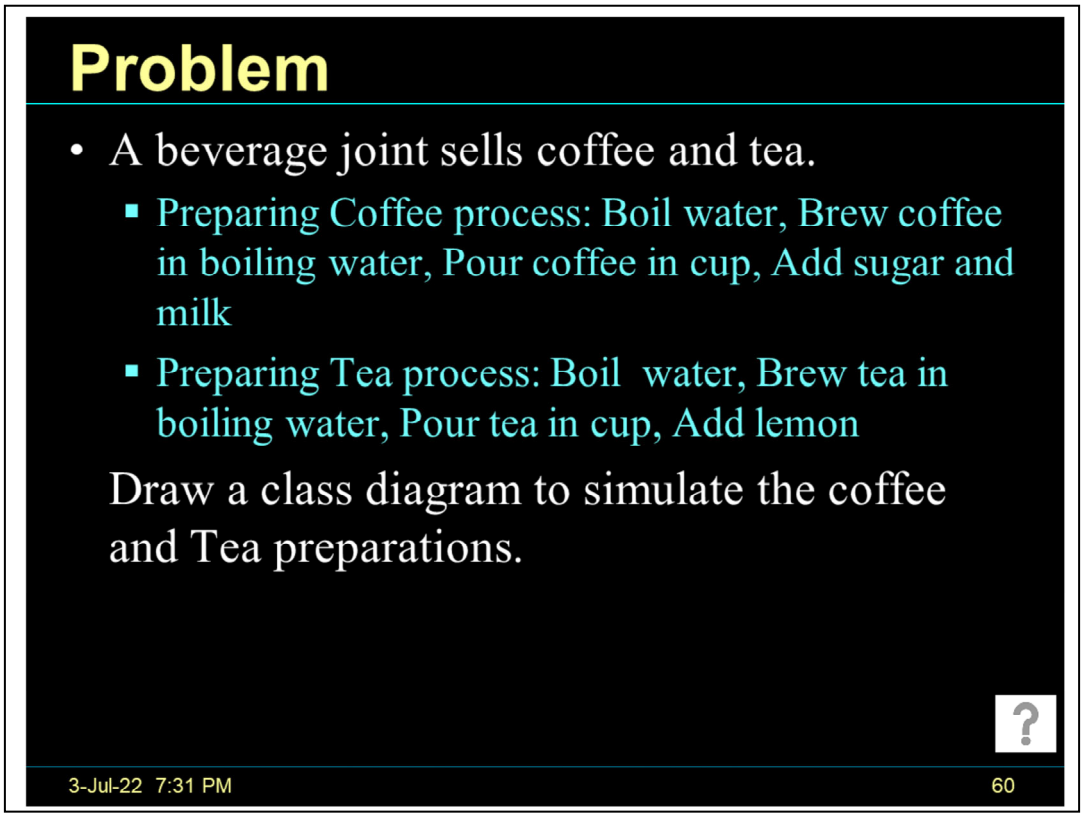
Template DP



Principle

A specific logic should be present once and once only in an application.

