Chapter 14
Composition (Aggregation)

Reading pp. 860-864

Spring 2015

Object Requests

- In a class, developers have three ways of dealing with object requests
  1. Deal directly with the request by implementing code in a function.
  2. Delegate the request to another object. This is known as **delegation**, **composition**, or **aggregation** (one object "has-a" object to perform the request)
  3. Let a superclass handle the request through the use of inheritance – to be discussed later

Composition / Aggregation

- We use composition when some class already exists to handle a request made on a class being designed
- The “host” class instantiates an object of the “helper” class and sends messages when appropriate
- Helper objects are typically privately stored
- Can you come up with a real world example of composition?

Example

- A book class could contain several members:
  - Title
  - Author
  - Publisher
  - Price
  - Date Published
- What are the types for each of the members?

Composition Advantages

- Objects can be simple in design
- More complex objects can easily be made from existing objects (code reusability)

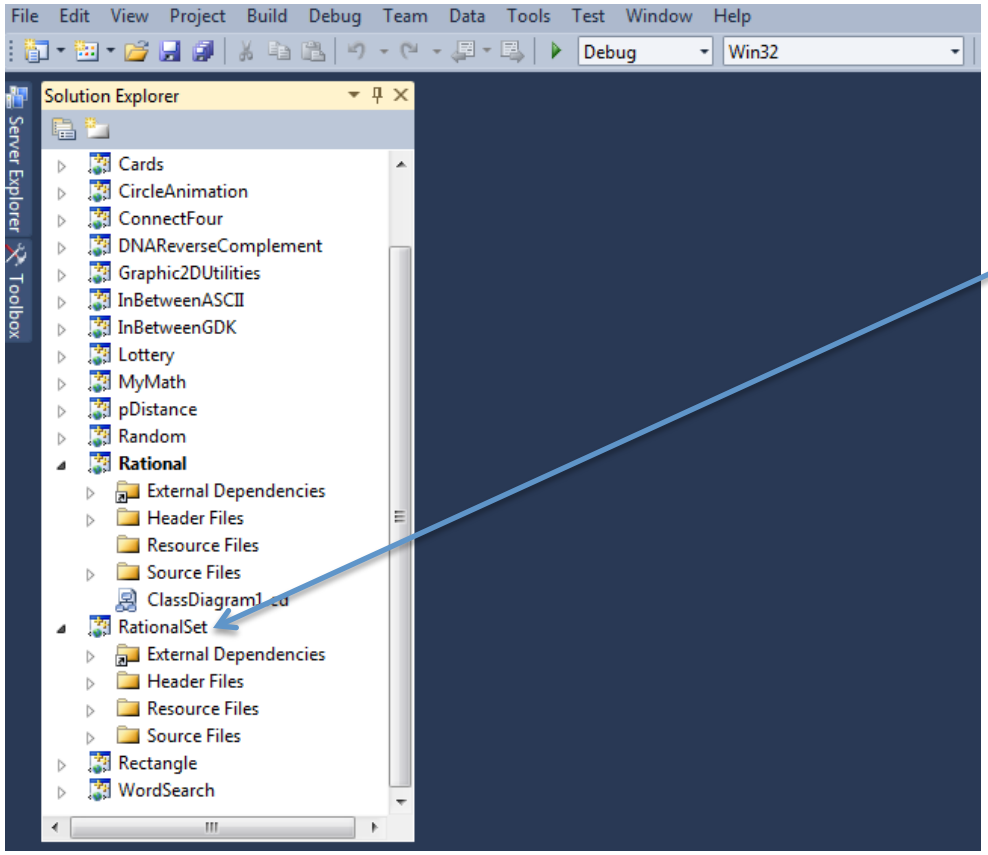
Problem

- We have implemented the ADT Rational.
- In mathematics, we often times want to have a set of something (e.g. the set of real numbers)
- How might we create the ADT RationalSet?

