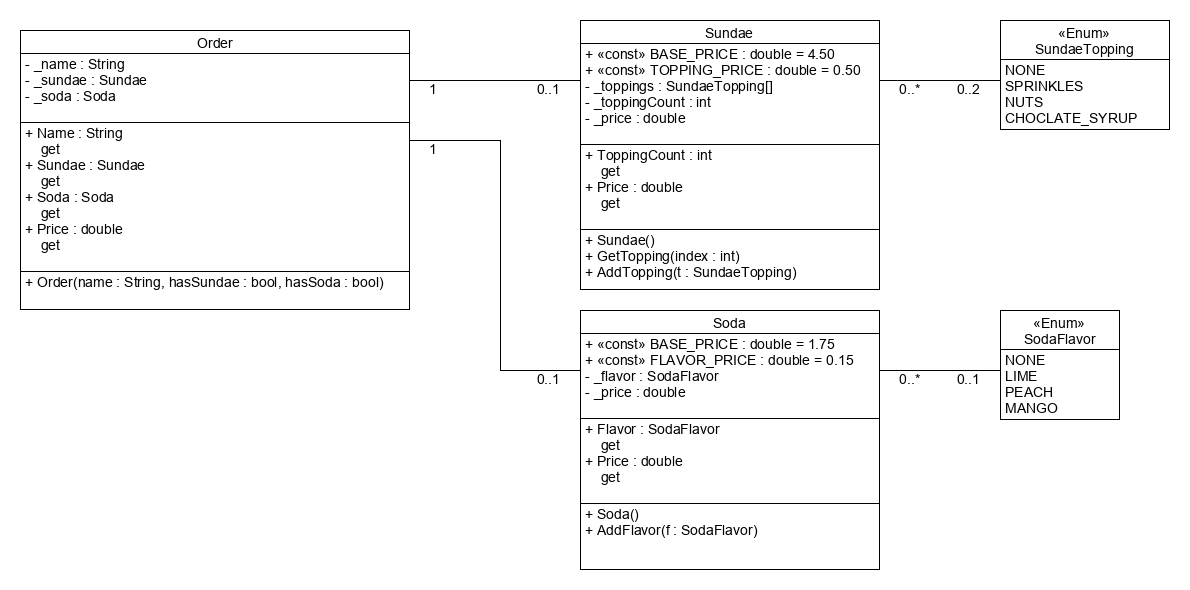
LBII. Complete Data Types Handout

**LB1 FoodOrder (Group)**

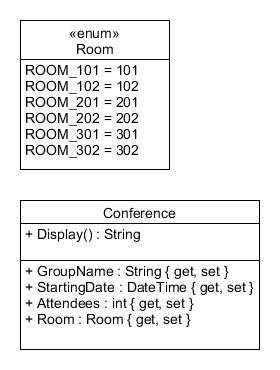
Create a GUI application for a fast food store.

* The user can choose what they want in their order
* Program will calculate the total cost of their order
* Program must use enums
* Program must use and demonstrate composition





**LB2 Conferences**

Write a GUI application for a hotel which is hosting several business conferences. The application will allow the user to enter a number of conferences and then search them by date range.

Create a **Conference** class according to the UML diagram below.

1. All of the properties should be auto-implemented.
2. Has a method named **Display()** that returns a string in the format **"({attendees}) {name} on {date} in {ROOM}"**

The user can enter up to 20 conferences.

Each time the user presses the add button:

1. A new **Conference** object should be created
2. The **Conference** should be added to an array of **Conference** objects
3. Displays the **Conference** in the output
4. Clears out the text fields and sets the focus to the name field

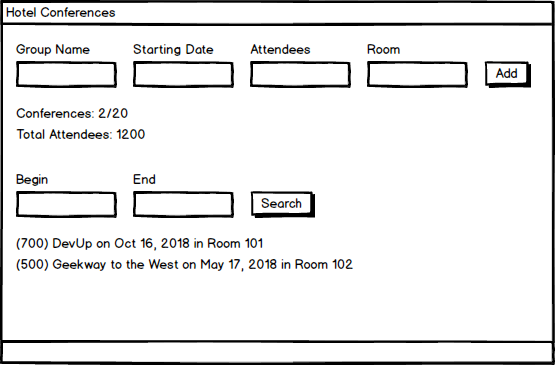
The user can also search for conferences using a date range.

*You can find out more about working with dates here:*

[*https://msdn.microsoft.com/en-us/library/system.datetime(v=vs.110).aspx*](https://msdn.microsoft.com/en-us/library/system.datetime(v=vs.110).aspx)

<https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings>

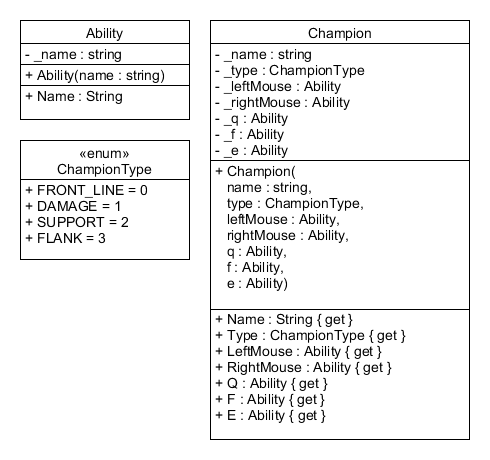
*Please remember to set the tab order.*

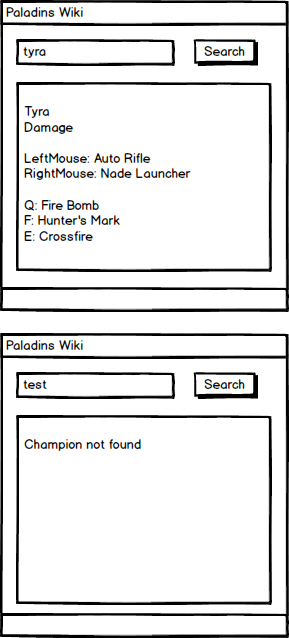


**LB3 Paladins**

Write a GUI application that allows the user to look up information about a champion in Paladins by name. (<https://paladins.gamepedia.com/Champions>)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Name & Type*** | ***LeftMouse*** | ***RightMouse*** | ***Q*** | ***F*** | ***E (Ultimate)*** |
| **Ruckus**  Front Line | Miniguns | Missile Launcher | Emitter | Advance | Hexa Fire |
| **Makoa**  Front Line | Cannon | Dredge Anchor | Shell Shield | Shell Spin | Ancient Rage |
| **Drogoz**  Damage | Rocket Launcher | Fire Spit | Salvo | Thrust | Dragon Punch |
| **Tyra**  Damage | Auto Rifle | Nade Launcher | Fire Bomb | Hunter's Mark | Crossfire |
| **Grover**  Support | Throwing Axe | Crippling Throw | Blossom | Vine | Whirlwind |
| **Evie**  Flank | Ice Staff | Ice Block | Blink | Soar | Ice Storm |





**LB4 Build-A-Lab**

Create your own lab.

* Submit your design to the bin in class.
* Submit your implementation to GitHub.
* Program must have tab order configured.
* Program must follow naming conventions for all controls, variables, constants, methods, classes, and enumerations.
* Program must have at least two classes.
* Program must demonstrate either composition or aggregation.