6.2 P

1. Abstraction

   Definition: Process of hiding complex implementation details and showing only necessary features of an object.

   Example: In Shape Drawer, Shape class is used as an abstract class for the other shape classes such as rectangle and circle class. There are private fields in the abstract class Shape that applies to rectangle and circle class such as a colour field.

2. Encapsulation

   Definition: Bundling fields and methods in a class, with access restrictions to protect private fields.

   Example: When creating a rectangle class, there would be private fields such as width and height and then there would be methods such as a constructor to initialize the field values.

3. Inheritance

   Definition: Inheritance allows the subclass to inherit fields and methods used in the parent class. This promotes reusability in coding.

   Example: Animal class (parent class) has a method such as speak() . The subclass Dog can inherit this method speak() and add its own implementation such that the output would produce "Dog barks".

4. Polymorphism

   Definition: The ability of different classes to be treated as instances of the same class through inheritance or interfaces.

   Example: In the Shape Drawer program, newShape is declared as the base class Shape, but it can refer to any subclass such as MyCircles, MyLine or MyRectangle based on the enumeration kindtoAdd. Having newShape allows to handle different shape types using the same reference.

Concept Drawing: