Inheritance

Chapter: 7, Teach Yourself C++

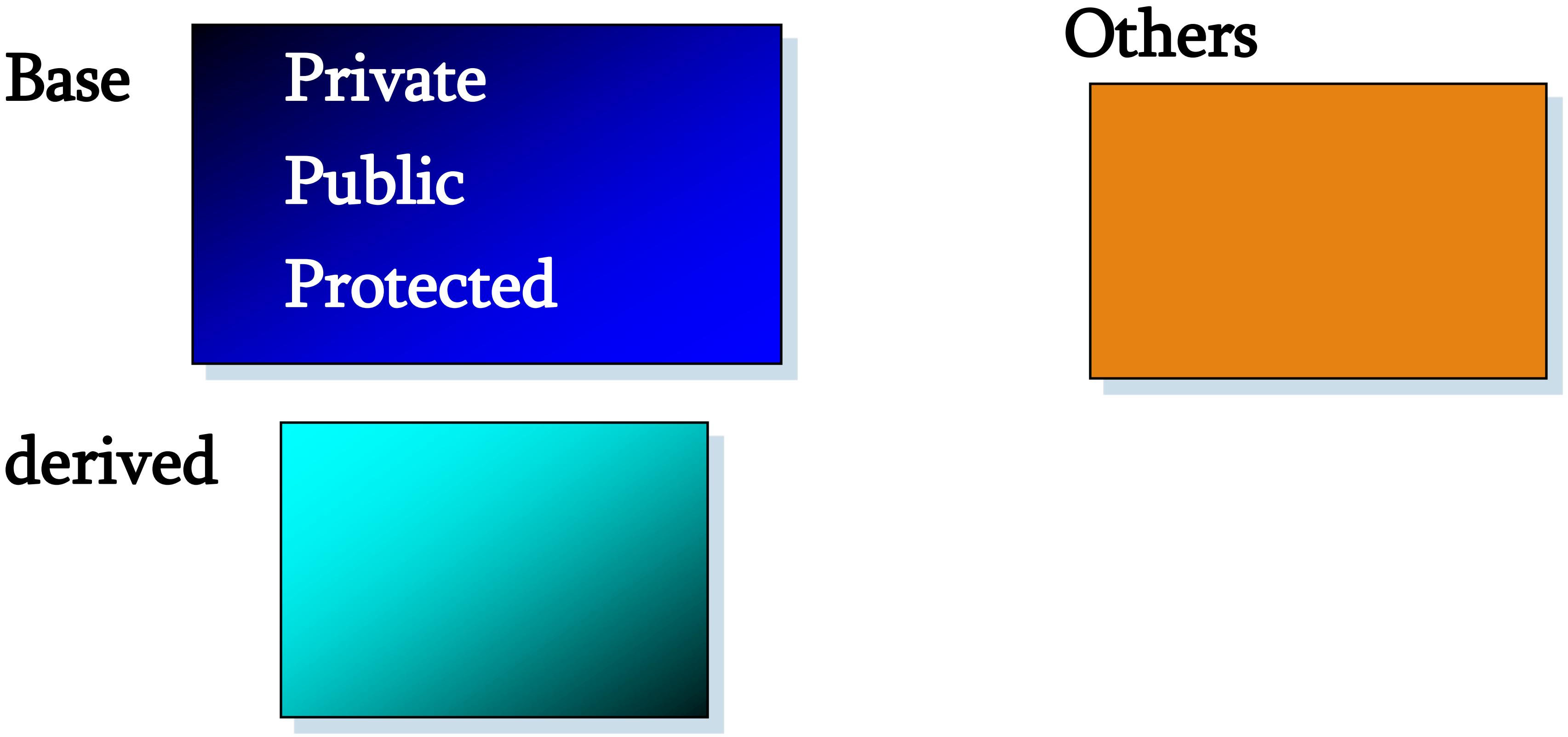
Protected Members

- ❑Protected members can be inherited
- ❑The inherited/Child class can access and use protected member functions and data; but any object from outside those classes cannot access them.
- ❑Therefore, we cannot access the protected members from any function outside the class by using an object & dot(.) operator.
- ❑In other words, protected members behave exactly same as private members with the exception that protected members can be inherited.

