Object-Oriented Programming with C++
ENSIA 2024-2025
Tutorial 9 (Polymorphism)

Exercise 1

Let's consider the UML class design shown in Figure 1 below:

Task 1

Create a simple `Shape` hierarchy:

- A base class called `Shape`
- Derived classes called `Circle`, `Rectangle`, and `Triangle`

Supported tasks on each shape:

1. **Input the shape attributes' value:**
    - **Rectangle:** Coordinates of the top-left and bottom-right points
    - **Triangle:** Coordinates of the three peaks of the triangle
    - **Circle:** Coordinate of the center point and the radius
2. **Compute the area of the shape**
3. **Display the attributes' value of the shape**
4. **Move the shape given x and y offset**
5. **Getters and setters**

Task 2

In the main program:

1. Create an array (or vector) of pointers to `Shape` objects created dynamically (use one array of `Shape` pointers)
2. Perform the following tasks:
    (a) Allow the user to enter the number of shapes n to create
    (b) Loop n times, each time:

- Ask the user which shape to create
- Create the shape as chosen
- Let the user input its attributes

(c) Display the attributes and area of the shapes

(d) Move all the shapes randomly

(e) Display the attributes again

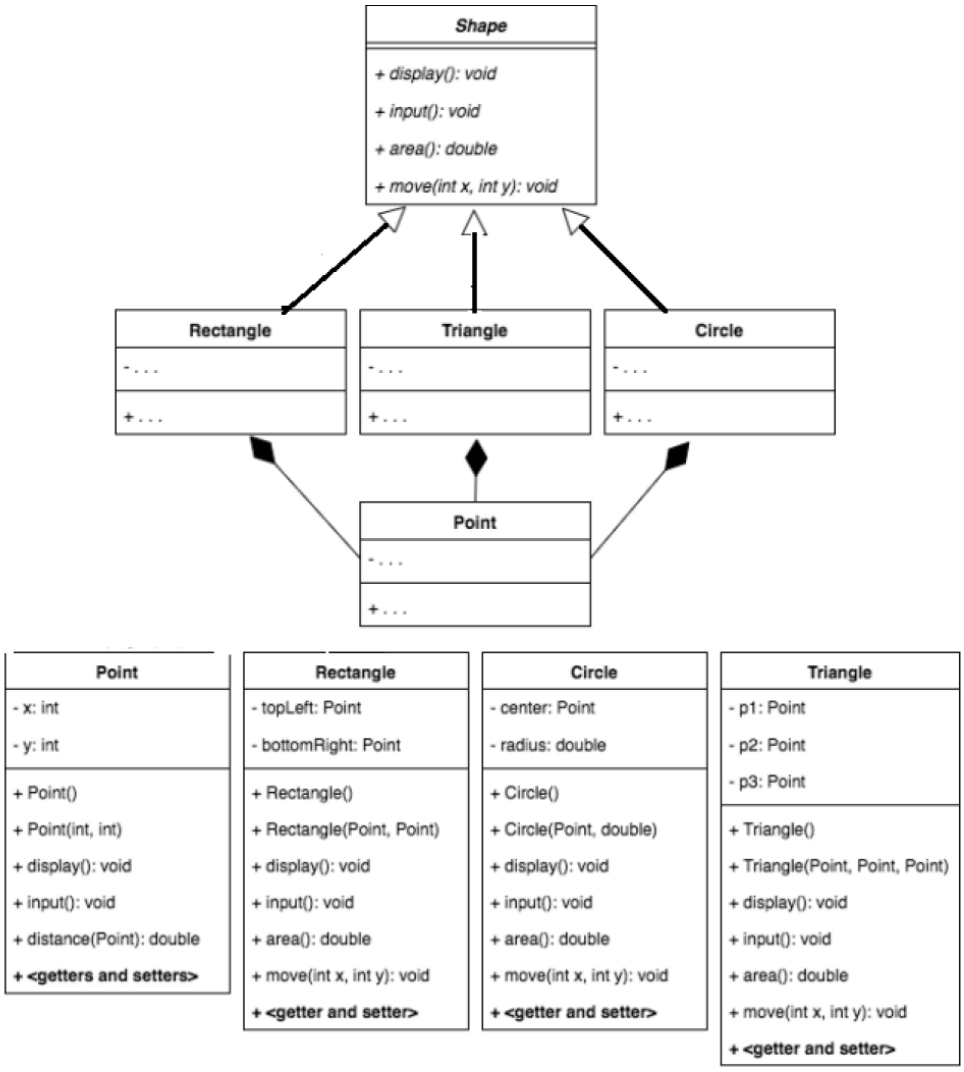

Figure 1: UML Class Diagram

Exercise 2

A Phone Company provides phone services for its customers.

Tasks

1. Create an abstract class named `PhoneCall`:
   - `String` field for the phone number
   - `double` field for the price of the call
   - A constructor requiring a phone number parameter and setting the price to 0.0
   - A setter function for the price
   - Three methods:
     (a) Return the phone number
     (b) Return the price of the call
     (c) Display information about the call
   - Keep track of the date of the call using a `Date` class
2. Create two child classes of `PhoneCall`:
   - `IncomingPhoneCall:`
     - The Constructor passes its phone number to the parent constructor
     - Sets the price of the call to 2 DA
     - Display method shows phone number, price, and cost
   - `OutgoingPhoneCall:`
     - Additional field: time of the call in minutes
     - The Constructor requires a phone number and time
     - Price is 10 DA per minute
     - Display method shows details: phone number, price per minute, minutes, and total cost
3. All classes should include a function that calculates the cost of each call.
4. Write a driver application:
   - Instantiate and display both `IncomingPhoneCall` and `OutgoingPhoneCall` objects
   - Use an array of base class pointers
   - Use a loop to display the data
   - Display the total price of all calls