**C++ Interlude 1 Answers**

**1. What is a class template?**

A class template allows the user to separate the functionality of a class from the type of data that it operates on. A class template is a pattern or blueprint that describes how to build a family of classes that all behave basically the same. The client programmer can specify the data type they want to use, and the compiler automatically generates the appropriate class.

2. Write a class definition (not a class template) for a class called *Circle*.A *Circle* is represented by its location in an x-y plane (also called the Cartesian plane) and its radius.

All data members should be private and all methods should be public. Only include prototypes for the methods - the implementations will be written in the next question. The class should include:

* Three data members called *xPos*, *yPos* and *radius*. These should all be double values. They represent the circle x position, y position and radius, respectively.
* A default (no argument) constructor that sets the x and y positions to zero, and the radius to one.
* A parameterized constructor that has parameters that allow the client to specify the initial values for the circle x position, y position and radius. This method should call the set methods (described below) to actually store the parameters in the data members.
* A mutator (set) method called *setPosition* that has two double parameters. These parameters should be used to update the *xPos* and *yPos* data members, respectively.
* A mutator method called *setRadius* that has one double parameter. If the value of the parameter is larger than 0.0, store its value in the *radius* data member. Otherwise, do not modify the object.
* An accessor (get) method for each data member that returns its value. Your three methods should be called *getX(), getY(), and getRadius().*
* A method to return the area of the circle (pi times radius squared). You can use 3.14 for pi.

Important: methods that are not intended to modify the data members of an objects must be constant methods (see Programming Tip on page 35).

class Circle : public CircleInterface  
 {  
 public:  
    Circle();  
    void setRadius(double newRadius);  
    void setPosition(double newX, double newY);  
    double getX() const;  
    double getY() const;  
    double getRadius() const;  
     double getArea() const;  
 private:  
    double xPos;  
    double yPos;  
    double radius;  
 };

3. Write implementations for all methods in question 2.

   Circle::Circle()  
   {  
       xPos = 0.0;  
       yPos = 0.0;  
       radius = 1.0;  
   }  
  
   void Circle::setPosition( double newX, double newY)  
   {  
       xPos = newX;  
       yPos = newY;  
   }  
  
   void Circle::setRadius(double newRadius)  
   {  
       if (newRadius > 0.0)  
       {  
           radius = newRadius;  
       }  
       // else leave this circle unchanged  
   }  
  
   double Circle::getX() const  
   {  
       return xPos;  
   }  
  
   double Circle::getY() const  
   {  
       return yPos;  
   }  
  
   double Circle::getRadius() const  
   {  
       return radius;  
   }  
  
   double Circle::getArea() const  
   {  
       return 3.14 \* radius \* radius;  
   }