**Abstract Classes**

**Writer:** Sayak Haldar

**1.Which class is used to design the base class?**

a) abstract class

b) derived class

c) base class

d) none of the mentioned

**Answer: a**

**2. Which is used to create a pure virtual function ?**

a) $

b) =0

c) &

d) !

**Answer: b**

Explanation: For making a method as pure virtual function, We have to append ‘=0’ to the class or method.

**3. What is the output of this program?**

#include <iostream>

using namespace std;

class p

{

protected:

int width, height;

public:

void set\_values (int a, int b)

{

width = a; height = b;

}

virtual int area (void) = 0;

};

class r: public p

{

public:

int area (void)

{

return (width \* height);

}

};

class t: public p

{

public:

int area (void)

{

return (width \* height / 2);

}

};

int main ()

{

r rect;

t trgl;

p \* ppoly1 = &rect;

p \* ppoly2 = &trgl;

ppoly1->set\_values (4, 5);

ppoly2->set\_values (4, 5);

cout << ppoly1 -> area() ;

cout << ppoly2 -> area();

return 0;

}

a) 1020

b) 20

c) 10

d) 2010

**Answer: d**

Explanation: In this program, We are calculating the area of rectangle and

triangle by using abstract class.

Output:

$ g++ abs.cpp

$ a.out

2010

Now, this is one of the way, how we can avoid having constructor in abstract class (you cannot directly instantiate base class. But, derived class invoke that indirectly.

)

**What do I mean In the last sentence?**

#include <iostream>

using namespace std;

class p

{

protected:

int width, height;

public:

void set\_values (int a, int b)

{

width = a; height = b;

}

virtual int area (void) = 0;

};

class r: public p

{

public:

int area (void)

{

return (width \* height);

}

};

class t: public p

{

public:

int area (void)

{

return (width \* height / 2);

}

};

int main ()

{

r rect;

t trgl;

p \* ppoly1 = &rect;

p \* ppoly2 = &trgl;

ppoly1->set\_values (4, 5);

ppoly2->set\_values (4, 5);

cout << ppoly1 -> area() ;

cout << ppoly2 -> area();

return 0;

}

You can either have this. Or, the following:

#include <iostream>

#include<cstdio>

using namespace std;

class p

{

protected:

int width, height;

public:

p(int width,int height)

{

this->width=width;

this->height=height;

}

virtual int area (void) = 0;

};

class r: public p

{

public:

r(int w,int h):p(w,h)

{

}

int area (void)

{

return (width \* height);

}

};

class t: public p

{

public:

t(int w,int h):p(w,h)

{

}

int area (void)

{

return (width \* height / 2);

}

};int main ()

{

int ch;

for(int i=0;i<2;i++)

{

printf("Enter your choice. 1 for rectangle, 2 for triangle\n");

scanf("%d",&ch);

p \*polygon;

switch(ch)

{

case 1:

{

printf("A new rectangle will be created\n");

//for simplicity. Otherwise, you can scan the arguments, too

polygon=new r(12,15);

printf("The rectangle's area is: %d\n",polygon->area());

break;

}

case 2:

{

printf("A new triangle will be created\n");

polygon=new t(12,15);

printf("The triangle's area is: %d\n",polygon->area());

break;

}

default:

{

printf("Invalid choice\n");

break;

}

}

}

}

**5. What is the output of this program?**

#include <iostream>

using namespace std;

class MyInterface

{

public:

virtual void Display() = 0;

};

class Class1 : public MyInterface

{

public:

void Display()

{

int a = 5;

cout << a;

}

};

class Class2 : public MyInterface

{

public:

void Display()

{

cout <<" 5" << endl;

}

};

int main()

{

Class1 obj1;

obj1.Display();

Class2 obj2;

obj2.Display();

return 0;

}

a) 5

b) 10

c) 5 5

d) None of the mentioned

**Answer: c**

Explanation: In this program, We are displaying the data from the two classes

by using abstract class.

Output:

$ g++ abs1.cpp

$ a.out

5 5

**6. What is the output of this program?**

#include <iostream>

using namespace std;

class sample

{

public:

virtual void example() = 0;

};

class Ex1:public sample

{

public:

void example()

{

cout << "ubuntu";

}

};

class Ex2:public sample

{

public:

void example()

{

cout << " is awesome";

}

};

int main()

{

sample\* arra[2];

Ex1 e1;

Ex2 e2;

arra[0]=&e1;

arra[1]=&e2;

arra[0]->example();

arra[1]->example();

}

a) ubuntu

b) is awesome

c) ubuntu is awesome

d) none of the mentioned

**Answer: c**

Explanation: In this program, We are combining the two statements from two classes and printing it by using abstract class.

Output:

$ g++ abs3.cpp

$ a.out

ubuntu is awesome

**7. What is the output of this program?**

#include <iostream>

using namespace std;

class Base

{

public:

virtual void print() const = 0;

};

class DerivedOne : virtual public Base

{

public:

void print() const

{

cout << "1";

}

};

class DerivedTwo : virtual public Base

{

public:

void print() const

{

cout << "2";

}

};

class Multiple : public DerivedOne, DerivedTwo

{

public:

void print() const

{

DerivedTwo::print();

}

};

int main()

{

Multiple both;

DerivedOne one;

DerivedTwo two;

Base \*array[ 3 ];

array[ 0 ] = &both;

array[ 1 ] = &one;

array[ 2 ] = &two;

for ( int i = 0; i < 3; i++ )

array[ i ] -> print();

return 0;

}

a) 121

b) 212

c) 12

d) none of the mentioned

**Answer: b**

Explanation: In this program, We are executing these based on the condition given in array. So it is printing as 212.

Output:

$ g++ abs4.cpp

$ a.out

212

**8. What is meant by pure virtual function?**

a) Function which does not have definition of its own

b) Function which does have definition of its own

c) Function which does not have any return type

d) None of the mentioned

Answer: a

**9. Pick out the correct option.**

a) We cannot make an instance of an abstract base class

b) We can make an instance of an abstract base class

c) We can make an instance of an abstract super class

d) None of the mentioned

**Answer: a**

Explanation: None.

**10. Where does the abstract class is used?**

a) base class only

b) derived class

c) both derived & base class

d) none of the mentioned

**Answer: a**