# CST209

## Object-oriented Programming C++

## (Week 9)

**Prepared by Dr. Teo Bee Guan**

**Presented by Mr. Venantius Kumar Sevamalai**

# Content

- Abstract Class
- Separating Class Specification from Implementation

# Abstract Class

- Sometimes it is helpful to begin a class hierarchy with an abstract base class.
- An abstract base class is not instantiated itself, but serves as a base class for other classes.
- The abstract base class represents the generic, or abstract, form of all the classes that are derived from it.
- An abstract class contains at least one **pure virtual function**.

# Abstract Class

- We cannot create objects of an abstract class.
- However, we can derive classes from them, and use their data members and member functions (except pure virtual functions).

**Practice: Example 1**

# In-class Exercise 1

Create an abstract class and name it as `Employee`. This class should have two member variables, `id` and `name`. Create *getter* and *setter* functions for each of this variable and a pure virtual function `calcSalary()` in the `Employee` class.

Create another class named as `HourWorker` that inherit the `Employee` class. This derived class should have a member variable, `work_hours`. Create a function `calcSalary()` to calculate the salary for `HourWorker`. The hour rate is $25 per hour.

Create another class named as `CommissionWorker` that inherit the `Employee` class. This derived class should have a member variable, `commission`. Create a function `calcSalary()` to calculate the salary for `CommissionWorker`. The salary for a commission worker is $1000 + commission.

# Separate Class Specification from Implemetation

- In the programs we've looked at so far, the class declaration, member function definitions, and application program are all stored in one file.
- A more conventional way of designing C++ programs is to store class declarations and member function definitions in their own separate files.

# Separate Class Specification from Implemetation

- Typically, program components can be stored in the following fashion:
  - Class declarations are stored in their own header files. A header file that contains a class declaration is called a class specification file. The name of the class specification file is usually the same as the name of the class, with a .h extension. For example, the Rectangle class would be declared in the file Rectangle.h
  - Any program that uses the class should #include the class's header file.

**Practice: Example 2, 3, 4**

# In-Class Exercise 2

Create a class named `Employee` with three member variables: `employeeID`, `name`, and `position`. This class should have one `printDetails()` details function to display the details of employee.

Create another class named `Company` with three member variables: `companyName`, `industry` and `employeeList`. The `employeeList` should hold an array of `Employee` objects (Just set the array size to 3 for simplicity). This class should have a `printDetails()` function to display to details of company.

Implement your classes in a main program by displaying the details of a company object along with its employees.

```
Company Info:
Company Name: Haskell Inc
Industry: Computing

Employee Details:
Employee ID: 100
Position: Associate

Employee ID: 101
Position: Senior Associate

Employee ID: 102
Position: Manager
```

**Note: You need to separate the class declaration from implementation**

See you next class