03 – Functions and Methods - Exercises

OBJECT-ORIENTED PROGRAMMING

EXERCISE 1

- Complete the following C++ code for the class Car by giving implementations for the method bodies. Draw a class diagram for this class.

```
class Car {
   string theMake;
   string theModel;
   int theCapacity;
public:
   Car(string aMake, string aModel, int aCapacity) { ... }
   string getMake() { ... }
   string getModel() { ... }
   int getCapacity() { ... }
}
```

- Now develop an application that creates an instance of the Car class and displays its details.

EXERCISE 2

- Complete the following C++ code for the class Student by giving implementations for the method bodies. Draw a class diagram for this class.

```
class Student {
   string theName;
   string theAddress;
   string theMatriculationNumber;
public:
   Student(string aName, string anAddress, string
     aMatriculationNumber) { ... }
   string getName() { ... }
   string getAddress() { ... }
   string getMatriculationNumber() { ... }
}
```

- Now develop an application that creates an instance of the Student class and displays its details.

EXERCISE 3

- Complete the following C++ code for the class House by giving implementations for the method bodies. Draw a class diagram for this class.

```
class House {
   string theAddress;
   int theNumberOfRooms;
public:
   House(string anAddress, int aNumberOfRooms) { ... }
   string getAddress() { ... }
   int getNumberOfRooms() { ... }
   void extend(int aNumberOfRooms) { ... } //Add new rooms
}
```

- Now develop an application that creates an instance of the House class, adds some rooms to it, and then displays its details.

EXERCISE 4

- Complete the following C++ code for the class Point by giving implementations for the method bodies.

```
class Point {
  double theX;
  double theY;
public:
  Point(double anX, double anY) { ... }
  double getX() { ... }
  double getY() { ... }
  void moveBy(double anX, double anY) { ... }
}
```

- Now develop an application that creates a Point object, and then moves it by some amount and display its new position.

EXERCISE 5

- Using the Point class, complete the following C++ code for the class Line by giving implementations for the method bodies.

```
class Line {
   Point theStart;
   Point theEnd;
public:
   Line(Point aStart, Point anEnd) { ... }
   bool isHorizontal() { ... }
   bool isVertical() { ... }
   void moveBy(double anX, double anY) { ... }
   void display() { ... }
}
```

- Now develop an application that creates a Line object through two Point objects , and then moves it by some amount and display its new position.

EXERCISE 6

- Using the Point class, a rectangle may be represented by two Point values representing, respectively, the upper left vertex and the lower right vertex. Complete the following C++ code for the class Line by giving implementations for the method bodies.

```
class Rectangle {
   Point theUpperLeft;
   Point theLowerRight;
public:
   Rectangle(Point anUpperLeft, Point aLowerRight) { ... }
   double getArea() { ... }
   double getHeight() { ... }
   double getWidth() { ... }
   void moveBy(double anX, double anY) { ... }
   bool isPointInRectangle(Point aPoint) { ... }
}
```

- Now develop an application that creates a Rectangle object, and then test the behaviours of its operation.