OOP/Computer Programming

By: Dr. Danish Shehzad

Date Class

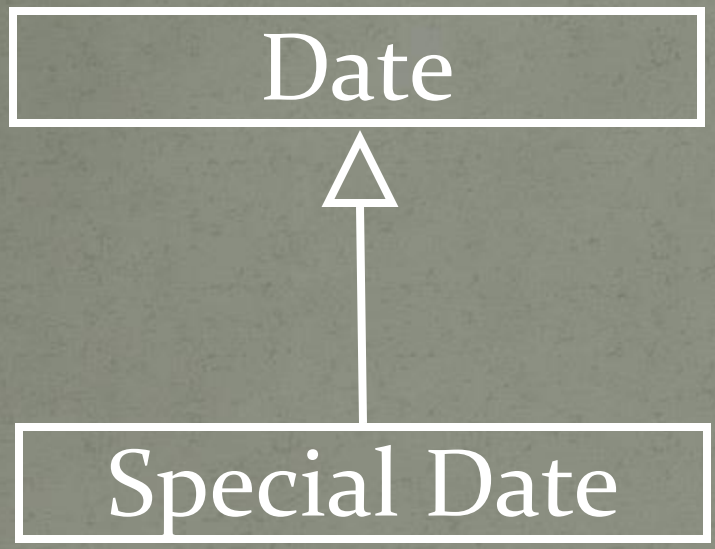
```
class Date{
   int day, month, year;
   static Date defaultDate;
   bool IsLeapYear();

public:
   void SetDay(int aDay);
   int GetDay() const;
   void AddDay(int x);
   ...
    static void SetDefaultDate(
         int aDay,int aMonth, int aYear);
};
```

Date Class

```
int main(){
   Date aDate;
   aDate.IsLeapYear();   //Error
   return 0;
}
```

Creating SpecialDate Class



Creating SpecialDate Class

```
class SpecialDate: public Date{
   …
public:
   void AddSpecialYear(int i){
            …
            if(day == 29 && month == 2
            &&     !IsLeapyear(year+i)){ //ERROR!
                     …
            }
   }
};
```

Modify Access Specifier

- We can modify access specifier “IsLeapYear” from private to public

Modified Date Class

```
class Date{
public:
   …
   bool IsLeapYear();
};
```

Modified AddSpecialYear

```
void SpecialDate :: AddSpecialYear
                                        (int i){
  ...
  if(day == 29 && month == 2
       &&   !IsLeapyear(year+i)){
       ...
  }
}
```

Date Class

```
int main(){
   Date aDate;
   aDate.IsLeapYear();
   return 0;
}
```

Protected members

- Protected members can not be accessed outside the class
- Protected members of base class become protected member of derived class in Public inheritance

Modified Date Class

```
class Date{
   …
protected:
        bool IsLeapYear();
};

int main(){
   Date aDate;
   aDate.IsLeapYear(); //Error
   return 0;
}
```

Modified AddSpecialYear

```
void SpecialDate :: AddSpecialYear
                                        (int i){
  ...
  if(day == 29 && month == 2
       &&   !IsLeapyear(year+i)){
       ...
  }
}
```

Private Inheritance

- If the user does not specifies the type of inheritance then the default type is private inheritance

```
class Child: private Parent {...}
```

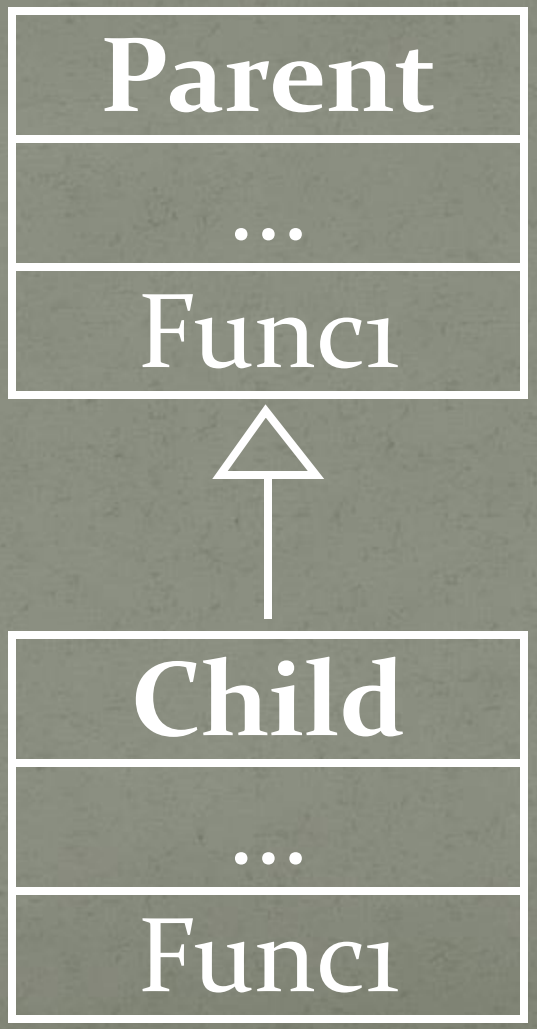
is equivalent to

```
class Child: Parent {...}
```