Result of this Principle

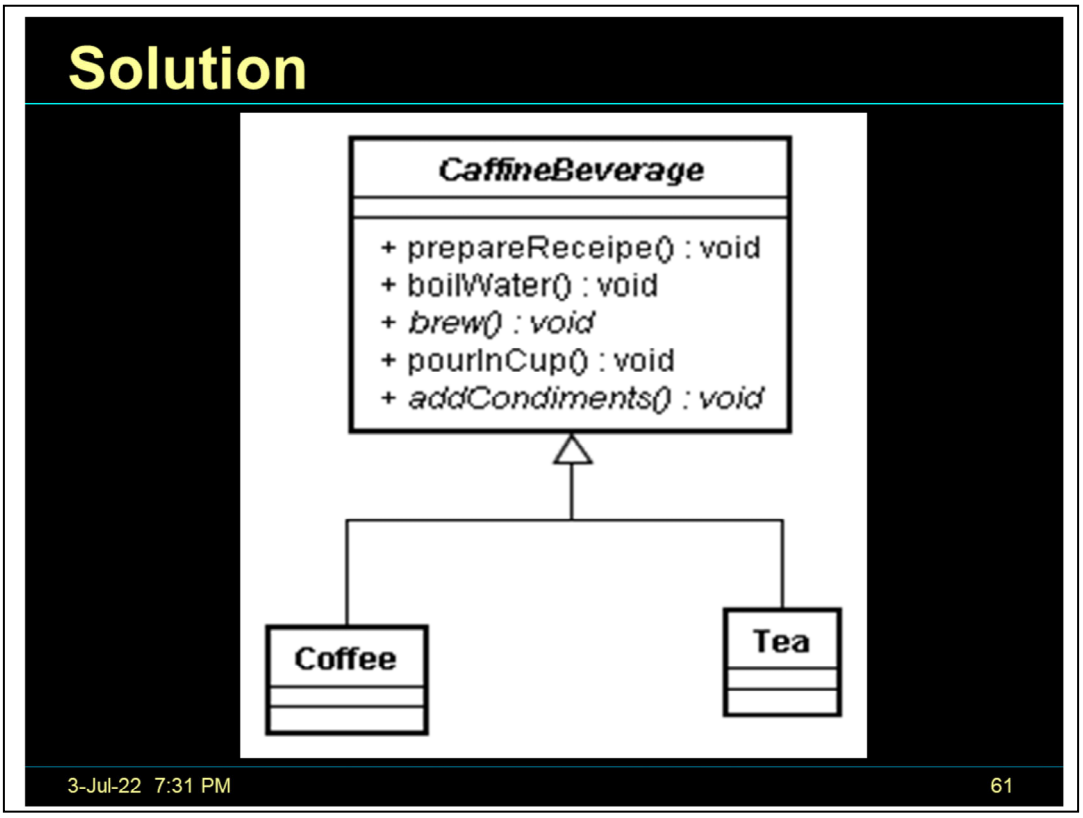
Code modifications are simple

Code size reduced.

Easy understanding.

Probably the most important principle.

This principle is called DRY (Don't Repeat Yourself) or Once and Only Once principle.



The code of prepareReceipe is

```
void final prepareReceipe() {
  boilWater();
  brew();
  pourInCup();
  addCondiments();
}
```

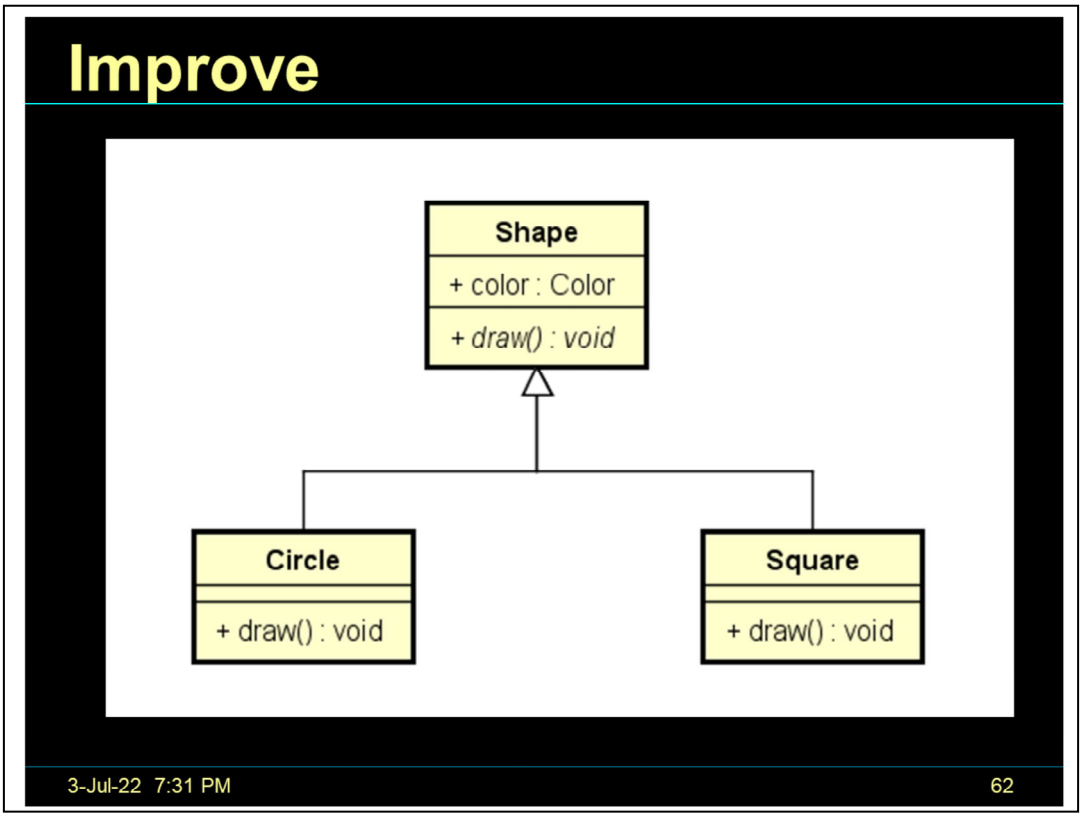
It defines the skeleton of an algorithm in a method, deferring some steps to subclasses.

