CRICOS PROVIDER 00123M

School of Computer Science

COMP SCI 1103/2103 Algorithm Design & Data Structure

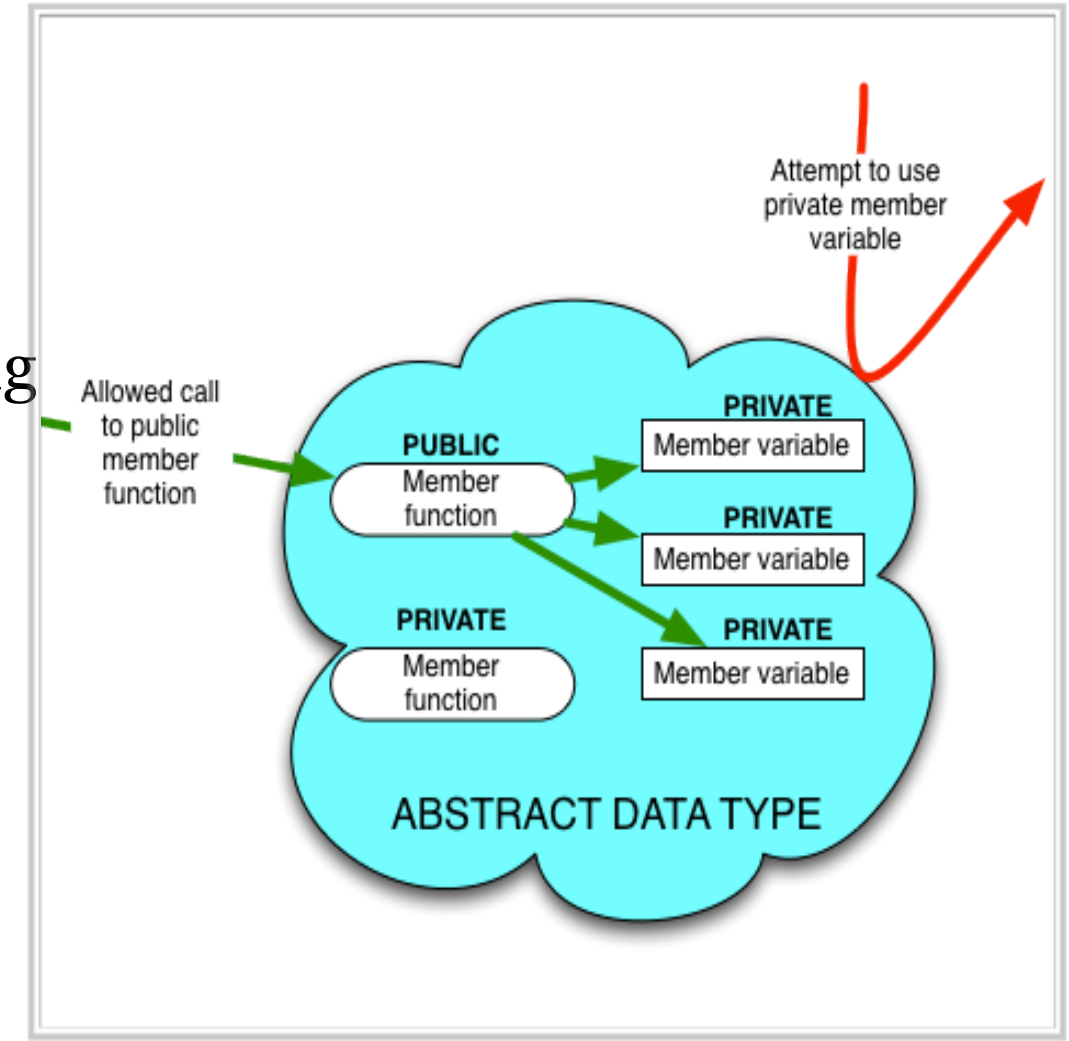
Class Hierarchies & Inheritance

adelaide.edu.au

seek LIGHT

Review

- ADT and Black boxes
- Interface
  - Public member functions
  - Description
- Interface is the only thing a user of ADT needs to know
  - Information hiding
  - Benefits?
- Three rules to make a class an ADT



Overview

- Class
- Objects = data + member functions.
- Design: What to do with similar entities
- In this course we will frequently use classes that are very similar to each other, but not quite the same.
- Can we use elements of C++ to make this more efficient?

Design

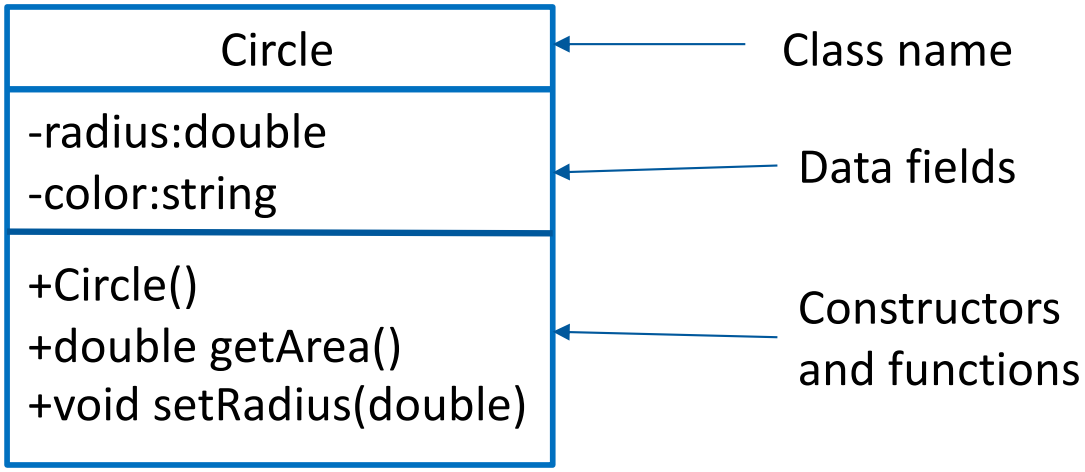
- Design a software such that it:
  - Solves the problem correctly.
  - Does so efficiently.
  - Is easy to maintain
  - Is potentially applicable to other areas.

OO approach

- You are familiar with OOP
- Identify entities that you should define a class for them
- Consider the entity “student”
  - information : name, address, field, scores, etc
  - Some functions/behaviors: e.g. calculate GPA

Classes

- A class is a template or a blueprint that defines what an object's data and functions will be.
- Objects will call methods on each other.
- The methods should be strongly associated with the data of that class.
- Practice and experience -> Good design
  - distinct groupings of variables
  - Split the desired behavior.



Splitting the Behavior

- Make sure that the separation:
  - makes sense
  - efficient

Example

- Consider a class of vehicles. There are many common features that all vehicles have.
  - Variables include:
    - carrying capacity, number of passengers
  - Methods include:
    - add passenger, move vehicle
- However, there are also some features that only belong to a certain type of vehicles.

Why hierarchies?

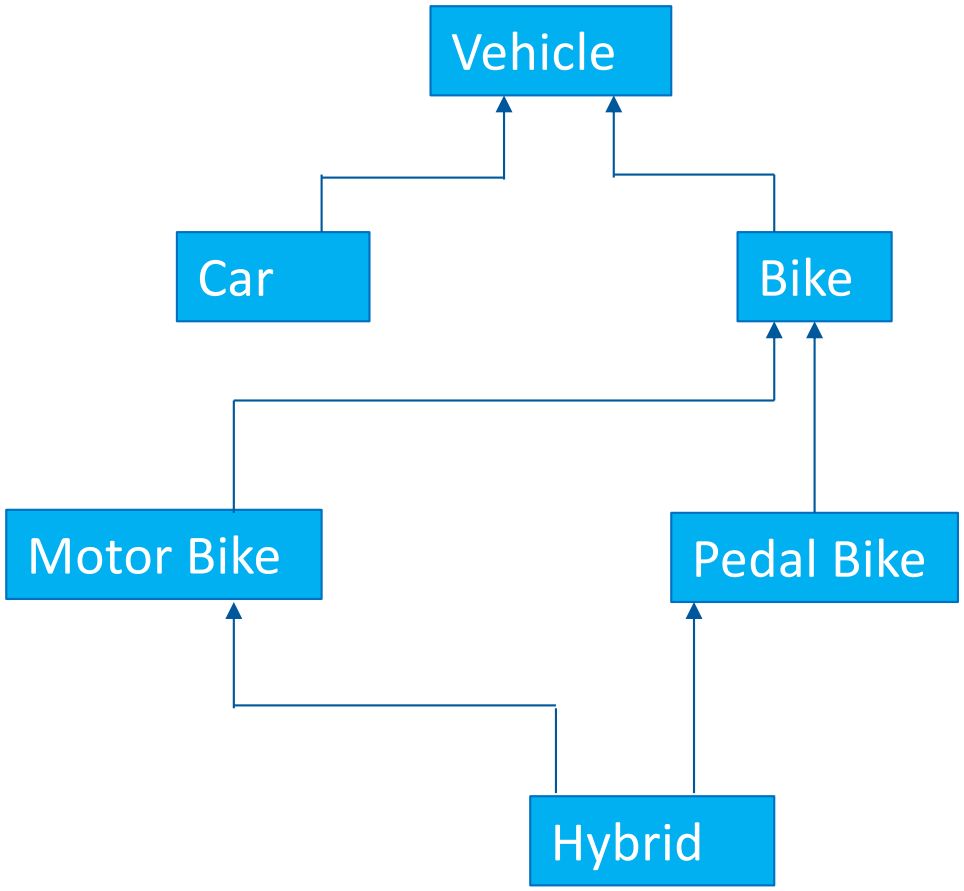
- We could write separate classes: Vehicle class, Car class, Bike class, Aircraft class.
  - Similar properties
  - Vast duplication of code
  - Hard to maintain
- Why not take a different approach and build classes out of OTHER classes?

- The class hierarchy defines the inheritance relationship between the classes.

Example

- Consider the Vehicle class
  - Subclass Cars
  - Subclass Bikes.
- What are the differences between a car and a bike?
- How do we model these in terms of:
  - Variables (data)?
  - Methods (behaviours)?

Example



?

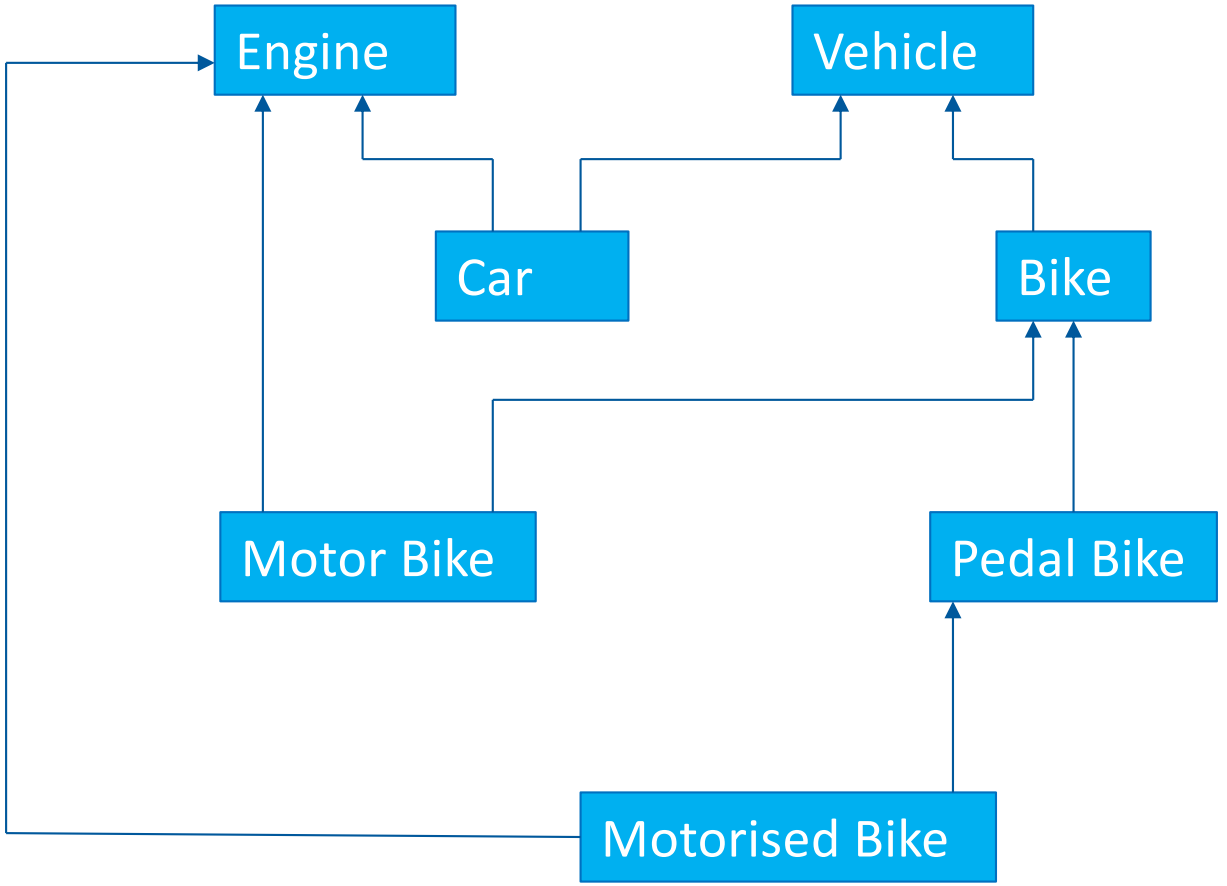
The classes are separated based on concepts.
Should be separated based on behaviours.

Class Hierarchies

- One key problem in constructing a class hierarchy:
  - focus on the relationships between the concepts (common sense) instead of considering the behaviours and how we should implement them.
- Is a Car a separate class near the top or is it just a vehicle with an engine with four wheels?
- Are we adding a behaviour or changing a default?

- We should design the class hierarchy based on our requirements.

Example- separating by behaviors



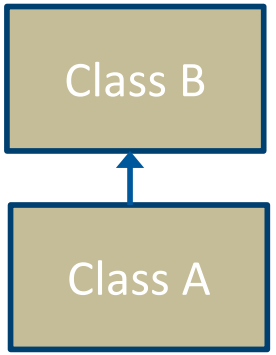
What impact does multiple inheritance have on the methods that we choose?
How do we handle this in C++?

Class Hierarchies

- The class hierarchies that you should form as part of your design must:
  - solve the problem in hand
  - be efficient
  - reduce code duplication
  - be well-defined and clearly understood
- Class hierarchies allow us to build new classes derived from existing classes.
  - Get it right once, then we can re-use it.
  - But how do we re-use it?

Inheritance

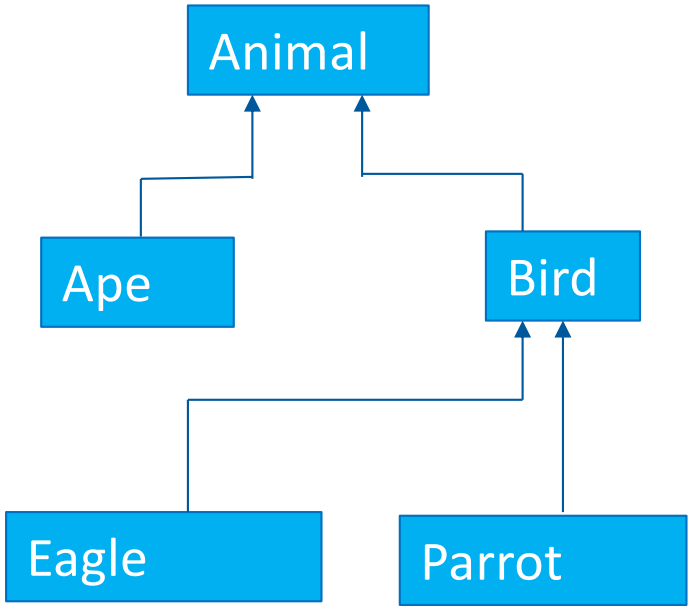
- Inheritance in Object-oriented programming: Derive new classes from existing classes.



