| Philadelphia University<br>Faculty of Engineering<br>Department of Computer Engineering | | Date:- 05/05/2015<br>Allowed time:- 60 minutes |
|---|---|---|

**Object Oriented Programming (630221) second Exam**

**Student Name: - …….....  ID: - ……**

**Question 1: mark the following statements as true or false. 5 Points**

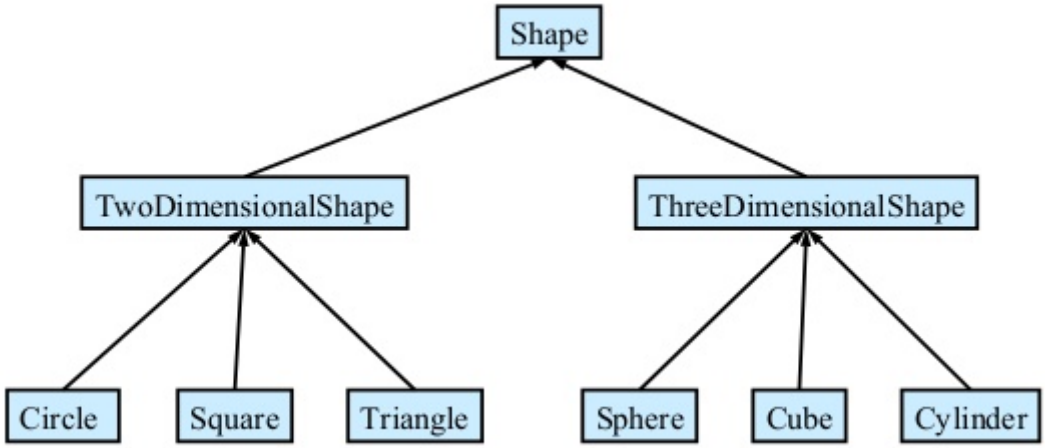
| | |
|---|---|
| Copy constructor and assignment operator = are the same. | F |
| Abstract class is the class that all its methods are pure virtual methods | F |
| Friend function cannot access private data members of the class but can access the protected and public members | F |
| To override operator << we can use either friend or member function. | F |
| Template can create a general code to be used with different data types. | T |

**Question 2 : Given the following class convert it with its methods to template format 3 Points**

```
class  Test
{
public:
      Test(int,int);
      int Get_x();
      int Get_y();
      void Set(int,int);
private:
      int x,y;
};
Test::Test(int a,int b)
{
      x=a;
      y=b;
}
int test::Get_x()
{
return x;
}
int test::Get_y()
{
return y;
}
void Test::Set(int a,int b)
{
      x=a;
      y=b;
}
```

```
template<class T>class  Test
{
public:
       Test(T,T);
       T Get_x();
       T Get_y();
       void Set(T,T);
private:
       T x,y;
};
template<class T> Test<T>::Test(T
a,T b)
{
       x=a;
       y=b;
}
template<class T> T Test<T>::Get_x()
{
return x;
}
template<class T> T Test<T>::Get_y()
{
return y;
}
template<class T>void Test<T>::Set(T
a,T b)
{
       x=a;
       y=b;
}
```

**Questing 2: Given the following hierarchy of class** **6 Points**




Define the classes in the hierarchy above using the appropriate data member and methods as the following.