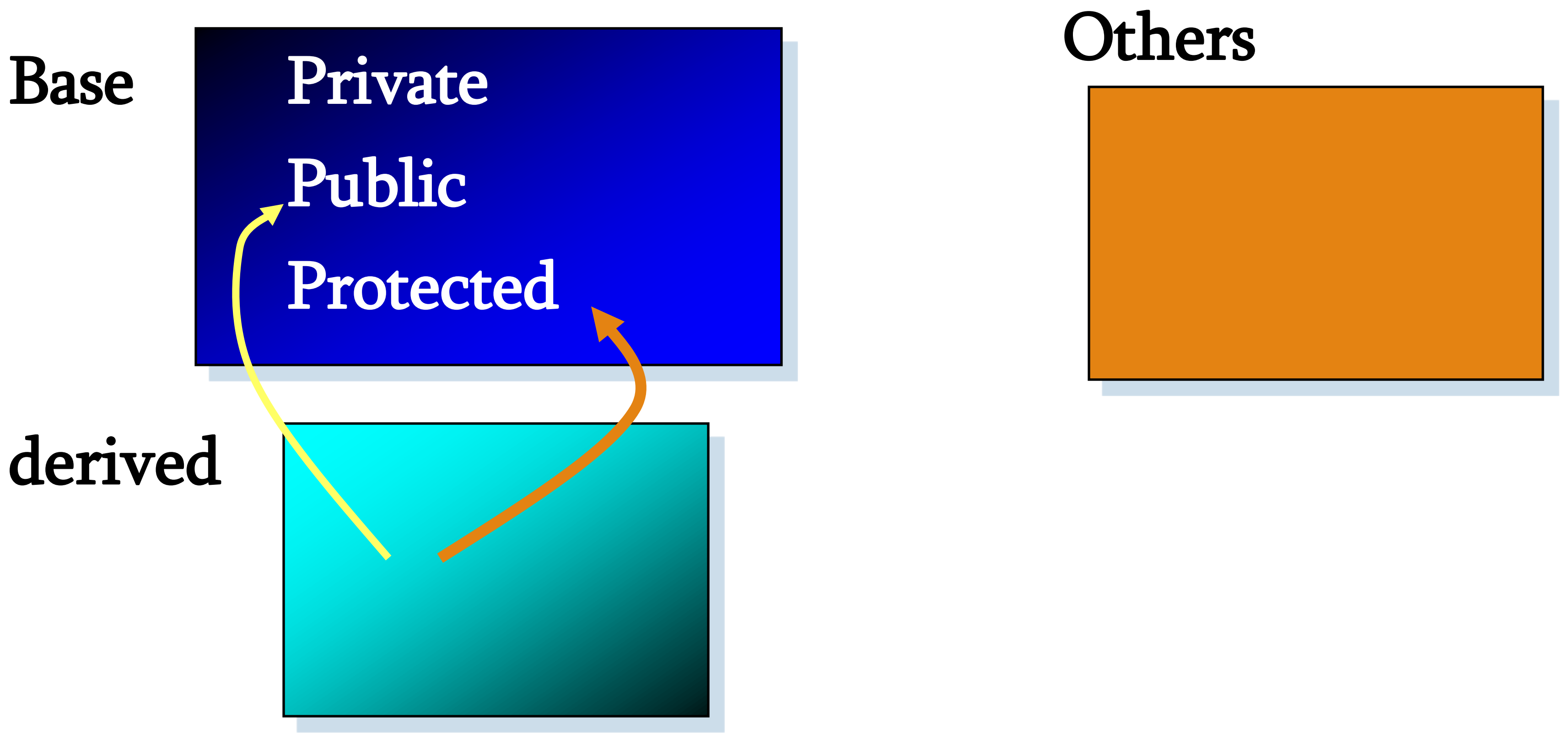
Protected Members

```
class samp
{
    int a;
    protected:int b;
    public: int c;
    samp(int m, int n){a=m, b=n;}
    int geta(){return a;}
    int getb(){return b;}
};
int main()
{   samp ob(10,20);
    //ob.b=9;    //Error as b is protected member of the class.
    ob.c=40;     //Ok
    cout<<ob.geta()<<" "<<ob.getb()<<endl;
    return 0;
}
```

