In-class exercises (lecture 3)

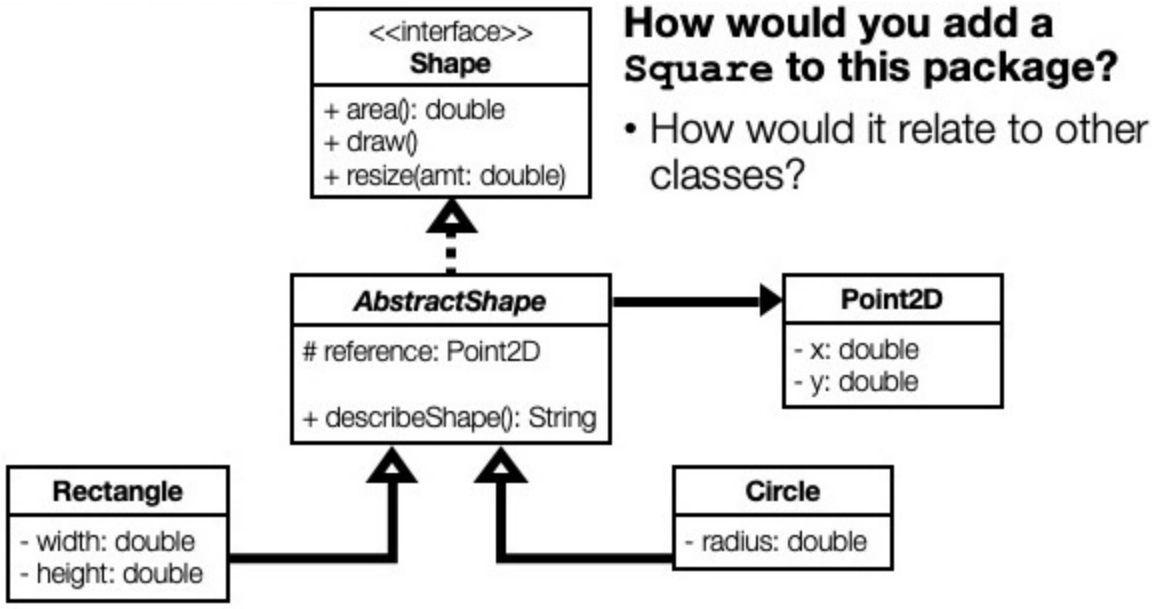
At the start of every lecture:

1. Pull the latest code from the lecture-code repo
2. Open the Evening_lectures folder
3. Copy this week's folder somewhere else so you can edit the code without causing GitHub conflicts
4. Open the code:
    1. Find the build.gradle file in the folder called LectureX
    2. Double click build.gradle to open the project in IntelliJ.

Exercise 1: Add a Square to the shapes package

You will be in breakout rooms for 10 minutes.

You're tasked with adding a Square class to the shapes package (see below). How / where would you add it? What forms of inheritance would you use, if any? There are several possibilities!
You're not expected to write code, just discuss.



When you're back in the main room, there will be a poll to see what the most popular approach is.

Exercise 2: Design activity

In breakout rooms until ~8:50

Your task: As a group, come up with an outline for an object-oriented design for the scenario below. This includes:

- The classes you will need (interface, abstract classes, concrete classes, enums).
- A rough idea of the fields you will need in each class.
- A rough idea of the public methods for each class.

You are not expected to write any code, although you can if it helps you think through the problem.

The scenario

A bank offers it's customers several payment methods that they can use for purchases. They have tasked you with building a system to help them track and manage the various payment methods their customers are using.

There are two main types of payment method: credit and debit.

The credit payment methods the bank offers are:

- a credit card, which charges interest on the balance. The balance is the total of principal + interest. The interest rate can be changed.
- a line of credit, which does not charge interest on the balance.

The debit payment methods the bank offers are:

- an ATM card, which requires a signature for charges above a given threshold
- a check, which requires a signature for all charges.

All payment methods have a balance (a double) and a method, **processCharge**, which takes a Charge and adjusts the balance. A Charge consists of: an amount (a double) and a boolean flag that indicates whether or not a signature was provided when the charge was created (e.g. a signed credit card receipt).

All credit payment methods have a credit limit (a double), which can be changed. When a credit account is charged, the balance increases. A Charge can only be processed if the amount will not cause the balance to exceed the credit limit. All credit payment methods need functionality to make a payment (a double amount) that will decrease the balance.

For all debit payment methods, the balance decreases when a charge is processed. A Charge cannot be processed if it would cause a negative balance.