# Department of Electrical Engineering

**CS212**

**Object Oriented Programming**



# Lab 8: Inheritance and Function Overriding

**Class**: BEE - 12C

**Date**: November 8th, 2021

**Time**: Monday (1400 – 1700)

**Name**: Muhammad Umer

**CMS ID**: 345834

**Tools**

* Microsoft Visual Studio 2013

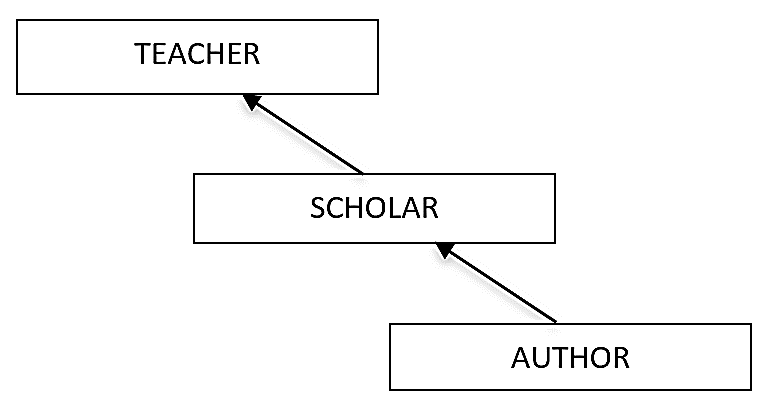
**Inheritance**

Inheritance is a process of creating new classes (derived class) from existing (base) class. We do it for one of the following reasons:

* Reusability
* Code/Software Reliability
* Improved Conceptualization and overall design of program

**Lab Tasks**

**Task 1: Implement the following Class heirarchy**



1. Add appropriate data member in all the classes.
2. Add name and age in both Teacher and Author classes.
3. Add default and parameterized constructors and destructor in all the classes
4. Write appropriate overridden getter, setter functions.
5. In main program create object of Author class and display all values inherited members from base classes and its own member.

**Code**

#include <iostream>

#include <string>

using namespace std;

class Teacher {

 protected:

  string name\_1;

  int age\_1;

 public:

  Teacher() {

    age\_1 = 45;

    name\_1 = "Kaleo";

  }

  Teacher(string a, int b) {

    age\_1 = b;

    name\_1 = a;

  }

  ~Teacher(){};

};

class Scholar : public Teacher {

 protected:

  int age\_2;

  string name\_2;

 public:

  Scholar() {

    age\_2 = 35;

    name\_2 = "Felix";

  }

  Scholar(string a, int b) {

    age\_2 = b;

    name\_2 = a;

  }

  ~Scholar(){};

};

class Author : public Scholar {

 protected:

  int age\_3;

  string name\_3;

 public:

  Author() {

    age\_3 = 99;

    name\_3 = "Dump";

  }

  Author(string a, int b) {

    age\_3 = b;

    name\_3 = a;

  }

  void setAge() { cin >> age\_3; }

  void setName() { getline(cin, name\_3); }

  void displayInheritedData() {

    cout << "\nName of Teacher Class (Inherited): " << name\_1 << endl;

    cout << "Age of Teacher Class (Inherited): " << age\_1 << endl;

    cout << "\nName of Scholar Class (Inherited): " << name\_2 << endl;

    cout << "Age of Scholar Class (Inherited): " << age\_2 << endl;

    cout << "\nName of Author Class: " << name\_3 << endl;

    cout << "Age of Author Class: " << age\_3 << endl;

  }

  void displayData() {

    cout << "\nName of Author Class: " << name\_3 << endl;

    cout << "Age of Author Class: " << age\_3 << endl;

  }

  ~Author() { cout << "\nDestructor was called."; };

};

int main() {

  Author m;

  cout << "\t\tInherited Data";

  m.displayInheritedData();

  cout << "\nInput a name to Mutator (Name): ";

  m.setName();

  cout << "Input a number to Mutator (Age): ";

  m.setAge();

  cout << "\n\t\tMutated Data";

  m.displayData();

  cout << "\n\tProgram Executed!\n";

  return 0;

}

**Terminal Output**

PS D:\NUST\Semester 3\Object Oriented Programming\Labs\Lab 8> lab8.exe

Inherited Data

Name of Teacher Class (Inherited): Kaleo

Age of Teacher Class (Inherited): 45

Name of Scholar Class (Inherited): Felix

Age of Scholar Class (Inherited): 35

Name of Author Class: Dump

Age of Author Class: 99

Input a name to Mutator (Name): Kayne

Input a number to Mutator (Age): 34

Mutated Data

Name of Author Class: Kayne

Age of Author Class: 34

        Program Executed!