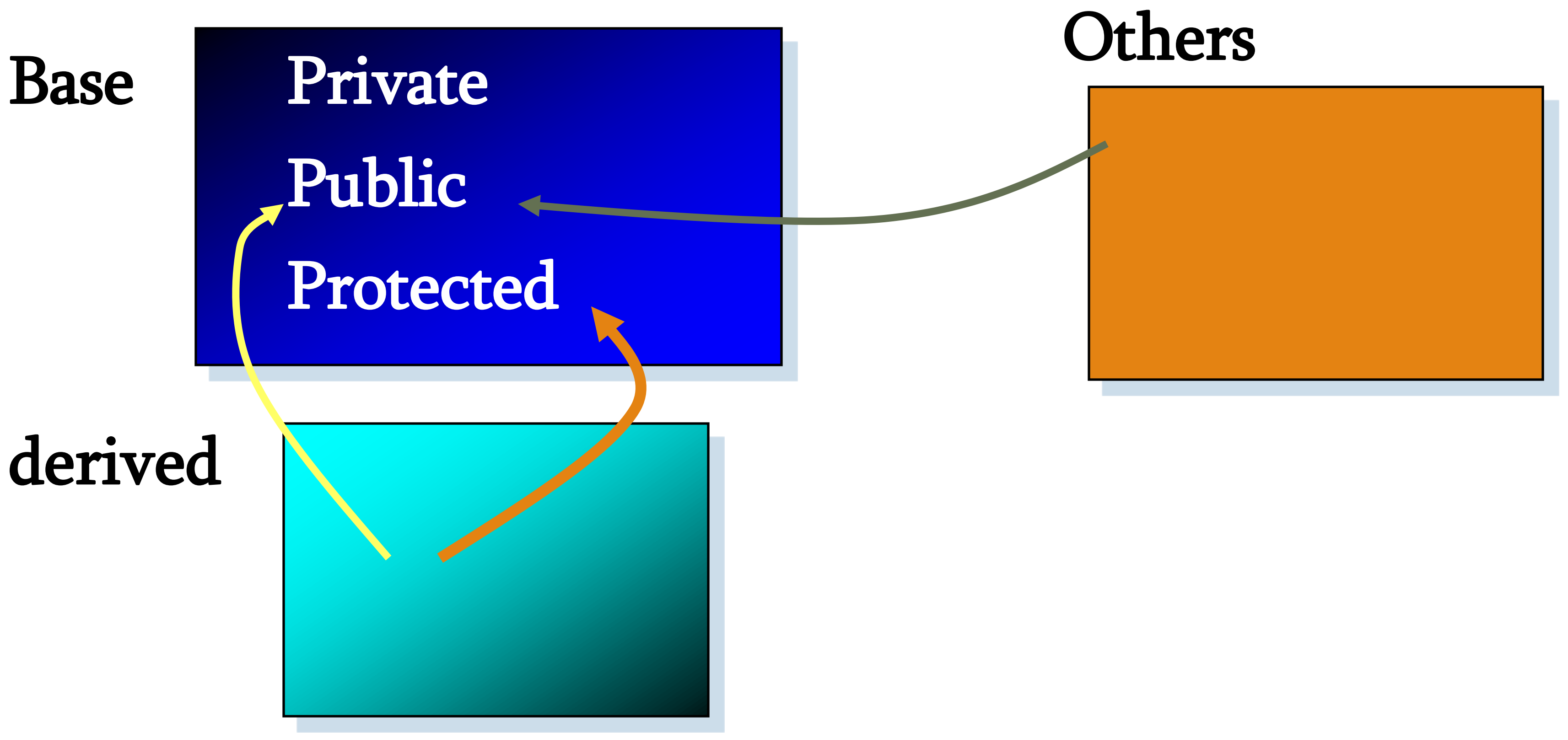
Access Methods



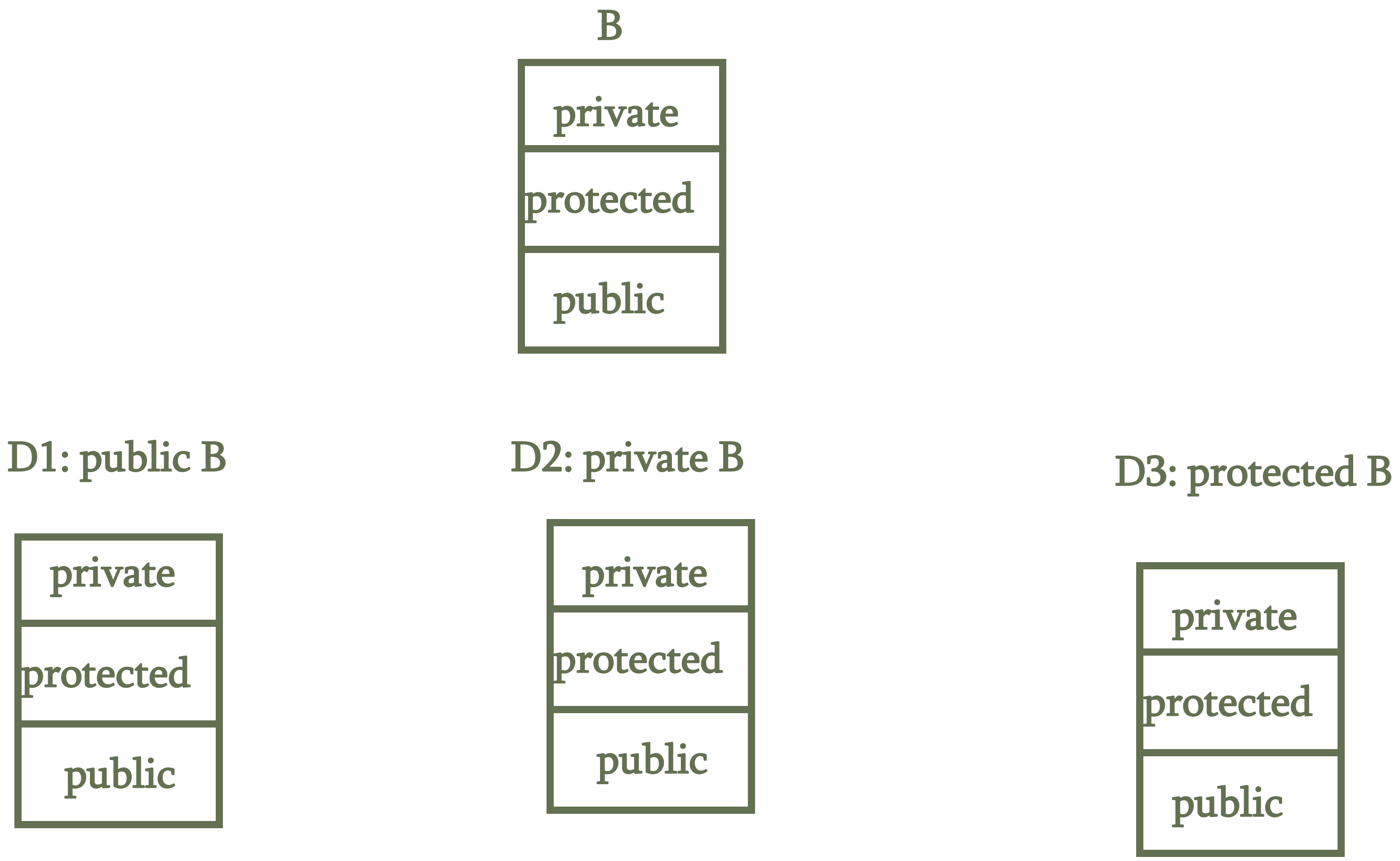
Access Methods



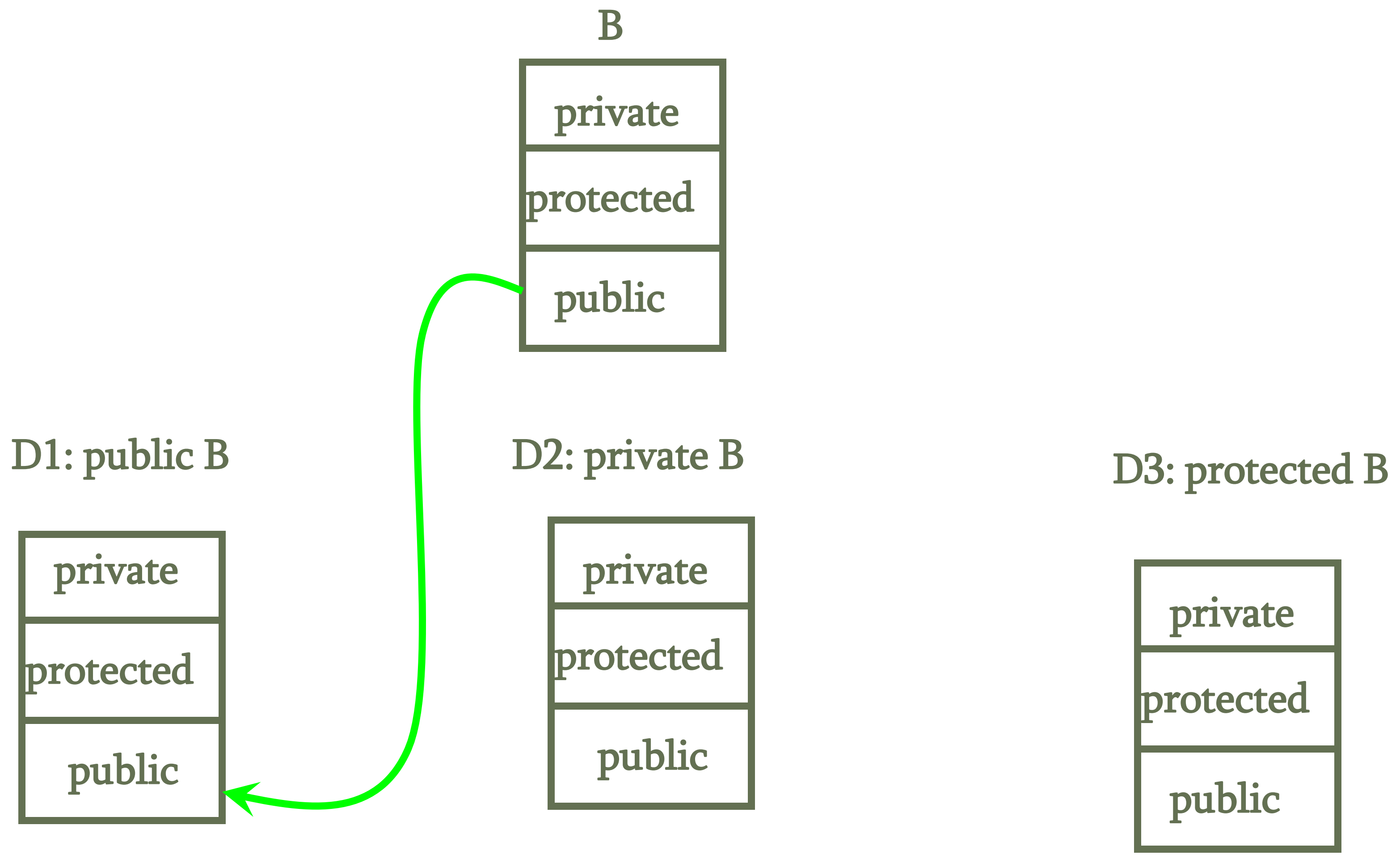
Access Methods



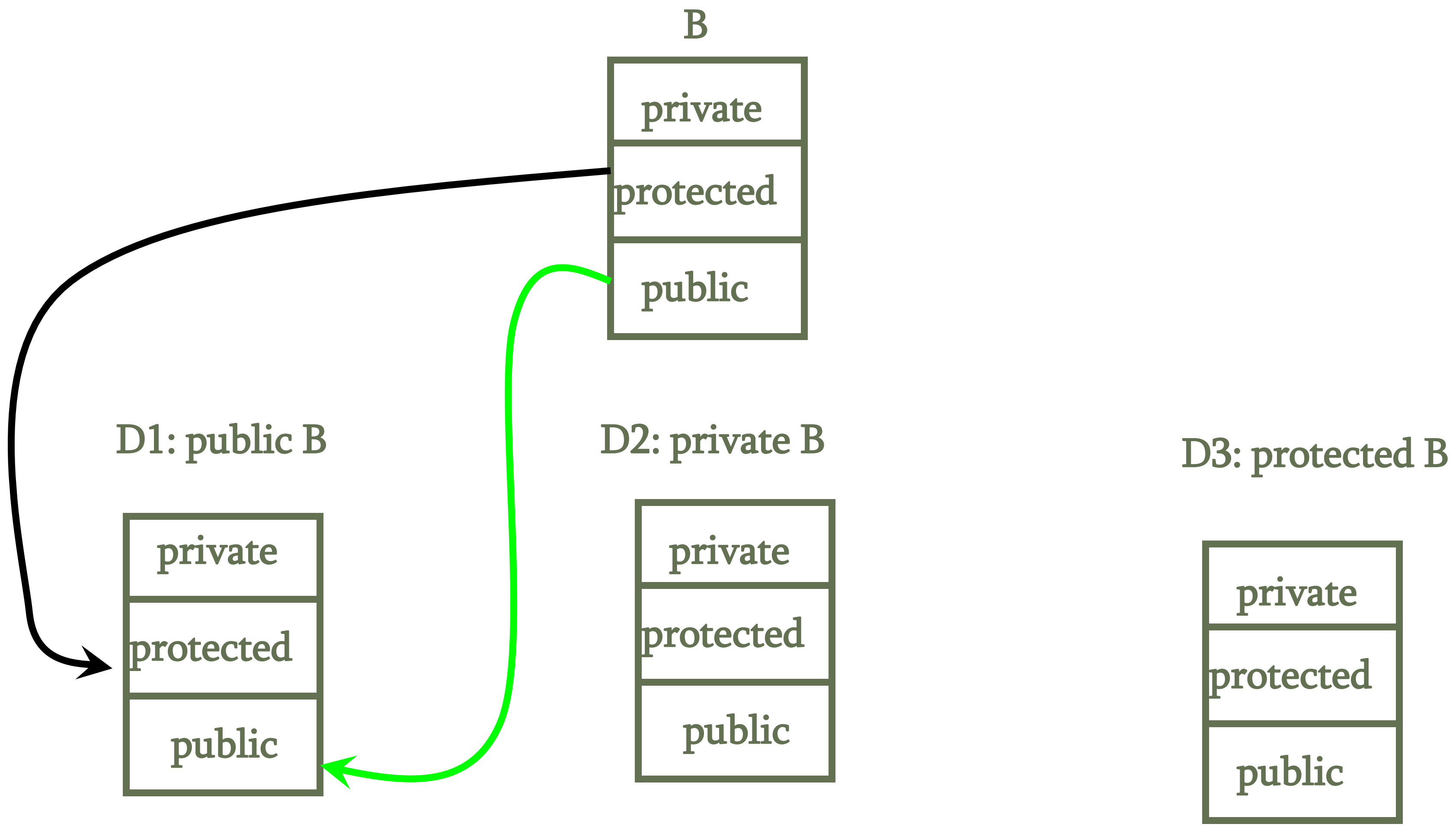
Visibility of inherited members



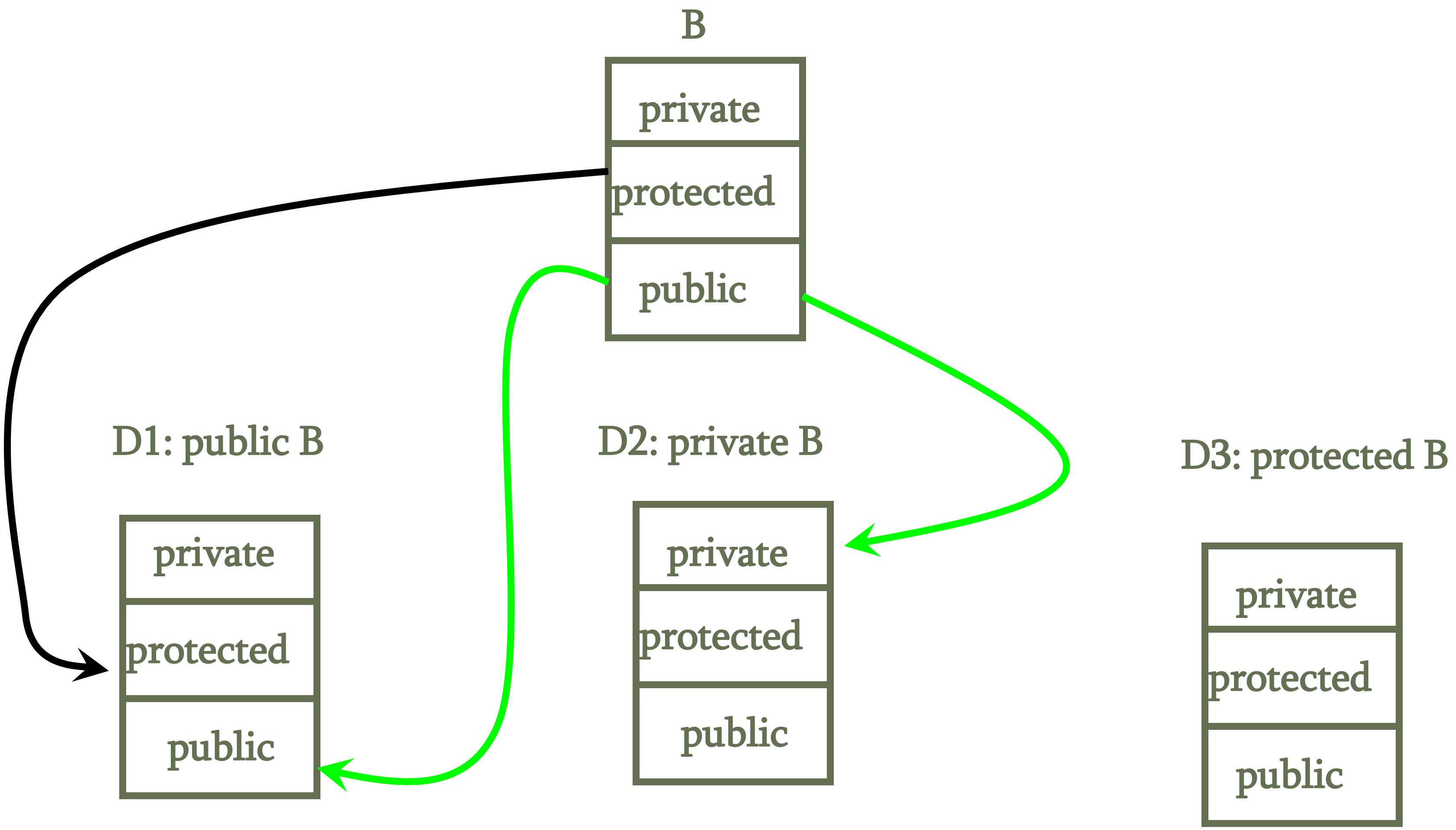
Visibility of inherited members



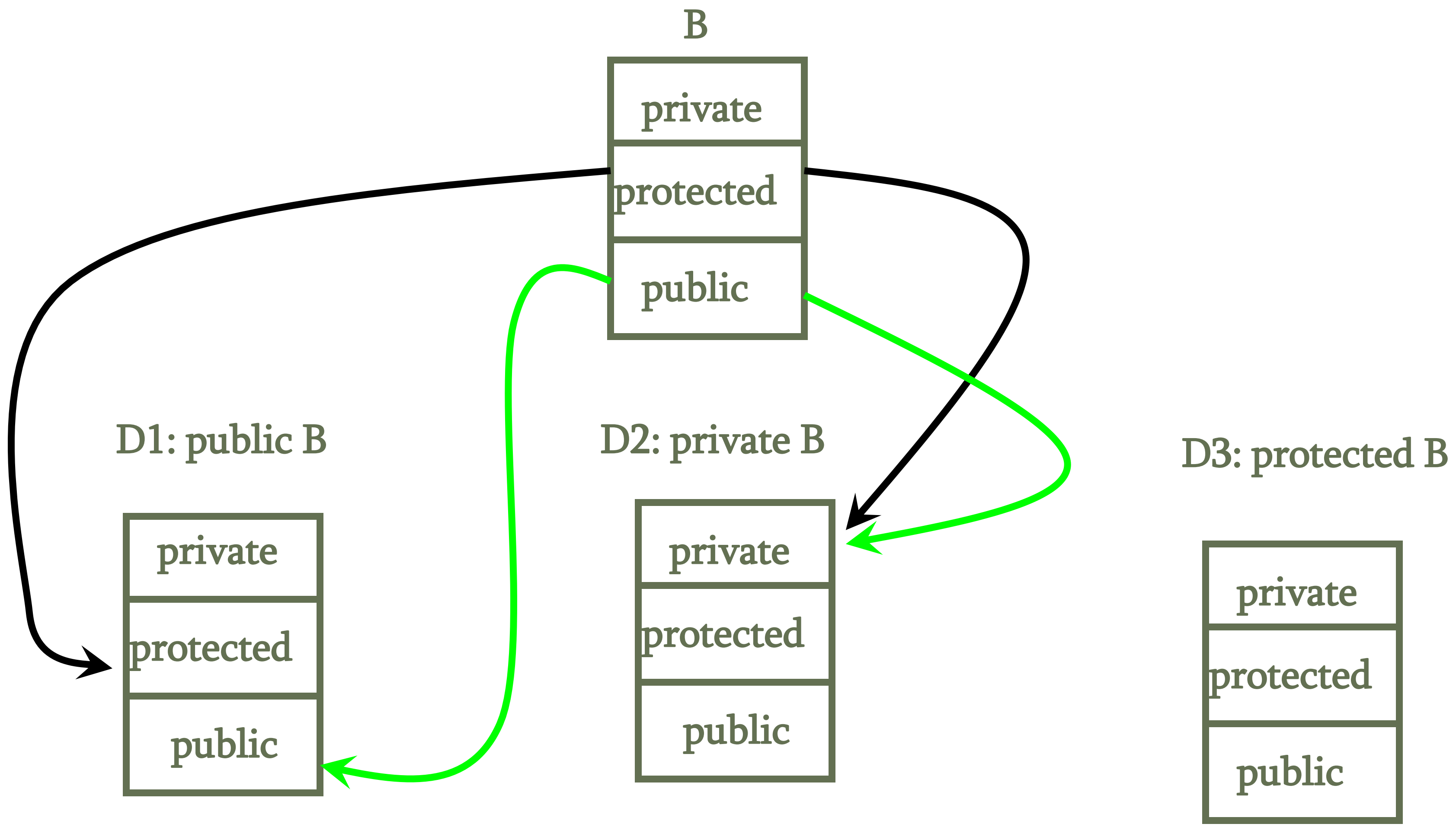
Visibility of inherited members



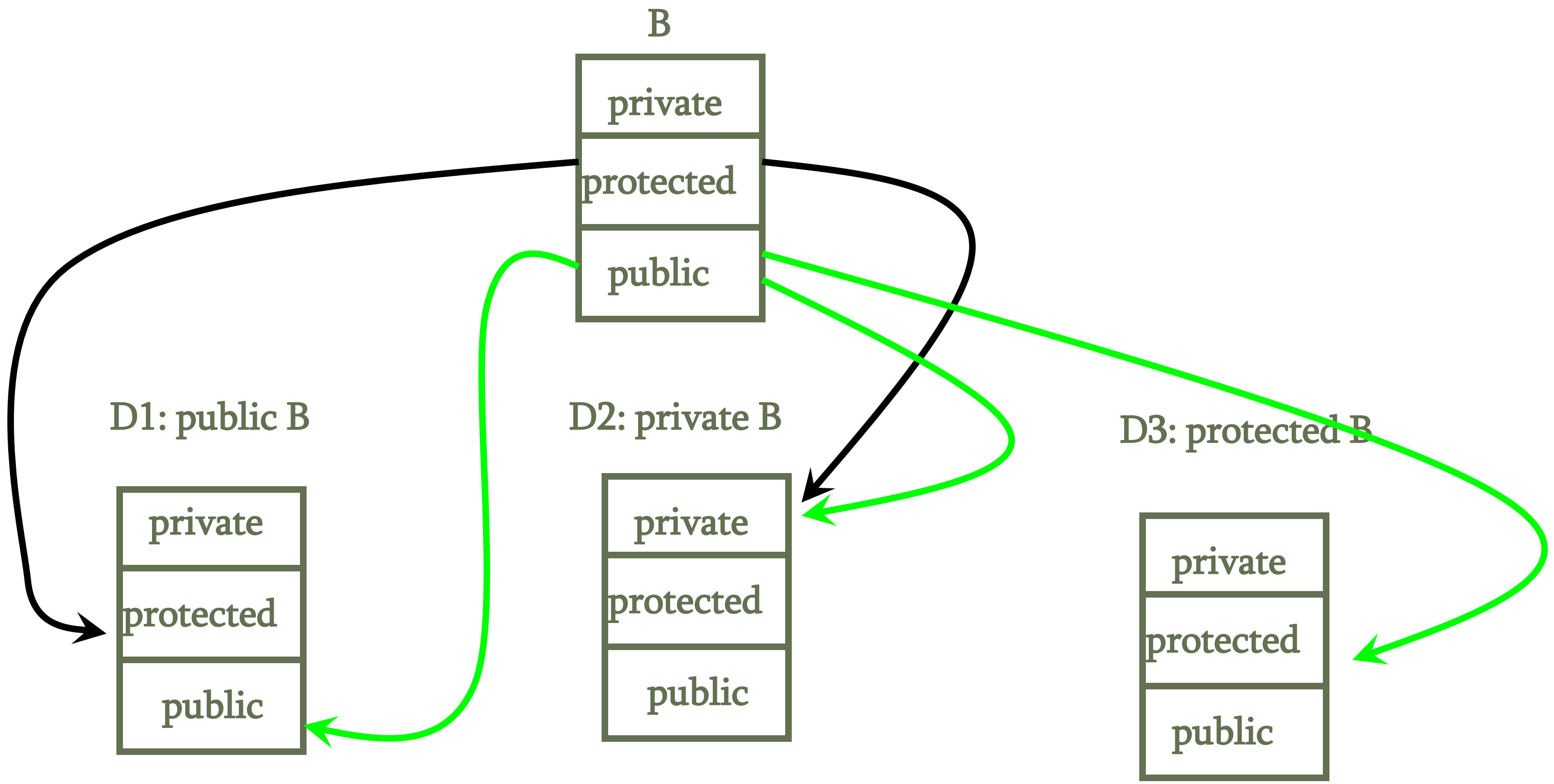
Visibility of inherited members



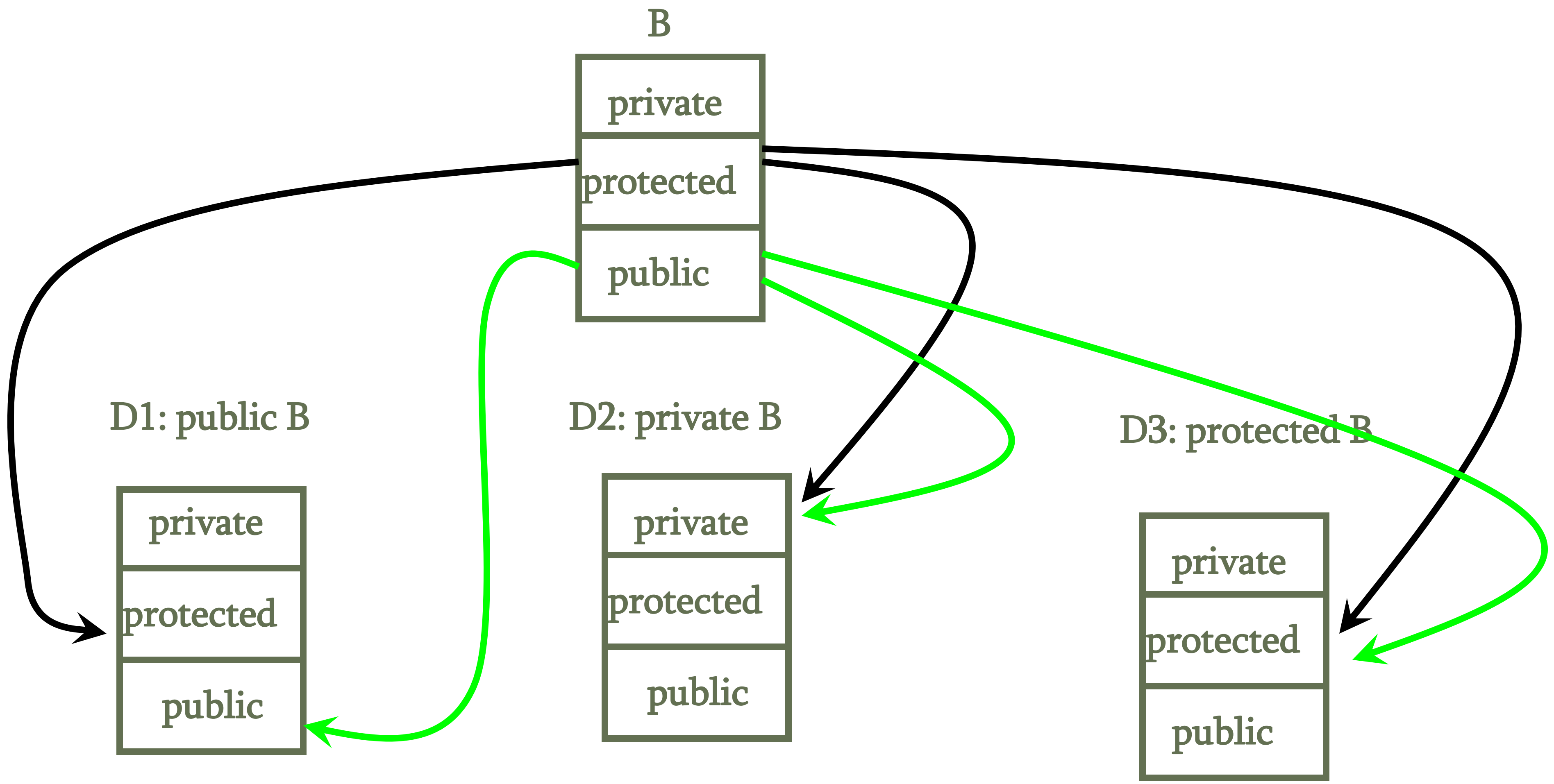
Visibility of inherited members



Visibility of inherited members



Visibility of inherited members



Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : public Person{
        private: int id;
        public:
{  void setVal (int i, char* nm, int cont, char gendr)  }
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
```

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : public Person{
        private: int id;
        public:
{ void setVal (int i, char* nm, ir          dr)  }
