The University of Melbourne

School of Computing and Information Systems

COMP90041 Programming and Software Development

Lecturers: Prof Udaya Parampalli, Dr Thuan Pham

Semester 1, 2021, Week 8

Workshop Instructions

Inheritance & Polymorphism

1. Consider a graphics system that has classes for various figures – say, rectangles, boxes, triangles, circles, and so on. For example, a rectangle might have data members' height, width, and center point, while a box and circle might have only a center point and an edge length or radius, respectively. In a well-designed system, these would be derived from a common class, **Figure**. You are to implement such a system. The class **Figure** is the base class. You should add only **Rectangle** and **Triangle** classes derived from **Figure**. Each class has stubs for methods **erase** and **draw**. Each of these methods outputs a message telling the name of the class and what method has been called. Because these are just stubs, they do nothing more than output this message. The method **center** calls the erase and draw methods to erase and redraw the figure at the center. Because you have only stubs for erase and draw, center will not do any "centering" but will call the methods erase and draw, which will allow you to see which versions of draw and center it calls. Also, add an output message in the method center that announces that center is being called. The methods should take no arguments. Also define a demonstration program for your classes. Following is a sample output from a demonstration program.

```
Creating a figure with no parameters.
Call to Figure's draw method.
Call to Figure's erase method.
Calling Figure's center method.
Call to Figure's erase method.
Call to Figure's draw method.
Creating a figure with no parameters.
Creating Triangle Class with no parameters.
Calling Triangle's draw method.
Calling Triangle's erase method.
```

```
Calling Figure's center method.
Calling Triangle's erase method.
Calling Triangle's draw method.
Creating a figure with no parameters.
Creating Rectangle Class with no parameters.
Calling Rectangle's draw method.
Calling Rectangle's erase method.
Calling Figure's center method.
Calling Rectangle's erase method.
Calling Rectangle's draw method.
```

2. Draw a UML class diagram for the three classes **Figure**, **Triangle**, **Rectangle** described in question 1.

3. Define a class named **MultiItemSale** that represents a sale of multiple items of type **Sale**. The class MultiItemSale will have an instance variable whose type is Sale[], which will be used as a partially filled array. There will also be another instance variable of type int that keeps track of how much of this array is currently used. The exact details on methods and other instance variables, if any, are up to you. Use the **MutltiItemSale** class in a program that obtains information for items of type **Sale** and of type **DiscountSale (**sample implementations of these two classes are provided**)** and that computes the total bill for the list of items sold. Following is a sample input-output when the program is executed.

```
Enter S for a sale, D for discounted sale or Q to quit
S
Enter name of item:
Teddy bear T-shirt
Enter item's price:
40.0
Enter S for a sale, D for discounted sale or Q to quit
S
Enter name of item:
Nike hat
Enter item's price:
80.0
Enter S for a sale, D for discounted sale or Q to quit
D
Enter name of item:
Adidas backpack
Enter item's price:
45.0
```

```
Enter percentage discount as a double.
For example 6.5 for 6.5%
10.0
Enter S for a sale, D for discounted sale or Q to quit
Q
Total bill: $160.5
```