Overriding Member Functions of Base Class

- Derived class can override the member functions of its base class
- To override a function the derived class simply provides a function with the same signature as that of its base class

Overriding



Overriding

```
class Parent {
public:
   void        Funcı();
   void        Funcı(int);
};

class Child: public Parent {
public:
   void Funcı();
};
```

Overloading vs. Overriding

- Overloading is done within the scope of one class
- Overriding is done in scope of parent and child
- Overriding within the scope of single class is error due to duplicate declaration

Overriding

```
class Parent {
public:
   void Func1();
   void Func1();        //Error
};
```

Access a Method

```
class Point{
    protected:
      int x, y;
    public:
      void set(int a, int b)
          {x=a; y=b;}
      void foo ();
      void print();
};
```

```
class Circle : public Point{
  private:  double r;
  public:
     void set (int a, int b, double c) {
          Point :: set(a, b); //same name function call
          r = c;
     }
     void print();  };
```

```
Point A;
A.set(30,50);  // from base class Point
A.print(); // from base class Point
```

```
Circle C;
C.set(10,10,100);   // from class Circle
C.foo ();  // from base class Point
C.print(); // from class Circle
```

point and circle classes

```
class Point
{
protected:
    int x,y;
public:
    Point(int ,int);
    void display(void);
};
Point::Point(int a,int b)
{
    x=a;
    y=b;
}
void Point::display(void)
{
     cout<<"point = [" <<x<<","<<y<<"]";
}
```

```
class Circle : public Point
{
    double radius;
public:
    Circle(int ,int ,double );
    void display(void);
};
Circle::Circle(int a,int b,double c):Point(a,b) {
    radius = c;
}
void Circle::display(void) {
    Point::display();
    cout<<" and radius = "<<radius;
}
```

```
int main(void)
{
   Circle c(3,4,2.5);
   c.display();
   return 0;
}
```

Output:

point=[3,4] and radius = 2.5

Overriding Member Functions of Base Class

- Derive class can override member function of base class such that the working of function is totally changed

Example

```
class Person{
public:
   void Walk();
};
class ParalyzedPerson: public Person{
public:
   void Walk();
};
```

Overriding Member Functions of Base Class

- Derive class can override member function of base class such that the working of function is similar to former implementation

Example

```
class Person{
  char *name;
public:
  Person(char *=NULL);
  const char *GetName() const;
  void Print(){
     cout << "Name: " << name
          << endl;
  }
};
```

Example

```
class Student : public Person{
  char * major;
public:
  Student(char * aName, char* aMajor);


  void Print(){
     cout<<"Name: "<< GetName()<<endl
         << "Major:" << major<< endl;
  }
...
};
```

Example

```
int main(){
  Student a("Ahmad",    "Computer
  Science");
  a.Print();
  return 0;
}
```

Output

Output:

Name: Ahmed
Major: Computer Science

Overriding Member Functions of Base Class

- Derive class can override member function of base class such that the working of function is based on former implementation

Example

```
class Student : public Person{
  char * major;
public:
  Student(char * aName, char* m);


  void Print(){
     Print();//Print of Person
     cout<<"Major:" << major <<endl;
  }
...
};
```

Example

```
int main(){
  Student a(“Ahmad”, “Computer Science”);
  a.Print();
  return 0;
}
```

Output

- There will be no output as the compiler will call the print of the child class from print of child class recursively
- There is no ending condition

Example

```
class Student : public Person{
  char * major;
public:
  Student(char * aName, char* m);

  void Print(){
     Person::Print();
     cout<<"Major:" << major <<endl;
  }
...
};
```

Example

```
int main(){
  Student a(“Ahmad”, “Computer Science”);
  a.Print();
  return 0;
}
```

