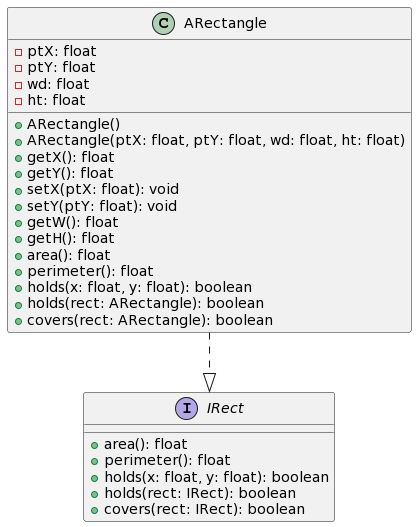
**Q1**



**Q4**



**Q9**

Design Enhancement and Progression

• Part (3) - Introduction of Interface `IRect`

We implemented an interface named `IRect` to outline a standard set of functionalities for rectangles. This interface encapsulates fundamental operations related to rectangles, such as computing area and perimeter. Through the utilization of interfaces, we achieve abstraction, enabling the interchangeability of various rectangle implementations.

• Part (5) - Introduction of Abstract Class `AShape` and Concrete Subclasses

We introduced an abstract class named `AShape`, which acts as a foundational structure for diverse shapes like rectangles, squares, and circles. Concrete subclasses inherit common methods from `AShape`, fostering code reusability and providing a uniform interface for interacting with shapes. This design fosters scalability and abstraction.

• Part (7) - Client Interaction through `Shape` Interface

Client code exclusively interacts with shapes via the `Shape` interface, concealing the implementation specifics of individual shape classes. This approach encapsulates shape functionalities, separates client code from concrete implementations, and augments flexibility. Clients are only required to be familiar with the `Shape` interface, streamlining usage and encouraging clean code practices.