Queensland University of Technology (QUT)
IFN556 Practical Exercise Questions – Using Classes and Objects

1. Create a program named **PaintingDemo** that instantiates an array of eight Room objects and demonstrates the Room methods. The Room constructor requires parameters for length, width and height fields; use a variety of values when constructing the objects. The Room class also contains a field for wall area of the Room and number of gallons of paint needed to paint the room. Both of these values are computed by calling private methods. Include read-only properties to get a Room's value. A room is assumed to have four walls, you do not need to allow for windows and doors, and you do not need to allow for painting the ceiling. A room requires one gallon of paint for every 350 square feet (plus an extra gallon for any square feet greater than 350). In other words, a 12 x 10 room with 9-foot ceilings has 396 square feet of wall space, and so requires two gallons of paint.

2. Create an application class named LetterDemo that instantiates objects of two classes named Letter and CertifiedLetter and that demonstrates all their methods. The classes are used by a company to keep track of letters they mail to clients. The Letter class includes auto-implemented properties for the name of the recipient and the date mailed. Also, include a ToString() method that overrides the Objectclass's ToString() method and returns a string that contains the name of the class (using GetType()) and the Letter's data field values. Create a child class named CertifiedLetter that includes an auto-implemented property that holds a tracking number for the letter.

3. Create an application class named PhotoDemo that demonstrates the methods of three related classes for a company that develops photographs. Create a class named Photo that includes fields for width and height in inches and properties for each field. Include a protected price field, and set it to $3.99 for an 8-inch by 10-inch photo, $5.99 for a 10-inch by 12-inch photo, and $9.99 for any other size (because custom cutting is required). The price field requires a get accessor but no set accessor. Also include a ToString() method that returns a string constructed from the return value of the object's GetType() method and the values of the fields. Derive two subclasses—MattedPhoto and FramedPhoto. The MattedPhoto class includes a string field to hold a color, and the FramedPhoto class includes two string fields

that hold the frame's material (such as silver) and style (such as modern). The price for a MattedPhotoincreases by $10 over its base cost, and the price for a FramedPhoto increases by $25 over its base cost. Each subclass should include a ToString() method that overrides the parent class version.

4. Create a program named **RelativesList** that declares an array of at least 12 relative objects and prompts the users to enter data about them. The Relative class includes auto-implemented properties for the Relative's name, relationship to you (for example, aunt), and three integers that together represent the Relative's birthday – month, day, and year. Display the Relative objects in alphabetical order by first name.