Output

Output:

Name: Ahmed
Major: Computer Science

Overriding Member Functions of Base Class

- The pointer must be used with care when working with overridden member functions

Example

```
int main(){
  Student a("Ahmad", "Computer
                      Scuence");
  Student *sPtr = &a;
  sPtr->Print();

  Person *pPtr;
  pPtr->Print();
  return 0;
}
```

Example

Output:

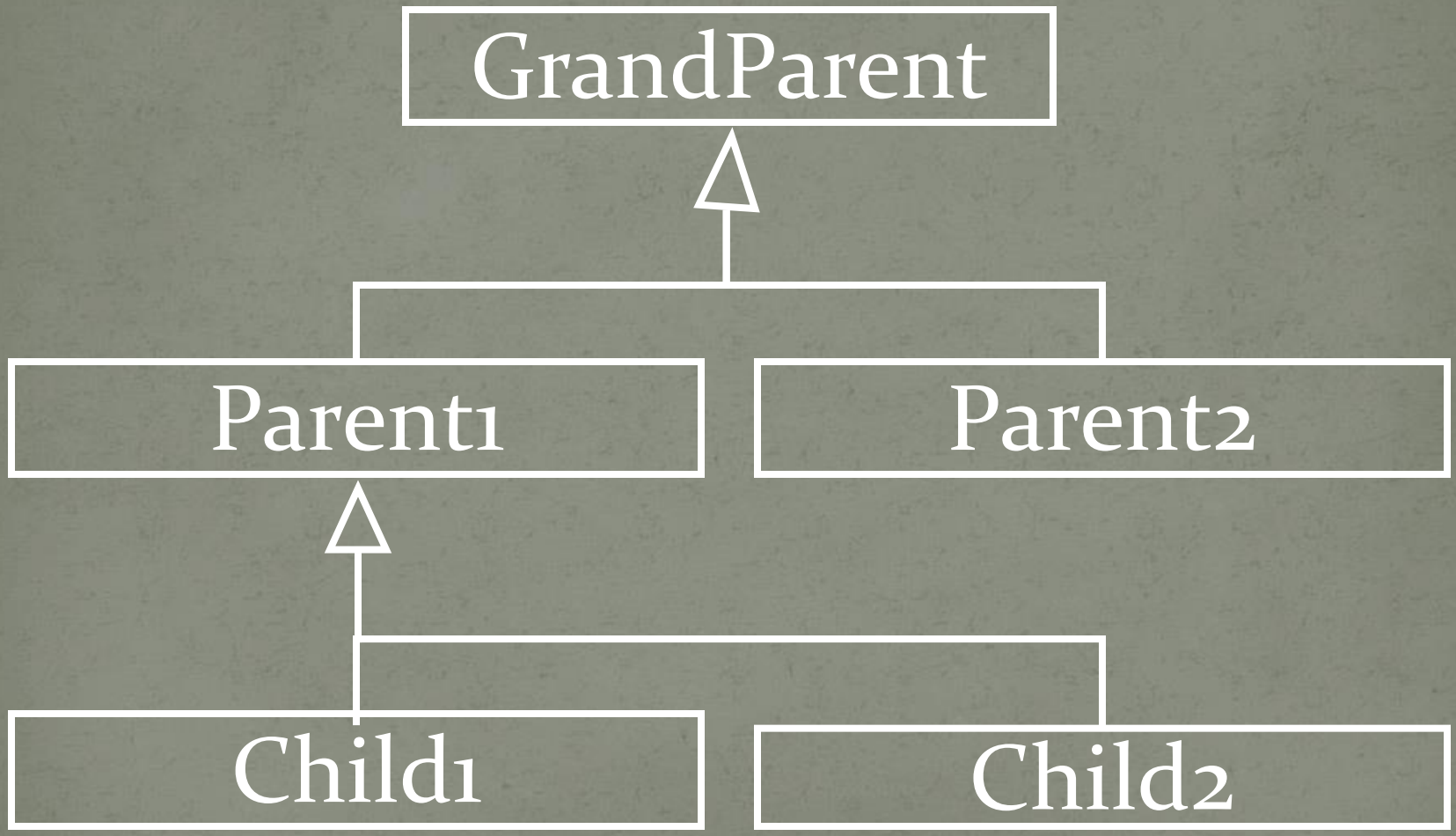
Name: Ahmed
Major: Computer Science

Name: Ahmed

Hierarchy of Inheritance

- We represent the classes involved in inheritance relation in tree like hierarchy

