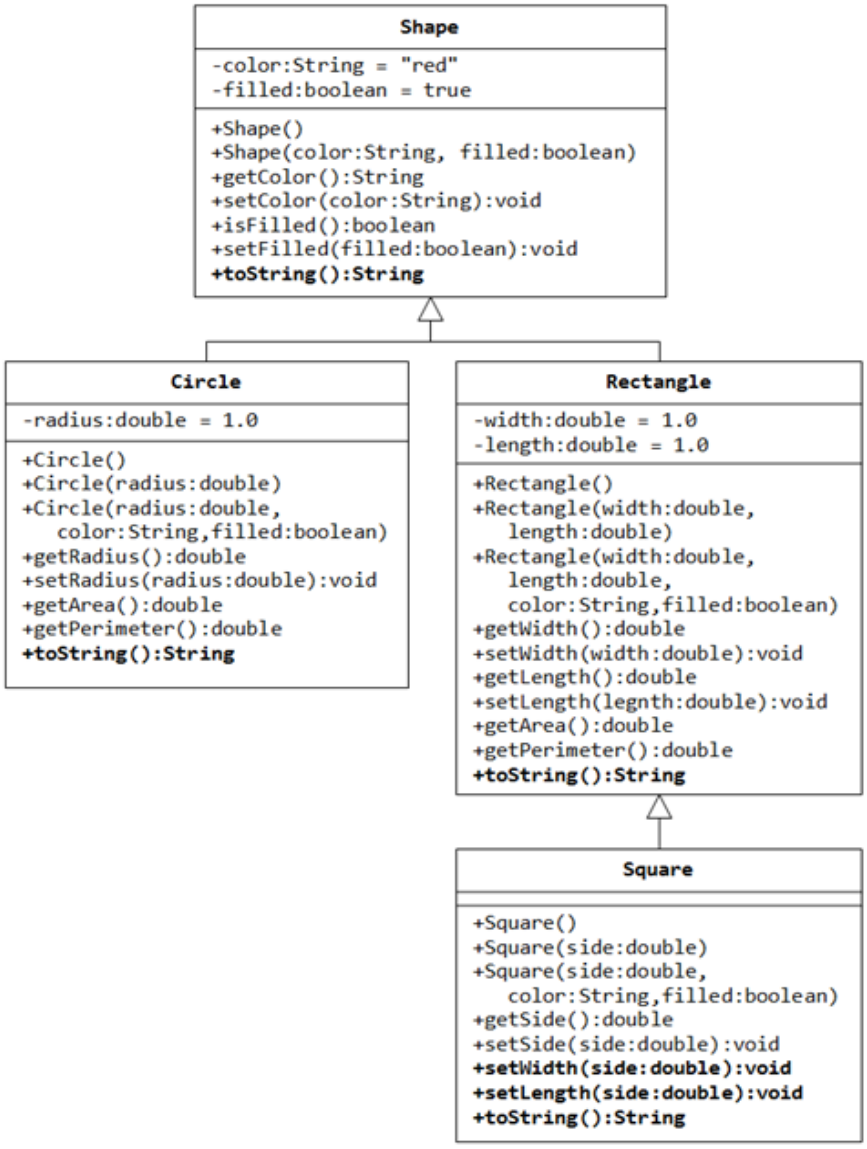
Lab 03

Kế thừa, final

Câu 1. Cho các đối tượng sau gồm Hoa Quả, Quả Cam, Quả Táo, Cam Cao Phong, Cam Sành.

- Dựa trên cách hiểu của mình về thừa kế, hãy viết chương trình mô tả quan hệ các đối tượng trên.
- Với từng đối tượng, hãy bổ sung các thuộc tính và phương thức bạn cho là cần thiết (VD: giá bán/cân, nguồn gốc xuất xứ, ngày nhập, số lượng, v.v.)
- Viết hàm main khởi tạo các đối tượng trên

Câu 2. Cho biểu đồ lớp như sau:



- Trong biểu đồ trên có tất cả bao nhiêu quan hệ thừa kế (is-a)? Tại sao lớp Circle có thể thừa kế lớp Shape mà không phải lớp Rectangle?
- Hiện thực các lớp trong sơ đồ trên. Viết hàm main để kiểm tra chương trình.
- Định nghĩa thêm số PI trong lớp Circle ở chương trình vừa tạo; sau đó sử dụng giá trị PI để tính chu vi và diện tích hình tròn. Biến PI này có nên để final không?

Đa hình, instanceof, abstract

Câu 3. Giả sử bạn cần viết một ứng dụng đồ họa với những thông tin thiết kế ban đầu như sau

Diagram là một sơ đồ hệ thống

- Diagram: là lớp đại diện cho sơ đồ đang được vẽ
- Layer: một đối tượng thuộc lớp Diagram có một hoặc nhiều đối tượng thuộc lớp Layer
- Shape: là lớp đại diện cho các hình vẽ khác nhau (Rectangle, Square, Triangle, Circle). Mỗi đối tượng Layer chứa nhiều đối tượng lớp Shape
- Các hình vẽ có thuộc tính để xác định vị trí và kích thước
- Các hình vẽ có thể được tô màu và có thể được di chuyển

Hãy:

- Định nghĩa các lớp trên (Diagram, Layer, Shape, Rectangle, Square, Triangle, Circle, và các lớp khác nếu cần thiết)
- Bổ sung phương thức cho lớp Layer để xóa tất cả các đối tượng thuộc lớp Triangle trong lớp
- Bổ sung phương thức cho lớp Diagram để xóa tất cả các đối tượng thuộc lớp Circle trong Diagram
- Viết phương thức main để kiểm thử các phương thức trên

****

Yêu cầu:

- Tất cả mọi chương trình phải có đủ comment cho từng class, từng hàm
- Các thuộc tính cần có đủ setter, getter tương ứng
- Sử dụng Junit để viết các test cases
- Test chương trình bằng Junit
- Deadline: Trước chủ nhật ngày 22/07/2018

Gợi ý:

- Các bạn nên viết các test case trước rồi mới viết test sau. Test cases nên phủ được càng nhiều càng tốt các trường hợp có thể xảy ra