Polymorphism

Last updated 4/15/20

Polymorphism

- Motivation
  - Since we can have inheritance in our code, we may want some functions to operate differently depending on the type passed to them
  - We could redefine the function in each class to customize the operation
    - This does not work when we are using pointers of a base class type to refer to objects of a derived class
    - To make this work we utilize the concepts of polymorphism

Polymorphism

- Details
  - Make the base class function “virtual”
  - This tells the compiler to check for a derived class version of the function when dealing with a pointer of the base type pointing to an object of the derived class

Polymorphism

- Syntax
  - In the base class –
    - Precede the function declaration with “virtual”

      ```
      virtual void print_val();
      ```

  - All derived class versions of the function – including further hierarchical versions will be considered virtual
  - A good practice is to tag all hierarchical versions of the function as virtual

Polymorphism

- Syntax
  - If the virtual function should not be virtual further down the hierarchy – mark it as final

    ```
    virtual void func() final;
    ```

Polymorphism

- Example – no polymorphism

```
/*
 * Triangle.h
 *
 *  Created on: Apr 15, 2019
 *      Author: johnsontimoj
 */

#ifndef TRIANGLE_H_
#define TRIANGLE_H_

#include <iostream>
using namespace std;

class Triangle {
  protected:
    double side1;
    double side2;
    double side3;
  public:
    Triangle();
    Triangle(double s1, double s2, double s3);
    void print_triangle(void);
};

#endif /* TRIANGLE_H_ */
```

```
/*
 * Triangle.cpp
 *
 *  Created on: Apr 15, 2019
 *      Author: johnsontimoj
 */

#include "Triangle.h"

Triangle::Triangle() {
  side1 = 1;
  side2 = 1;
  side3 = 1;
}
Triangle::Triangle(double s1, double s2, double s3){
  side1 = s1;
  side2 = s2;
  side3 = s3;
}
void Triangle::print_triangle(void){
  cout << "triangle: " << side1 << " - "
       << side2 << " - " << side3 << endl;
}
```

Polymorphism

- Example – no polymorphism

```
*
 * RtTri.h
 *
 *  Created on: Apr 15, 2019
 *      Author: johnsontimoj
 */

#ifndef RTTRI_H_
#define RTTRI_H_

#include "Triangle.h"

class RtTri: public Triangle {
  public:
    RtTri();
    RtTri(double s1, double s2, double s3);
    void print_triangle(void);
};

#endif /* RTTRI_H_ */
```

```
/*
 * RtTri.cpp
 *
 *  Created on: Apr 15, 2019
 *      Author: johnsontimoj
 */

#include "RtTri.h"

RtTri::RtTri() : Triangle(){

}
RtTri::RtTri(double s1, double s2, double s3) : Triangle(s1, s2, s3){

}
void RtTri::print_triangle(void){
  cout << "Right triangle: " << side1 << " - "
       << side2 << " - " << side3 << endl;
}
```

Polymorphism

- Example – no polymorphism

```
/*
 * IsoTri.h
 *
 *  Created on: Apr 15, 2019
 *      Author: johnsontimoj
 */

#ifndef ISOTRI_H_
#define ISOTRI_H_

#include "Triangle.h"

class IsoTri: public Triangle {
  public:
    IsoTri();
    IsoTri(double s1, double s2, double s3);
    void print_triangle(void);
};

#endif /* ISOTRI_H_ */
```

```
/*
 * IsoTri.cpp
 *
 *  Created on: Apr 15, 2019
 *      Author: johnsontimoj
 */

#include "IsoTri.h"

IsoTri::IsoTri() : Triangle(){

}
IsoTri::IsoTri(double s1, double s2, double s3) : Triangle(s1, s2, s3){

}
void IsoTri::print_triangle(void){
  cout << "Isosceles triangle: " << side1
       << " - " << side2 << " - " << side3 << endl;
}
```

Polymorphism

- Example – no polymorphism

```
#include <iostream>
using namespace std;

#include "RtTri.h"
#include "IsoTri.h"

int main(void){
// create 3 triangles
Triangle T1(1,2,3);
RtTri T2(4,5,6);
IsoTri T3(7,8,9);
// call each print statement
// to show function override
T1.print_triangle();
T2.print_triangle();
T3.print_triangle();
// create an array of 3 Triangle pointers
Triangle * array[3];
// assign each triangle to the array as
// Triangle type
array[0] = &T1;
array[1] = &T2;
array[2] = &T3;
// call each print statement
// to show function override
// via pointer
array[0]->print_triangle();
array[1]->print_triangle();
array[2]->print_triangle();

}
```

```
triangle: 1 - 2 - 3
Right triangle: 4 - 5 - 6
Isosceles triangle: 7 - 8 - 9
triangle: 1 - 2 - 3
triangle: 4 - 5 - 6
triangle: 7 - 8 - 9
```

the array is of Triangle pointer types

elements point to Triangle, RtTri and IsoTri objects

Note: all three pointers used the Triangle print method

Polymorphism

- Example – polymorphism

```
/*
 * Triangle.h
 *
 *  Created on: Apr 15, 2019
 *      Author: johnsontimoj
 */

#ifndef TRIANGLE_H_
#define TRIANGLE_H_

#include <iostream>
using namespace std;

class Triangle {
  protected:
    double side1;
    double side2;
    double side3;
  public:
    Triangle();
    Triangle(double s1, double s2, double s3);
    virtual void print_triangle(void);
};

#endif /* TRIANGLE_H_ */
```

```
triangle: 1 - 2 - 3
Right triangle: 4 - 5 - 6
Isosceles triangle: 7 - 8 - 9
triangle: 1 - 2 - 3
Right triangle: 4 - 5 - 6
Isosceles triangle: 7 - 8 - 9
```

Note: all three pointers used the pointed-to class print method