Rational ADT
Step #1

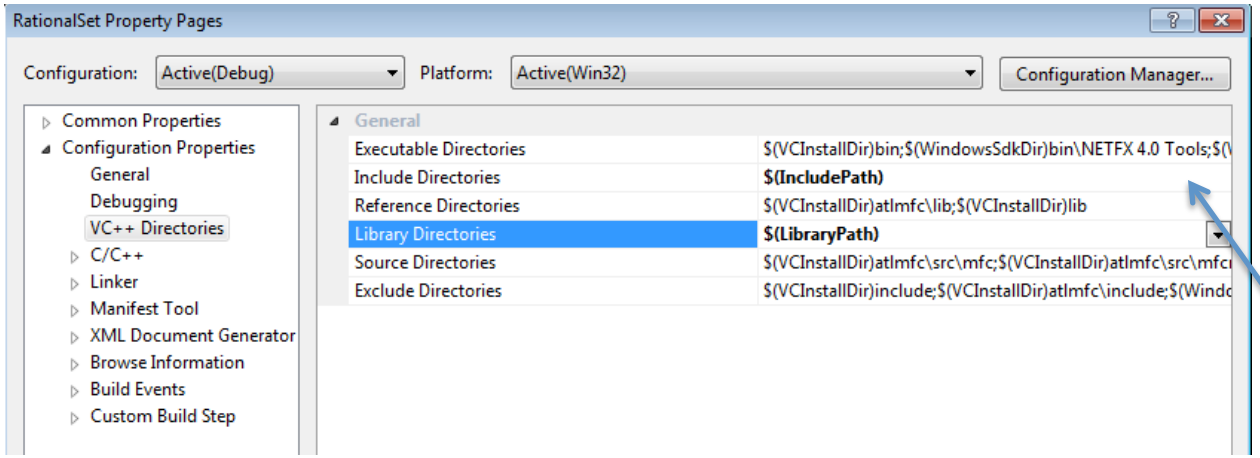
- Copy the solution (Composition) from the public folder on Grace to your desktop
- Notice that it contains a complete Rational project
- Create a project called RationalSet
- Set up the include, library, and dependency information for the project RationalSet
  - We will do this together. Take Notes!!!!

Linking Projects



Right click on the project RationalSet and select **properties**

Linking Projects

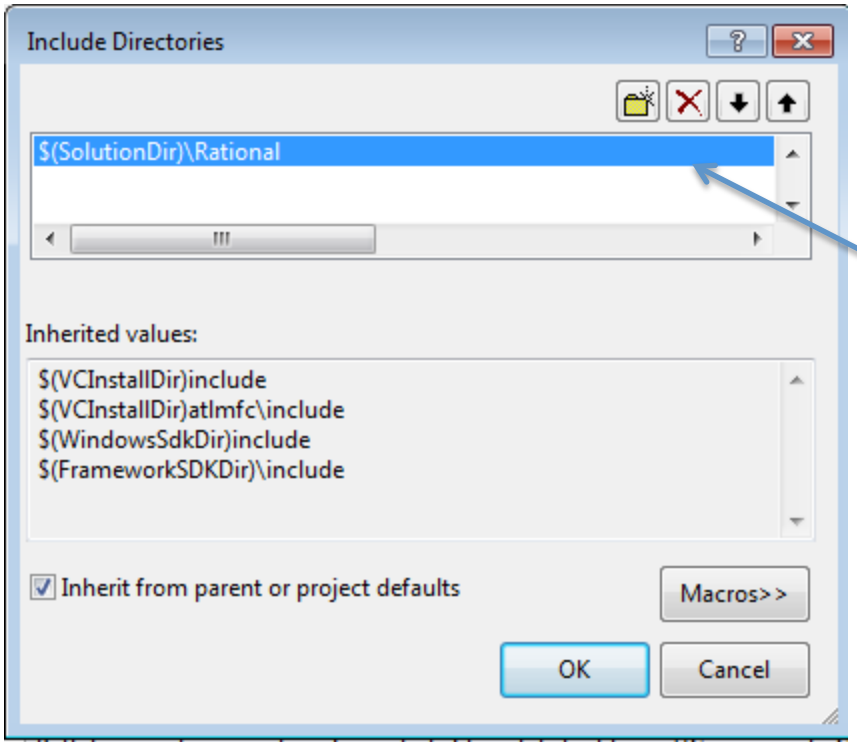


Click on Configuration Properties and then VC++ Directories

We will be adding directory paths to the Include Directories and the Library Directories

