

#include<iostream>

#include<conio.h>

using namespace std;

class student

{

protected :

char name[15]; int roll;

public:

void get();

void disp();

};

class mark: public student

{

protected : int m,e;

public:

void get();

void disp();

};

class result: public mark

{

private : int t;

public:

void get();

void disp();

};

void student::get()

{

cout<<"\n Enter name of the student : \t";

cin>>name;

cout<<"\n Enter roll number : \t";

cin>>roll;

}

void student::disp()

{

cout<< name <<" \t "<< roll <<" \t ";

}

void mark::get()

{

student::get();

cout<<"\n Mark in English in 50 : \t ";

cin>>e;

cout<<"\n Mark in Maths in 50 : \t ";

cin>>m;

}

void mark::disp()

{

student::disp();

cout << e << " \t " << m << " \t " ;

}

void result::get()

{

mark::get(); t=e + m;

}

void result::disp()

{

mark::disp();

if (t >= 36)

cout<<"PASSED \n";

else cout<<"FAILED \n";

}

int main()

{

int i,n; result r[5]; cout<<"How many students :"; cin>>n; for(i=0;i<n;i++) r[i].get();

cout<<"\n\nName\tRoll\tEnglish\tMaths\tResult\n---------------------------------------\n";

for(i=0;i<n;i++)

{

r[i].disp();

}

getch();

}

[Difference between private, public, and protected inheritance](http://stackoverflow.com/questions/860339/difference-between-private-public-and-protected-inheritance)

There are three accessors that I'm aware of: public, protected and private.

Let:

class Base {

public:

int publicMember;

protected:

int protectedMember;

private:

int privateMember;

};

* Everything that is aware of Base is also aware that Base contains publicMember.
* Only the children (and their children) are aware that Base contains protectedMember.
* No one but Base is aware of privateMember.

By "is aware of", I mean "acknowledge the existence of, and thus be able to access".

The same happens with public, private and protected inheritance. Let's consider a class Base and a class Child that inherits from Base.

* If the inheritance is public, everything that is aware of Base and Child is also aware that Child inherits from Base.
* If the inheritance is protected, only Child, and its children, are aware that they inherit from Base.
* If the inheritance is private, no one other than Child is aware of the inheritance.

class A

{

public:

int x;

protected:

int y;

private:

int z;

};

class B : public A

{

// x is public

// y is protected

// z is not accessible from B

};

class C : protected A

{

// x is protected

// y is protected

// z is not accessible from C

};

class D : private A

{

// x is private

// y is private

// z is not accessible from D

};

IMPORTANT NOTE: Classes B, C and D all contain the variables x, y and z. It is just question of access.