EEE102 C++ Programming and Software Engineering II

# Lab Practice 9

**Polymorphism**

Notice:

* + The aim of this lab is for you to become familiar with the usage of the dynamic binding and polymorphism.
  + Practice with the exercises. These parts are not for submission.

|  |  |
| --- | --- |
| ***Exercise 1.1***  State the following statements are TRUE or FALSE | |
| 1 | **this** pointer can be used like any other pointer to access the members of the object it points to; |
| 2 | **this** pointer can be made to point to any object by assigning the address of the object; |
| 3 | A pointer to a base class cannot be made to point to objects of derived class; |
| 4 | Virtual functions allow user to use the same function call to invoke member functions of objects of different classes; |
| 5 | Pure virtual functions force the programmer to redefine the virtual function inside the derived classes. |

|  |  |
| --- | --- |
| ***Exercise 1.2***  Read the piece of program given below, and identify the errors. | |
| **Class Point**  **{**  **int x,y;**  **public:**  **Point (int a=0, int b=0)**  **{x=a;y=b;}**  **void setXY(int a, int b)**  **{x=a;y=b;}**  **void display()**  **{cout<<”(”<<x<<”,”<<y<<”)”;}**  **};**  **class ThreeD : public Point**  **{**  **int z;**  **public:**  **ThreeD (int a=0, int b=0, int c=0)**  **{x=a;y=b;z=c;}**  **void setZ(int c)**  **{z=c};**  **void display()**  **{cout<<”(”<<x<<”,”<<y;**  **cout<<”,”<<z<<”)”;}**  **};** | **int main()**  **{**  **Point \*ptr = new ThreeD();**  **ptr->setXY(1,1);**  **\*ptr.setZ(1);**  **(\*ptr).display;**  **return 0;**  **}** |

|  |  |
| --- | --- |
| ***Exercise 1.3***  Debug and run the following program. What will be the output? | |
| **#include <iostream.h>**  **class A**  **{**  **protected:**  **int a,b;**  **public:**  **A(int x=0, int y)**  **{**  **a=x; b=y;**  **}**  **virtual void print();**  **};**  **class B : public A**  **{**  **private:**  **float p,q;**  **public:**  **B(int m, int n, float u, float v)**  **{**  **p=u; q=v;**  **}**  **B()**  **{**  **p=q=0;**  **}**  **void input(float u, float v);**  **virtual void print(float);**  **};** | **void A::print(void)**  **{**  **cout<<A values: <<a<<" "<<b<<"\n";**  **}**  **void B::print(float)**  **{**  **cout<<B values: <<u<<" "<<v<<"\n";**  **}**  **void B::input(float x, float y)**  **{**  **p=x; q=y;**  **}**  **main()**  **{**  **A a1(10,20), \*ptr;**  **B b1;**  **b1.input(7.5, 3.14);**  **ptr=&a1;**  **ptr->print();**  **ptr=&b1;**  **ptr->print();**  **}** |

|  |  |
| --- | --- |
| ***Exercise 2***  Take the code we've provided for the Employee class (employee.h and employee.cpp) and the Manager class (manager.h and manager.cpp). | |
| 1 | Add a method print() to the Employee class that prints out some employee data for a pay period. E.g.:  Name: John Burke  Pay rate: 25  Pay: 1000  Your print() method should print out the name and pay rate itself, but it should call the printPay() method to print the pay.  **Note:** print() will have to receive the number of hours worked.  Your code should compile and run correctly with the test program polytest.cpp. |
| 2 | After getting the first part to work, consider that we want the pay for a Manager to be printed differently than for an Employee. I.e., print the manager's pay labelled as "Salary:" if they are salaried, and as "Wages:" if they are not.  Do so by overriding (i.e., redefining) the printPay() method in Manager and Employee. (*Hint:* use the virtual mechanism). |
| *Constraint for this exercise:* Only make methods virtual if it is necessary to make the above work. | |