Click here and select edit path

Linking Projects

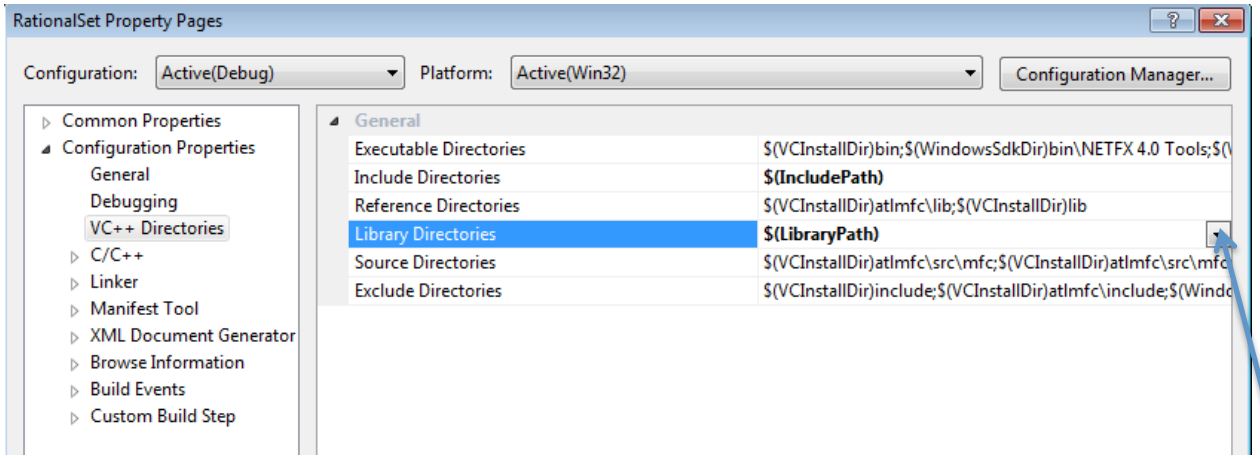


We are linking to Rational.h, which is located in the directory Rational within the solution directory.

Type:

