**20103153\_Tutorial\_6**

Q1.

D. In Derived

In Base

Q2.

D. In Derived

In Base

Q3.

B. There is compiler error in line “Base b;”

Q4.

B. Compiler Error: Derived is abstract

Q5.

B. No

Q6.

A. a > b

Q7.

A. Constructor: Base

Constructor: Derived

Destructor: Derived

Destructor: Base

Q8.

C. C::fun()

Q9.

A. In Base

Q10.

#include<iostream>

#include<string>

using namespace std;

class shape

{

public:

int a;

int b;

virtual void Area()=0;

virtual void Perimeter()=0;

};

class rectangle:public shape

{

public:

rectangle()

{

cout<<"Enter length and breadth of Rectangle ";

cin>>a>>b;

Area();

Perimeter();

}

void Area()

{

cout<<"Area of Rectangle is "<<a\*b<<" sq units"<<endl;

}

void Perimeter()

{

cout<<"Perimeter of Rectangle is "<<2\*(a+b)<<" units"<<endl<<endl;

}

};

class circle:public shape

{

public:

circle()

{

cout<<"Enter radius of circle ";

cin>>a;

Area();

Perimeter();

}

void Area()

{

cout<<"Area of Circle is "<<3.14\*a\*a<<" sq units"<<endl;

}

void Perimeter()

{

cout<<"Perimeter of Circle is "<<2\*3.14\*a<<" units"<<endl<<endl;

}

};

class square:public shape

{

public:

square()

{

cout<<"Enter side of square ";

cin>>a;

Area();

Perimeter();

}

void Area()

{

cout<<"Area of Square is "<<a\*a<<" sq units"<<endl;

}

void Perimeter()

{

cout<<"Perimeter of Square is "<<4\*a<<" units"<<endl<<endl;

}

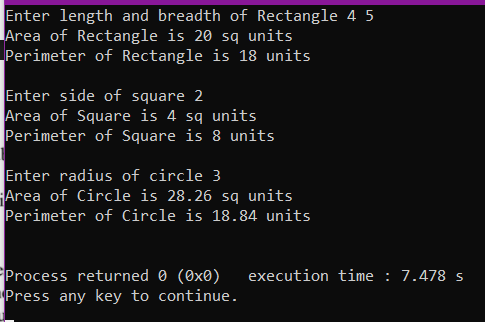
};

int main()

{

shape \*s=new rectangle,\*s1=new square,\*s2=new circle;

}



Q11.

#include<iostream>

#include<string>

using namespace std;

class Person{

public:

string name;

public:

int age;

virtual void getdata(){}

virtual void putdata(){}

virtual int getID(){return 0;}

};

class Professor: public Person {

static int id;

protected:

int publications;

int current\_id;

public:

void getdata () {

std::cin >> name >> age >> publications;

current\_id = ++id;

}

void putdata() {

cout<< name << " " << age << " " << publications<< " " << current\_id << "\n";

}

};

int Professor::id = 0;

class Student: public Person{

float marks[6];

static int id;

int getID(){return id;}

public:

void getdata () {

cin >> name;

cin >> age;

for(int i =0 ; i< 6; i++){

cin >> marks[i];

}

}

public:

void putdata(){

id++;

float markSum = 0;

for(int i =0 ; i < 6; i++){

markSum += marks[i];

}

cout << name << " "<< age << " "<<markSum<<" "<<getID()<<"\n";

}

};

int Student::id = 0;

int main(){

int n, val;

cin>>n; //The number of objects that is going to be created.

Person \*per[n];

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

cin>>val;

if(val == 1){

// If val is 1 current object is of type Professor

per[i] = new Professor;

}

else per[i] = new Student;

per[i]->getdata();

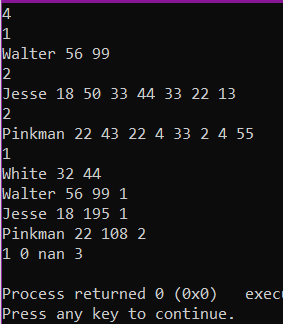
}

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

per[i]->putdata();

return 0;

}



Q12.

Virtual destructor is used to free memory space of derived class objects that were initiated using base class pointer objects. It is declared by using a “virtual” keyword in the base class destructor. It mainly aims at destroying not only the base class object/pointer object on calling it once but also destroying the objects associated with that base class pointer along with it. This is the speciality of virtual destructors.