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

Error

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
}
```

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : public Person{
        private: int id;
        public:
 void setVal (int i, char* nm, ir          dr)  }
{
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

Error

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
```

ERROR!

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : protected Person{
        private: int id;
        public:
  void setVal (int i, char* nm, int cont, char gendr)
{
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
}
```

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : protected Person{
        private: int id;
        public:
{  void setVal (int i, char* nm, ir          dr)  }
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

Error

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
```

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : protected Person{
        private: int id;
        public:
{  void setVal (int i, char* nm, ir              dr)  }
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

Error

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
```

ERROR!

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : private Person{
        private: int id;
        public:
  void setVal (int i, char* nm, int cont, char gendr)  {
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
}
```

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : private Person{
        private: int id;
        public:
{  void setVal (int i, char* nm, in        dr)  }
                id = i;
                strcpy (name,nm);
                contactNo = cont;
                gender = gendr; }
};
```

Error

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
```

Example

```
class Person{
        private: char name[30];
        protected: char gender;
        public: int contactNo;
        // here goes member functions
};
class Student : private Person{
        private: int id;
        public:
  void setVal (int i, char* nm, ir          dr)  }
{
                  id = i;
                  strcpy (name,nm);
                  contactNo = cont;
                  gender = gendr; }
};
```

Error

```
int main(){
        Student st;
        strcpy (st.name,"Sun")
        st.contactNo = 420007;
        st.gender = 'M';
        st.id = 0;
```

ERROR!

Practice

Create the 2 classes which will have the following members:

Shape: **Private**: int height, width

Public: Create necessary setter and getter functions to set and get the private values

Rectangle: **Public**: write a function to calculate the area of the rectangle

Make the **Rectangle** class a child class of the **Shape** class. Remember that you can only create object of **Rectangle** class. By this object you have to set all the values of each shape and calculate their area. In output section you need to show height, width and area of the rectangle.

Take all information as user input.

Constructor in Inheritance

- When base class and derived class both have constructors and destructor functions, the constructor functions are executed in order of derivation.
- The destructor functions are executed in reverse order.
- That is, the base class's constructor is executed before derived class's constructor.
- Destructor functions are called in reverse order.

Inheritance – Constructor

```
class Person{
    … … …
    public:
    Person(){
        cout << "Constructing Person\n";
    }
}
 class Student : public Person{
     … … …
     public:
      Student(){
         cout << "Constructing Student\n";
     }
 };
```

```
int main{
    Student st;
}
```

Output: ?

Inheritance – Constructor

```
class Person{
    … … …
    public:
    Person(){
        cout << "Constructing Person\n";
    }
}
 class Student : public Person{
     … … …
     public:
      Student(){
         cout << "Constructing Student\n";
     }
 };
```

```
int main{
    Student st;
}
```

Output: ?