Example



Direct Base Class

- A direct base class is explicitly listed in a derived class's header with a colon (:)

```
class Child1:public Parent1
...
```

Indirect Base Class

- An indirect base class is not explicitly listed in a derived class's header with a colon (:)
- It is inherited from two or more levels up the hierarchy of inheritance

```
class GrandParent{};
class Parent1:
        public GrandParent {};
class Child1:public Parent1{};
```

Base Initialization

- The child can only perform the initialization of direct base class through *base class initialization list*
- The child can not perform the initialization of an indirect base class through *base class initialization list*

Example

```
class GrandParent{
   int gpData;
public:
   GrandParent() : gpData(0){…}
   GrandParent(int i) : gpData(i){…}
   void Print() const;
};
```

Example

```
class Parent1: public GrandParent{
   int pData;
public:
   Parent1() : GrandParent(),
          pData(0) {...}
};
```

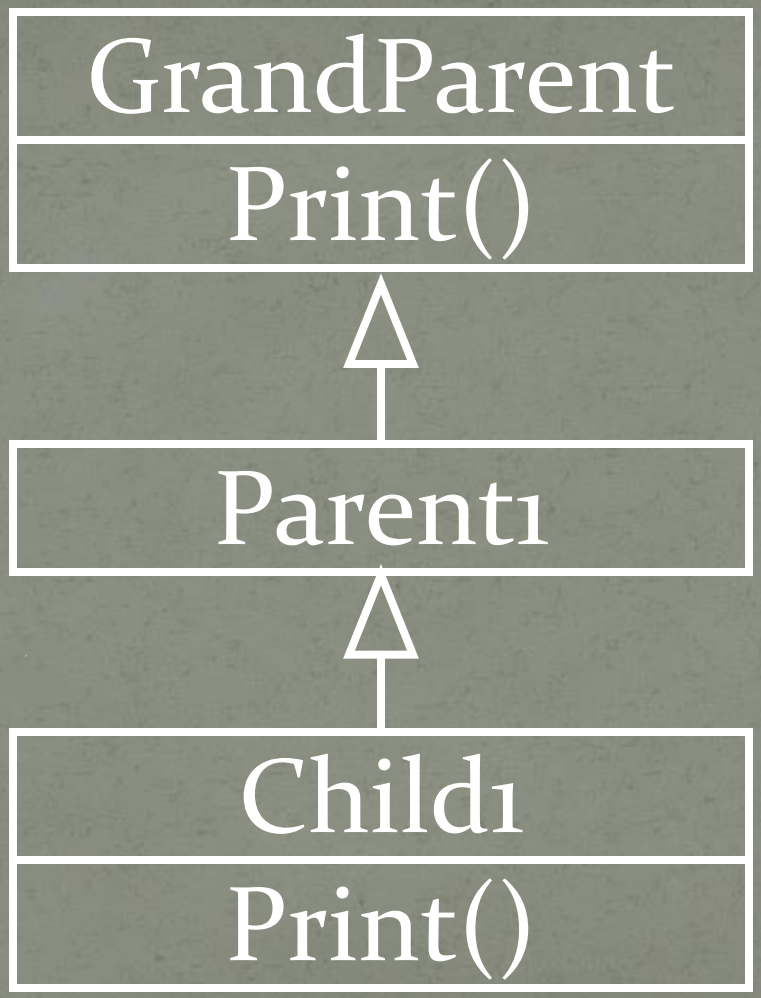
Example

```
class Childı : public Parentı {
public:
   Childı() : Parentı()        {…}
   Childı(int i) : GrandParent (i) //Error
   {…}
   void Print() const;
};
```

Overriding

- Child class can override the function of GrandParent class

Example



Example

```
void GrandParent::Print() {
    cout     << “GrandParent::Print”
             << endl;
}

void Child1::Print() {
    cout << “Child1::Print” << endl;
}
```

Example

```
int main(){
   Child1 obj;
   obj.Print();
   obj.Parent::Print();
   obj.GrandParent::Print();
   return 0;
}
```

Output

- Output is as follows

Child1::Print
GrandParent::Print
GrandParent::Print

Summary

- Inheritence
  - Public
  - Private
  - Protected
- Overriding
- Direct vs Indirect Inheritance

Example Code

- https://www.tutorialspoint.com/compile_cpp_online.php