- Class ***A*** **extended** from class ***B***
- Class ***A*** : **derived class** or **child class**.
  Class ***B*** : **base class** or **parent class**
- A derived class and its base class must have the ***is-a*** relationship.
- We usually assume that:
  - a derived class has extra features
  - we derive our new class from the most appropriate original class

Example

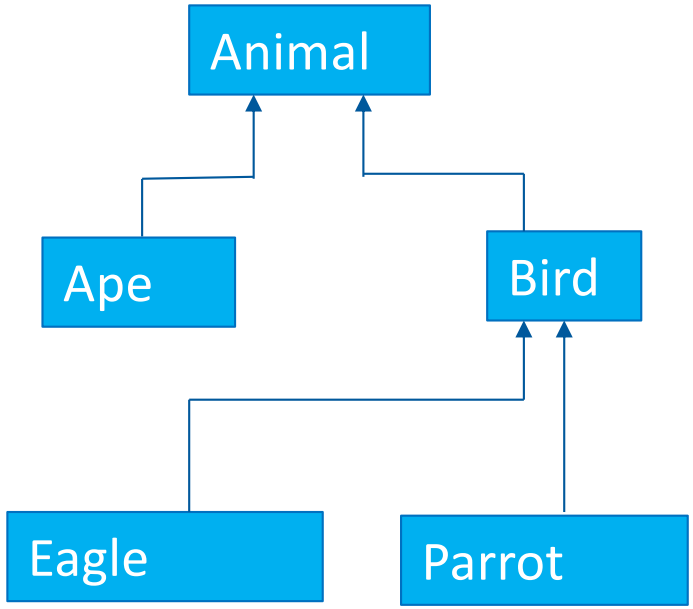
- Why do we have Birds, then Eagles and Parrots?
  - Animals have heart rates, geographical location and food type.
  - Birds have wing spans, feather type and egg colours.
  - Parrots: A rank for their talking ability
  - Birds are Animals.
  - Parrots are birds.