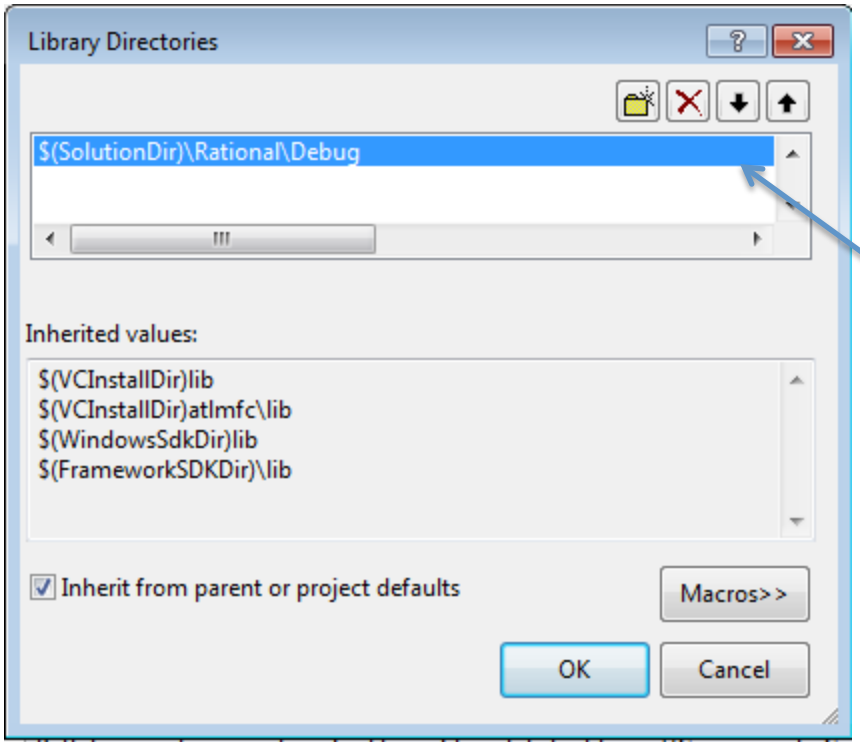
$(SolutionDir)\Rational

Linking Projects



Click here and select edit path

Linking Projects

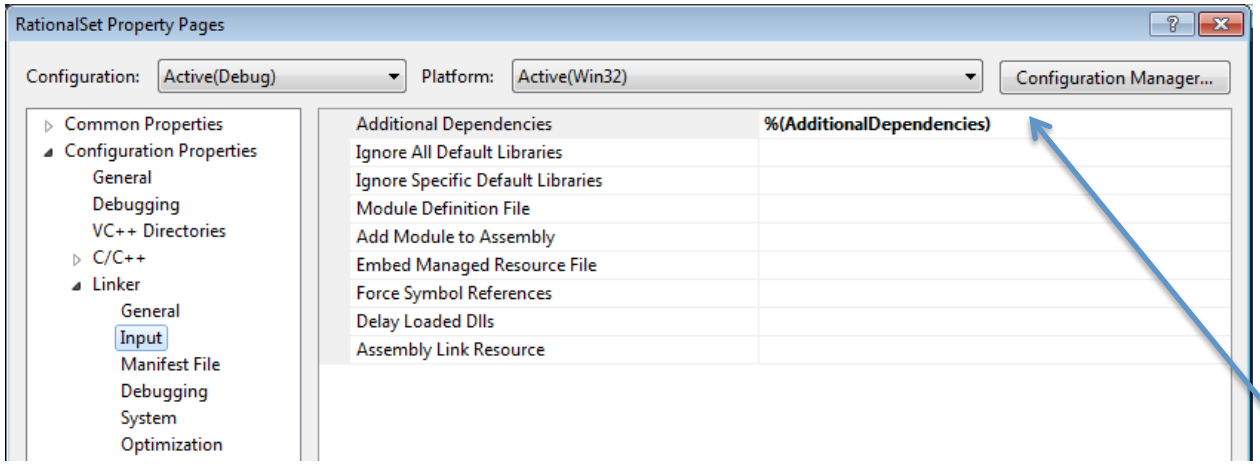


We are linking to Rational.obj, which is located in the directory Debug within the directory Rational within the solution directory.

Type:

