C++

Inheritance: introduction

Introduction

- Inheritance is important to C++: it is what separates abstract data type (ADT) programming from object oriented (OO) programming
- ADT is about the “has-a” relationship. Example: a car has an engine and has tires
- Inheritance is about the “is-a” relationship. Example: a car is a vehicle

Inheritance

- It is the ability to define a new class based on an existing class with a hierarchy
- As a consequence it enables **code reuse**
- The derived class inherits all the members (data and methods) of the base class
  - Note: all constructors, assignment operator and destructor are never inherited
- The derived class can specialize the base class by adding new members (data and methods)
- Because the derived class only has to implement the behavior that differs from the base class, the code from the base class is reused

Inheritance

- It is expressed in C++ by the “ : public “ syntax:
  - class Car : public Vehicle {};
- Car “is a” / “is derived from” / “is a specialized” / “is a subclass of” / “is a derived class of” Vehicle
- Vehicle “is a base class of” / “is a super class of” Car

Example

```
#include <string>
class Person {
 private:
  std::string name;

 public:
  Person(std::string n) : name(n) {}
  void set_name(std::string n);
  std::string get_name();
};
```

```
class Student : public Person {
  private:
    int student_id;
    int year;

  public:
    Student(std::string n, int y, int id) :
      Person(n), year(y), student_id(id) {}
    void set_student_id(int id);
    int get_student_id();
    void set_year();
    int get_year();
};
```

Example - continued

```
#include <string>
class Person {
 private:
  std::string name;

 public:
  Person(std::string n) : name(n) {}
  void set_name(std::string n);
  std::string get_name();
};
```

```
class Teacher : public Person {
  private:
    int employee_id;
    std::string lab;

  public:
    Teacher(std::string n, std::string l, int id) :
      Person(n), lab(l), employee_id(id) {}
    void set_employee_id(int id);
    int get_employee_id();
    void set_lab();
    std::string get_lab();
};
```

Inheritance

- Person is the base class for Student and is the base class for Teacher
- Student is a derived class of Person and Teacher is a derived class of Person
- Student and Teacher inherits all members of Person.
  Example: Student data members are: name (from Person) and student_id and year (extra data in Student)

Is-a relationship

- Derived class objects can always be treated like a base class objects
- Example: an object of type Student can always be used like an object of type Person
  - Especially, we can call all methods of Person on an object of type Student

Example

```
void print_name (const Person& p)
{
  string name = p.get_name();
  cout << name << endl;
}

// …..

Student s(“Yamamoto”, 1, 12345);
Teacher t(“Honda”, “CG”, 789);
print_name(s);
print_name(t);
```

Example 2

- Same thing for pointers: converting a Student* to a Person* is safe and does not need cast

```
void f(Person* p);

void g(Student* s){
  f(s); // no cast, safe
}
```

Inheritance

- Inheritance hierarchy can be extended
- Example: we can have a Foreign_student inheriting from Student, then Foreign_student will also be a Person

```
class Foreign_student : public Student{
 private:
   std::string country;
 // …
};

Foreign_student f(“Jean”, 3, 789, “France”);
print_name(f);
```