Destructor was called.

**Task 2: Modification & Extension**

1. Add a common base class Person
2. Remove name and age data members from Teacher and Author classes and add these data members in common base class Person
3. Also add constructor and destructor in Person class
4. Create object of Author class in main program. Display all data and see the behavior and order for constructors and destructors. Explain why?

**Code**

#include <iostream>

#include <string>

using namespace std;

class Person {

 public:

  int age;

  string name;

  Person() {

    age = 45;

    name = "Kaleo";

  }

  Person(string a, int b) {

    age = b;

    name = a;

  }

  ~Person() { cout << "\nDestructor was called."; }

};

class Teacher : public Person {

 public:

  Teacher() {

    age = 45;

    name = "Kaleo";

  }

  Teacher(string a, int b) {

    age = b;

    name = a;

  }

  ~Teacher() { cout << "\nDestructor was called."; };

};

class Scholar : public Teacher {

 public:

  Scholar() {

    age = 35;

    name = "Felix";

  }

  Scholar(string a, int b) {

    age = b;

    name = a;

  }

  ~Scholar() { cout << "\nDestructor was called."; };

};

class Author : public Scholar {

 protected:

  int age;

  string name;

 public:

  Author() {

    age = 99;

    name = "Dump";

  }

  Author(string a, int b) {

    age = b;

    name = a;

  }

  void setAge() { cin >> age; }

  void setName() { getline(cin, name); }

  void displayInheritedData() {

    cout << "\nName of Teacher Class (Inherited): " << name << endl;

    cout << "Age of Teacher Class (Inherited): " << age << endl;

    cout << "\nName of Scholar Class (Inherited): " << name << endl;

    cout << "Age of Scholar Class (Inherited): " << age << endl;

    cout << "\nName of Author Class: " << name << endl;

    cout << "Age of Author Class: " << age << endl;

  }

  void displayData() {

    cout << "\nName of Author Class: " << name << endl;

    cout << "Age of Author Class: " << age << endl;

  }

  ~Author() { cout << "\nDestructor was called."; };

};

int main() {

  Author m;

  cout << "\t\tInherited Data";

  m.displayInheritedData();

  cout << "\nInput a name to Mutator (Name): ";

  m.setName();

  cout << "Input a number to Mutator (Age): ";

  m.setAge();

  cout << "\n\t\tMutated Data";

  m.displayData();

  cout << "\n\tProgram Executed!\n";

  return 0;

}

**Terminal Output**

PS D:\NUST\Semester 3\Object Oriented Programming\Labs\Lab 8> lab8.exe

Inherited Data

Name of Teacher Class (Inherited): Dump

Age of Teacher Class (Inherited): 99

Name of Scholar Class (Inherited): Dump

Age of Scholar Class (Inherited): 99

Name of Author Class: Dump

Age of Author Class: 99

Input a name to Mutator (Name): Kanye

Input a number to Mutator (Age): 34

Mutated Data

Name of Author Class: Kanye

Age of Author Class: 34

        Program Executed!

Author Destructor was called.

Scholar Destructor was called.

Teacher Destructor was called.

Person Destructor was called.

**Explanation**

**The order of Constructors and Destructors for the case of multiple levels of inheritance works like a stack;** If you consider pushing an item onto the stack as construction, and taking it off as destruction, then you can look at multiple levels of inheritance like a stack. This works for any number of levels. Hence, following this logic, we get the order of Destructors as;

**Author Destructor was called.**

**Scholar Destructor was called.**

**Teacher Destructor was called.**

**Person Destructor was called.**

Where we can infer that the Destructor of the Base class was indeed called last.