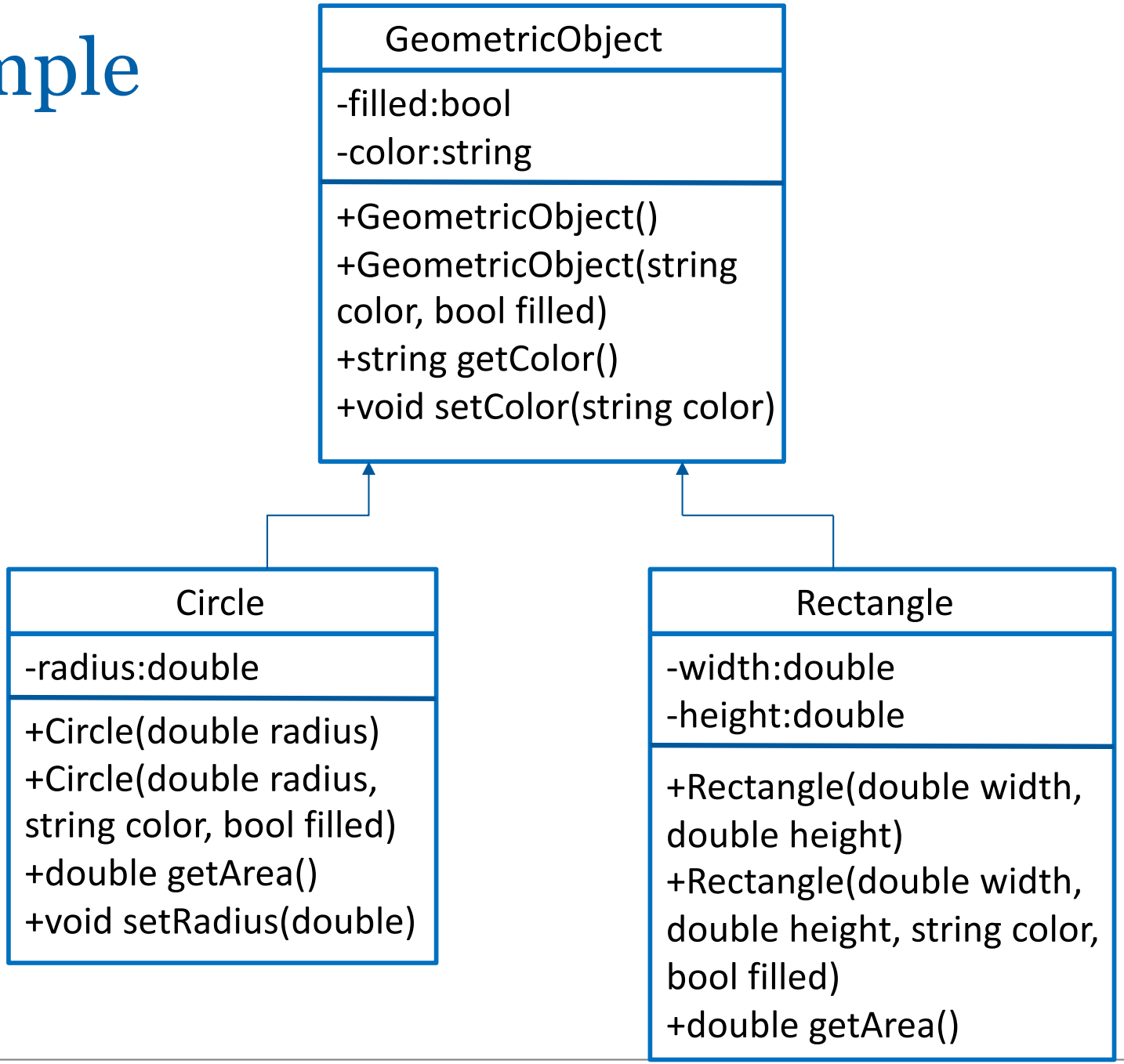
- Every Bird is an Animal but not every Animal is a Bird!

Family relationships

- The derived class Bird is a **subclass/child class** of Animal.
- Animal is a **parent class** of Bird.
- Bird *inherits* the member functions of Animal.



Example


GeometricObject
-filled:bool
-color:string
+GeometricObject()
+GeometricObject(string color, bool filled)
+string getColor()
+void setColor(string color)
Circle
-radius:double
+Circle(double radius)
+Circle(double radius, string color, bool filled)
+double getArea()
+void setRadius(double)
Rectangle
-width:double
-height:double
+Rectangle(double width, double height)
+Rectangle(double width, double height, string color, bool filled)
+double getArea()

Child Class

- The child class inherits from the parent:
  - all public member variables
  - all public member functions
- We can add member functions and variables to the child.
  - Are these available to the parent? NO!
  - Are available to children of the child? Yes
- We can also redefine (avoid)/override (using virtual) the member functions of the parent as viewed by the child.

Advantages

- Update easily
  - If the child classes inherit methods from the parent and then we change the parent, all children automatically get access to the new, improved code.
- Very, very powerful technique.
  - We can use the child in places where we could use the parent, although we are restricted to the methods publicly available from the parent.

Example

```
class Bird : public Animal {
   public:
      Bird();
      Bird(string loc);
}; // End of class definition

Bird::Bird( ): Animal( ) { }
Bird::Bird(string loc):Animal(loc) {}
// Constructor of the child calls the constructor of the parent
```

- The constructors of a base class are not automatically inherited in the derived class.
- One constructor of the base class must be invoked to initialize the variables in the base class
- If you don't name it, the constructor with no argument will be automatically invoked.


THE UNIVERSITY
of ADELAIDE