Practice 12
Inheritance

2018 second semester
Computer Engineering Programming
Inheritance

• Important concept of Objected Oriented Programming

• General form of class is already defined
  - **Specialized versions, then inherit properties of general class**
  - And add to it/modify it’s functionality for it’s appropriate use

Eg. There’re different kinds of **animals** like cats, dogs, pigs etc.
Inheritance

• **Base class**
  - “General” class from which others derive

• **Derived class**
  - New class
  - Automatically has base class’s:
    - member variables
    - member functions
  - Can then add additional member functions and variables.
Qualifier – Protected

• New classification of class members

• Allows access "by name" in derived class
  – But nowhere else
  – Still no access "by name" in other classes

• In class it’s defined --> acts like private

• Considered "protected" in derived class
  – To allow future derivations

• Many feel this "violates" information hiding
Build “Animal” Class

•Animal class only Contains **general information** about animals.

```cpp
class Animal {
    public:
        Animal () {};
        Animal (string name, string type, string species);
        Animal (string name, string species);
        Animal (int age, string name, string type, string species, int numLegs);

        void PrintInfo ();
        void Bark ();
        void HowToMove ();
        void HowLongLive ();

    protected:
        string animal_name;
        string animal_species;

    private:
        int m_age;
        string m_type;
        int m_numLegs;
};
```

* These functions will be redefined.
Build “Dogs” Class

• Dogs class contains extra information likes IsHairy, IsLoyal etc.

```cpp
class Dogs : public Animal {
    public:
        Dogs (string name, string species, bool hairy, bool loyal);
        Dogs (bool hairy);
        Dogs (bool hairy, bool loyal);
        void Bark () { cout << "Bow-wow" << endl; }
        void HowToMove () { cout << "Walk on four legs" << endl; }
        void PrintInfo () { cout << animal_name << " " << animal_species << " " << m_isHairy << " " << m_isLoyal << endl; }
        bool IsLoyal ();
        bool IsHairy ();
    private:
        bool m_isHairy;
        bool m_isLoyal;
};
```
Build “Cats” Class

• Cats class also contains extra information likes IsHairy, IsSensitive etc. *(But, Cats don’t bark like “Bow-wow”)*.
Examples

```cpp
int main() {
    Dogs d1 = Dogs ("BangUl", "Mammal", true, false);
    Cats c1 = Cats ("NaBi", "Mammal", true, true);

    d1.Bark ();
    c1.Bark ();

    d1.PrintInfo ();
    c1.PrintInfo ();
}
```

```
hsherlco@uni:~/18_fall_cpp$ ./ani
Bow-wow
Meow
BangUl Mammal 1 0
NaBi Mammal 1 1
```