**Task 3:**

Develop programs in C++ for following problem:

* **Consider a class Item that stores the:**

1. Title of an item
2. Price
3. Array of three variables to only record the sales in **Rupees** of a particular item for the last three months
4. Has its own function getters and setters

* **Another class HardwareItem stores:**

1. Title of an item
2. Price
3. Array of three variables to record the sales in **Rupees** of a particular item for the last three months
4. Equipment Manufacturer Name
5. Has its own overridden getters and setters

* **Another class SoftwareItem stores:**

1. Title of an item
2. Price
3. Array of three variables to record the sales in **Rupees** of a particular item for the last three months
4. Supported operating system name on which software item runs
5. Has its own overridden getters and setters

***Develop a program****,* usingObject Oriented techniques (inheritance) and appropriate datatypes to construct the necessary classes ensuring maximum code reuse. Don’t add code for constructors.

* The application must work for the following test function:

int main()

{

    Item \*it = new Item;

    HardwareItem h1;

    SoftwareItem \*s = new SoftwareItem;

    it -> getData();

    it -> DisplayData();

    h1.getData();

    h1.DisplayData();

    s -> getData();

    s -> DisplayData();

    delete h1;

}

**Code**

#include <iostream>

#include <string>

using namespace std;

class Item {

  string title;

  double price;

  double record[3];

 public:

  virtual void getData() {

    cout << "\nEnter name of Item: ";

    cin.clear();

    cin.sync();

    getline(cin, title);

    cout << "Enter price of Item: ";

    cin >> price;

    cout << "Enter sales of last three months: ";

    cin >> record[0] >> record[1] >> record[2];

  }

  virtual void DisplayData() {

    cout << "\nItem: " << title << endl;

    cout << "Price: " << price << " Rs" << endl;

    cout << "Record: " << record[0] << " Rs " << record[1] << " Rs "

         << record[2] << " Rs " << endl;

  }

};

class HardwareItem : public Item {

  string title, manufacturer;

  double price;

  double record[3];

 public:

  void getData() {

    cout << "\nEnter name of Item: ";

    cin.clear();

    cin.sync();

    getline(cin, title);

    cout << "Enter price of Item: ";

    cin >> price;

    cout << "Enter manufacturer's name: ";

    cin.clear();

    cin.sync();

    getline(cin, manufacturer);

    cout << "Enter sales of last three months: ";

    cin >> record[0] >> record[1] >> record[2];

  }

  void DisplayData() {

    cout << "\nItem: " << title << endl;

    cout << "Price: " << price << " Rs" << endl;

    cout << "Record: " << record[0] << " Rs " << record[1] << " Rs "

         << record[2] << " Rs " << endl;

  }

};

class SoftwareItem : public Item {

  string title, supported\_os;

  double price;

  double record[3];

 public:

  void getData() {

    cout << "\nEnter name of Item: ";

    cin.clear();

    cin.sync();

    getline(cin, title);

    cout << "Enter price of Item: ";

    cin >> price;

    cout << "Enter supported OS: ";

    cin.clear();

    cin.sync();

    getline(cin, supported\_os);

    cout << "Enter sales of last three months: ";

    cin >> record[0] >> record[1] >> record[2];

  }

  void DisplayData() {

    cout << "\nItem: " << title << endl;

    cout << "Price: " << price << " Rs" << endl;

    cout << "Record: " << record[0] << " Rs " << record[1] << " Rs "

         << record[2] << " Rs " << endl;

  }

};

int main() {

  Item\* it = new Item;

  HardwareItem h1;

  SoftwareItem\* s = new SoftwareItem;

  it->getData();

  it->DisplayData();

  h1.getData();

  h1.DisplayData();

  s->getData();

  s->DisplayData();

  delete it;

  return 0;

}

**Terminal Output**

PS D:\NUST\Semester 3\Object Oriented Programming\Labs\Lab 8> lab8.exe

Enter name of Item: Zephiro

Enter price of Item: 230.50

Enter sales of last three months: 1200 780 2420

Item: Zephiro

Price: 230.5 Rs

Record: 1200 Rs 780 Rs 2420 Rs

Enter name of Item: Leaf Blower

Enter price of Item: 1999.99

Enter manufacturers name: Worx

Enter sales of last three months: 6700.31 8900.00 15023.99

Item: Leaf Blower

Price: 1999.99 Rs

Record: 6700.31 Rs 8900 Rs 15024 Rs

Enter name of Item: Photoshop

Enter price of Item: 9999.99

Enter supported OS: Windows

Enter sales of last three months: 19920 78833 550000

Item: Photoshop

Price: 9999.99 Rs

Record: 19920 Rs 78833 Rs 550000 Rs