**Tutorial 6**

class Pet

{

public:

Pet () {} // Constructors, Destructor

~Pet() {}

void speak()

{cout << "Growl" << endl;}

};

class Rat: public Pet

{

public:

Rat() {}

~Rat() {}

void speak()

{cout << "Rat noise" << endl;}

};

class Cat: public Pet

{

public:

Cat() {}

~Cat() {}

void speak()

{cout << "Meow" << endl;}

};

void chorus(Pet pet, Pet \*petPtr, Pet &petRef)

{

pet.speak();

petPtr->speak();

petRef.speak();

}

int main()

{

Pet \*ptr; //Pointer to base class

ptr = new Pet;

cout << "Pet Created" << endl;

cout << "Pets singing" << endl;

chorus(\*ptr,ptr,\*ptr);

cout << endl << endl;

delete ptr;

ptr = new Rat;

cout << "Rat Created" << endl;

cout << "Rats singing" << endl;

chorus(\*ptr,ptr,\*ptr);

cout << endl << endl;

delete ptr;

ptr = new Cat;

cout << "Cat Created" << endl;

cout << "Cats singing" << endl;

chorus(\*ptr,ptr,\*ptr);

cout << endl << endl;

delete ptr;

return 0;

}

1. Trace the program.
2. Convert Speak() function to virtual function then trace the program again.
3. Change class pet to be an abstract pet.