$(SolutionDir)\Rational\Debug

Linking Projects

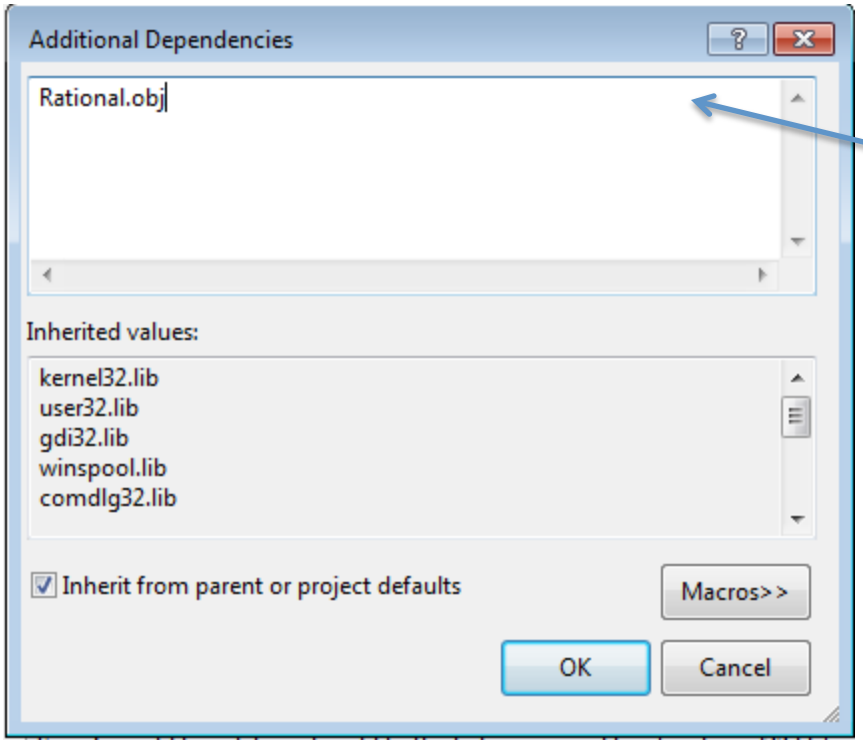


Click on Linker on the left, then go to Input

Here we need to link to the Rational.obj file

Click here

Linking Projects

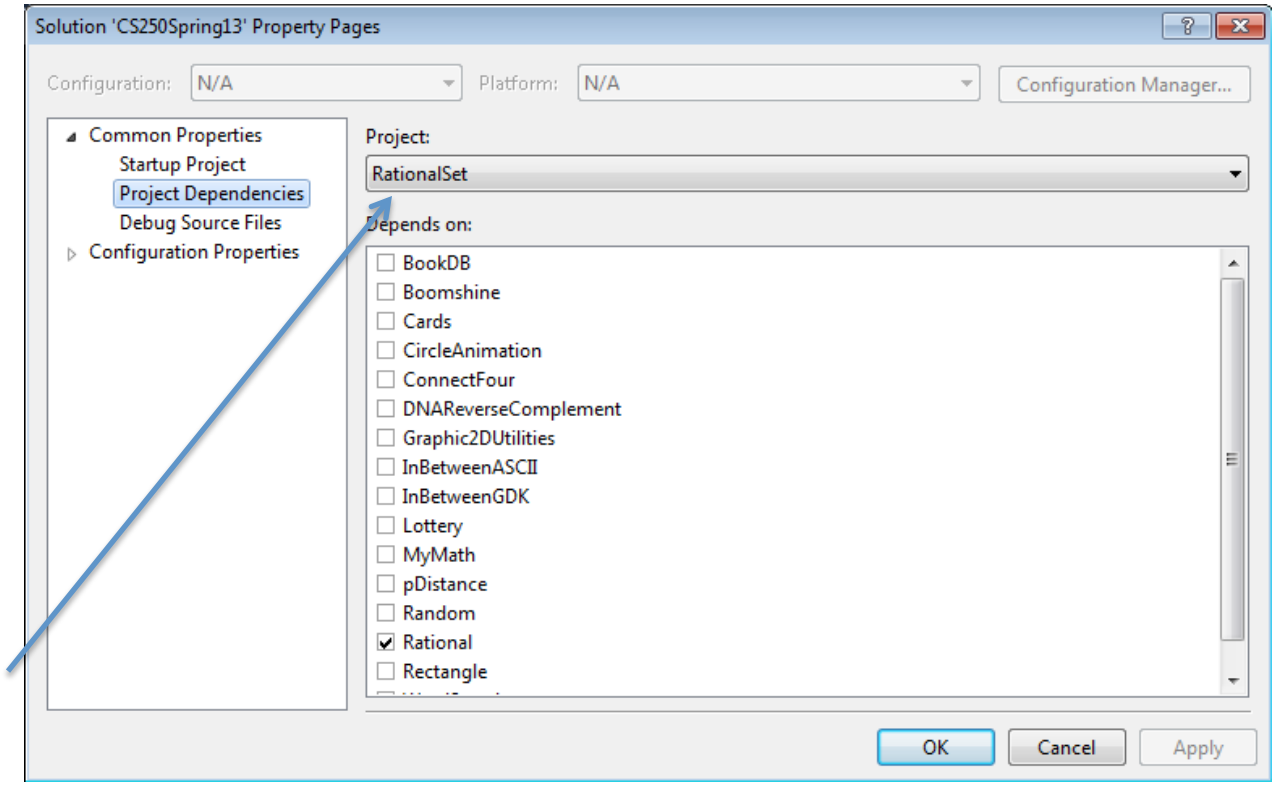


Type:

Rational.obj

Linking Projects

Finally, let's check that the dependencies are set correctly. Right click on the solution name and select properties



Select RationalSet here and make sure that the Rational box is checked

Linking Projects

- For a video on all of this, go to:
  - http://zeus.cs.pacificu.edu/PacificCSVideos/studio2010/resolvingdependencies.html

Rational ADT
Step #2

- Create the interface for RationalSet

Rational ADT
Step #3

- Implement each function one at a time
- Write the driver and test each function
- Note: Do not include any headers that are not necessary in any of the files