a. Classes **shape , TowDimensionalShape** and **ThreeDimensionalShape** are abstract classes
b. **Shape** class should define **print** method.
c. **TowDimensionalShape** should define **area** method
d. **ThreeDimensionalShape** should define **volume** and **surface_area** methods

```
class Shape
{
public:
	Shape(string s){name=s;}
	virtual void print()=0;
protected:
	string name;
};

class TwoDimensionalShape:public Shape
{
public:
	TwoDimensionalShape(string s):Shape(s){}
	virtual float area()=0;
};
class ThreeDimensionalShape:public Shape
{
public:
	ThreeDimensionalShape(string s):Shape(s){}
	virtual float volume()=0;
	virtual float surface_srea()=0;
};
class Circle:public TwoDimensionalShape
{
public:
	Circle(float r):TwoDimensionalShape("Circle"){radius=r;}
	void print()
	{	cout<<"name:- "<<name<<endl;
		cout<<"radius="<<radius<<"\tarea="<<area()<<endl;
	}
	float area(){return 3.14*pow(radius,2);}
protected:
	float radius;
};
class Square:public TwoDimensionalShape
{
public:
	Square(float s):TwoDimensionalShape("Square"){side=s;}
```

```
	void print()
	{	cout<<"name:- "<<name<<endl;
		cout<<"side="<<side<<"\tarea="<<area()<<endl;
	}
	float area(){return pow(side,2);}
protected:
	float side;
};
class Triangle:public TwoDimensionalShape
{
public:
	Triangle(float b,float h):TwoDimensionalShape("Triangle")
	{	base=b;
		hight=h;
	}
	void print()
	{	cout<<"name:- "<<name<<endl;
		cout<<"base="<<base<<"\thight="<<hight<<"\tarea="<<area()<<endl;
	}
	float area(){return 0.5*base*hight;}
protected:
	float base;
	float hight;
};
class Sphere:public ThreeDimensionalShape
{
	Sphere(float r):ThreeDimensionalShape("Sphere"){radius=r;}
	void print()
	{	cout<<"name:- "<<name<<endl;
		cout<<"radius="<<radius<<"\tvolume="<<volume()
			<<"surface_area="<<surface_area()<<endl;
	}
	float volume()
	{	return 4/3*3.14*pow(radius,3);
	}
	float surface_area()
	{	return 4*3.14*pow(radius,2);
	}
protected:
	float radius;
};
class Cube:public ThreeDimensionalShape
{
	Cube(float s):ThreeDimensionalShape("Cube"){side=s;}
	void print()
	{	cout<<"name:- "<<name<<endl;
		cout<<"side="<<side<<"\tvolume="<<volume()<<"surface_area="<<surface_area()<<endl;
	}
	float volume()
	{	return pow(side,3);
	}
	float surface_area()
	{	return 6*pow(side,2);
	}
protected:
	float side;
};
class Cylinder:public ThreeDimensionalShape
{
	Cylinder(float r,float h):ThreeDimensionalShape("Cylinder")
	{	radius=r;
		hight=h;
	}
	void print()
	{	cout<<"name:- "<<name<<endl;
```

```
            cout<<"radius="<<radius<<"\thight="<<hight
                  <<"\tvolume="<<volume()<<"surface_area="<<surface_area()<<endl;
      }
      float volume()
      {     return 3.14*pow(radius,2)*hight;
      }
      float surface_area()
      {     return 2*3.14*pow(radius,2)+2*3.14*radius*hight;
      }
protected:
      float radius;
      float hight;
};
```

---

**Question 4: Given the following class classA perform the following:** **6 Points**

1. override operator << so you can output the value of x and y respectively
2. override the operator += where object a +=b is a=a+b;
3. override the operator == which will return Boolean value as a friend function
4. override the operator != which will return a Boolean value as a member method.

```
class classA
{
      public:
       void print() const;
      classA(int a,int b)
       {x=a;y=b;}
private:
      int x;
      int y;
};
```

```
class classA
{     friend ostream& operator<<(ostream& out,classA& temp);
      friend bool operator==(classA& t1,classA& t2);
public:
      void operator+=(classA& t)
      {     x=x+t.x;
            y=y+t.y;
      }
      bool operator!=(classA& t)
      {     if(x!=t.x || y!=t.y)
                  return true;
            return false;
      }
      void print() const;
      classA(int a,int b){x=a;y=b;}
private:
      int x;
      int y;
};
ostream& operator<<(ostream& out,classA& temp)
{     out<<"x="<<temp.x<<"\ty="<<temp.y<<endl;
      return out;
}
bool operator==(classA& t1,classA& t2)
{     if(t1.x==t2.x && t1.y==t2.y)
            return true;
      return false;
}
```

---

**Good Luck**

*Eng. Sultan M. Al-Rushdan*