1 Base class with functions set as virtual

```
class Base{
public:
    Base(){
        cout << "Base()\n";
    }
    virtual void Print() {
        cout << "Base\n";
    }
    virtual ~Base(){
        cout << "~Base\n";
    }
};

class Derrived : public Base{
public:
    Derrived(){
        cout << "Derrived()\n";
    }
    void Print(){
        cout << "Derrived\n";
    }

    ~Derrived(){
        cout << "~Derrived\n";
    }
};
```

1.1 Creating reference to Derrived

```
int main(){
    Derrived d;
    d.Print();
}
```

```
out:
Base()
Derrived()
Derrived
~Derrived
~Base
```

1.2 Creating pointer to Derrived

```
int main(){
    Derrived* d = new Derrived;
    d->Print();
    delete d;
}
```

```
out:
Base()
Derrived()
Derrived
~Derrived
~Base
```

1.3 Creating pointer to Base

```
int main(){
    Base* d = new Derrived;
    d->Print();
    delete d;
}
```

```
out:
Base()
Derrived()
Derrived
~Derrived
~Base
```

2 Base class with functions set as non-virtual

```
class Base{
public:
    Base(){
        cout << "Base()\n";
    }
    void Print() {
        cout << "Base\n";
    }
    ~Base(){
        cout << "~Base\n";
    }
};

class Derrived : public Base{
public:
    Derrived(){
        cout << "Derrived()\n";
    }
    void Print(){
        cout << "Derrived\n";
    }

    ~Derrived(){
        cout << "~Derrived\n";
    }
};
```

2.1 Creating reference to Derrived

```
int main(){
    Derrived d;
    d.Print();
}
```

```
out:
Base()
Derrived()
Derrived
~Derrived
~Base
```

2.2 Creating pointer to Derrived

```
int main(){
    Derrived* d = new Derrived;
    d->Print();
    delete d;
}
out:
Base()
Derrived()
Derrived
~Derrived
~Base
```

2.3 Creating pointer to Base

```
int main(){
    Base* d = new Derrived;
    d->Print();
    delete d;
}
out:
Base()
Derrived()
Base
~Base
```