```
Constructing Person
Constructing Student
```

Inheritance – Constructor

```
class Person{
    … … …
    public:
    Person(){
        cout << “Constr
    }
}
 class Student : public P
     … … …
     public:
      Student(){
         cout << “Constructing Student\n”;
      }
 };
```

Constructor called in order of derivation

```
int main{
    Student st;
}
```

Output: ?

```
Constructing Person
Constructing Student
```

Inheritance – Destructor

```
class Person{
 public:
 ~Person(){
       cout << “Destructing Person\n”;
 }
};
 class Student : public Person{
       … … …
       public:
        ~Student(){
            cout << “Destructing Student”<<endl;
       }
 };
```

```
int main{
     Student st;
}
```

Output: ?

Inheritance – Destructor

```
class Person{
 public:
 ~Person(){
      cout << “Destructing Person\n”;
 }
};
 class Student : public Person{
      … … …
      public:
       ~Student(){
          cout << “Destructing Student”<<endl;
      }
 };
```

```
int main{
    Student st;
}
```

Output: ?

```
Destructing Student
Destructing Person
```

Inheritance – Destructor

```
class Person{
 public:
 ~Person(){
       cout << “Destructing
 }
};
 class Student : public Person{
       … … …
       public:
        ~Student(){
            cout << “Destructing Student”<<endl;
       }
 };
```

Destructor called in reverse order of derivation

```
int main{
      Student st;
}
```

Output: ?

Destructing Student

Destructing Person

Example

```
class base{
public:
   base (){cout<<"Constructor in base class"<<endl;}
   ~base (){cout<<"Destructor in base class"<<endl;}
};
class derived: public base{
public:
   derived (){cout<<"Constructor in derived class"<<endl;}
   ~derived (){cout<<"Destructor in derived class"<<endl;}
};
int main()
{
   derived ob;
}
```

Example

```
class base{
public:
   base (){cout<<"Constructor in base class"<<endl;}
   ~base (){cout<<"Destructor in base class"<<endl;}
};
class derived: public base{
public:
   derived (){cout<<"Constructor in derived class"<<endl;}
   ~derived (){cout<<"Destructor in derived class"<<endl;}
};
int main()
{
   derived ob;
}
```

```
Constructor in base class
Constructor in derived class
Destructor in derived class
Destructor in base class
```

Example

```
class base{
public:
   base (){cout<<"Constructor in base class"<<endl;}
   ~base (){cout<<"Destructor in base class"<<endl;}  };
class derived: public base{
   private: int a;
public:
   derived(int x){
      cout<<"Constructor in derived class"<<endl;
      a=x;  }
   ~derived(){cout<<"Destructor in derived class"<<endl;}
   void showa() { cout<<a<<endl;}   };
int main()
{ derived ob(100);
  ob.showa();
}
```

Example

```
class base{
public:
   base (){cout<<"Constructor in base class"<<endl;}
   ~base (){cout<<"Destructor in base class"<<endl;}  };
class derived: public base{
   private: int a;
public:
   derived(int x){
      cout<<"Constructor in derived class"<<endl;
      a=x;  }
   ~derived(){cout<<"Destructor in derived class"<<endl;}
   void showa() { cout<<a<<endl;}   };
int main()
{ derived ob(100);
  ob.showa();
}
```

```
Constructor in base class
Constructor in derived class
100
Destructor in derived class
Destructor in base class
```

Example

```
class base{
private: int y;
public:
   base (int n){cout<<"Constructor in base class"<<endl;
                  y=n;}
   ~base (){cout<<"Destructor in base class"<<endl;}  };
class derived: public base{
   private: int a;
public:
   derived (int x){
      cout<<"Constructor in derived class"<<endl;
      a=x;  }
   ~derived(){cout<<"Destructor in derived class"<<endl;}
   void showa() { cout<<a<<endl;}   };
int main()
{ derived ob(100);
}
```

Example

```
class base{
private: int y;
public:
   base (int n){cout<<"Constructor in base class"<<endl;
                  y=n;}
   ~base (){cout<<"Destructor in base class"<<endl;}  };
class derived: public base{
   private: int a;
public:
   derived (int x){
      cout<<"Constructor in derived class"<<endl;
      a=x;  }
   ~derived(){cout<<"Destructor in derived class"<<endl;}
   void showa() { cout<<a<<endl;}   };
int main()
{ derived ob(100);
}
```

ERROR!!

Example

```
class base{
private: int y;
public:
   base (int n){cout<<"Constructor in base class"<<endl;
                 y=n;}
   ~base (){cout<<"Destructor in base class"<<endl;}  };
class derived: public base{
   private: int a;
public:
   derived (int x){
      cout<<"Constructor in derived class"<<endl;
      a=x;  }
   ~derived(){cout<<"Destructor in derived class"<<endl;}
   void showa() { cout<<a<<endl;}   };
int main()
{ derived ob(100);
}
```

ERROR!!
Because there is no matching function to call the constructor function of the base class.

Example

```
class base{
private: int y;
public:
   base (int n){cout<<"Constructor in base class"<<endl;
                  y=n;}
   ~base (){cout<<"Destructor in base class"<<endl;}  };
class derived: public base{
   private: int a;
public:
    derived (int x, int y) : base (y) {
      cout<<"Constructor in derived class"<<endl;
      a=x;  }
   ~derived(){cout<<"Destructor in derived class"<<endl;}
   void showa() { cout<<a<<endl;}   };
int main()
{ derived ob(100, 200);
}
```

Example

```
class base{
private: int y;
public:
   base (int n){cout<<"Constructor in base class"<<endl;
                  y=n;}
   ~base (){cout<<"Destructor in base class"<<endl;}  };
class derived: public base{
   private: int a;
public:
    derived (int y) : base (y) {
      cout<<"Constructor in derived class"<<endl;
      a=0;  }
   ~derived(){cout<<"Destructor in derived class"<<endl;}
   void showa() { cout<<a<<endl;}   };
int main()
{ derived ob(100);
}
```