> It lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

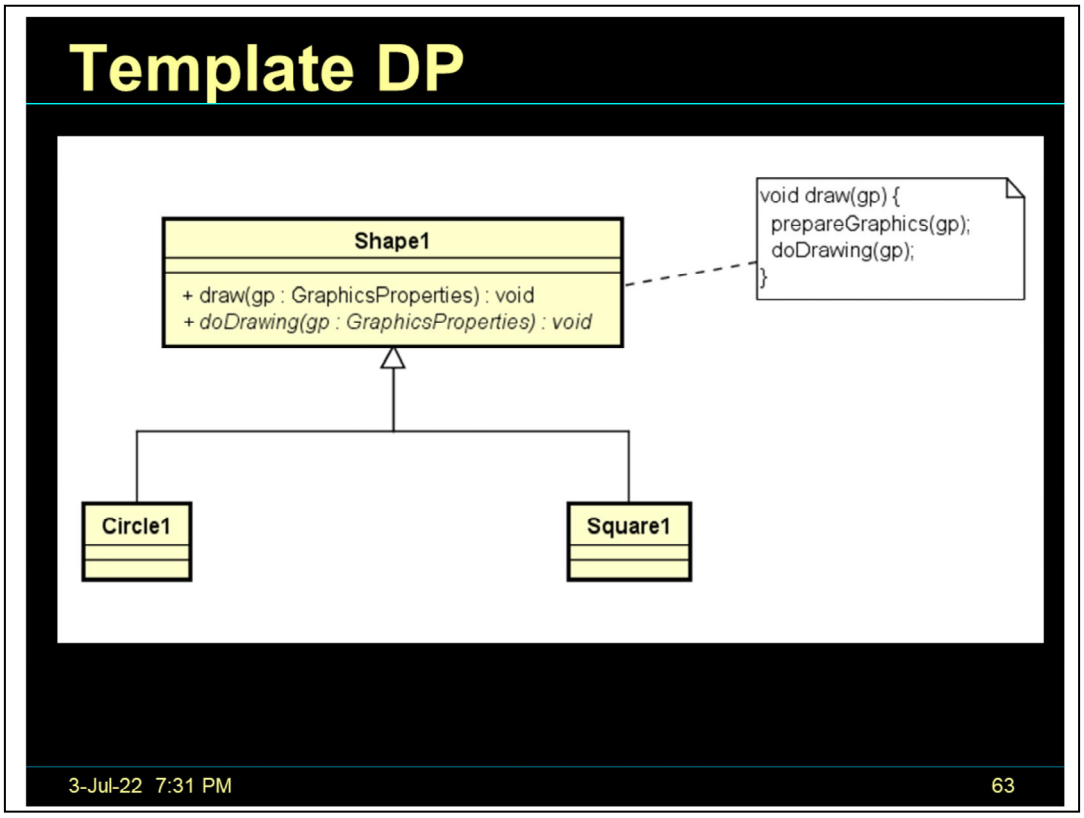
Maximized code reuse.

> Otherwise code would have been duplicated.

Knowledge about the algorithm is concentrated in the base class. Subclasses can provide custom implementations.



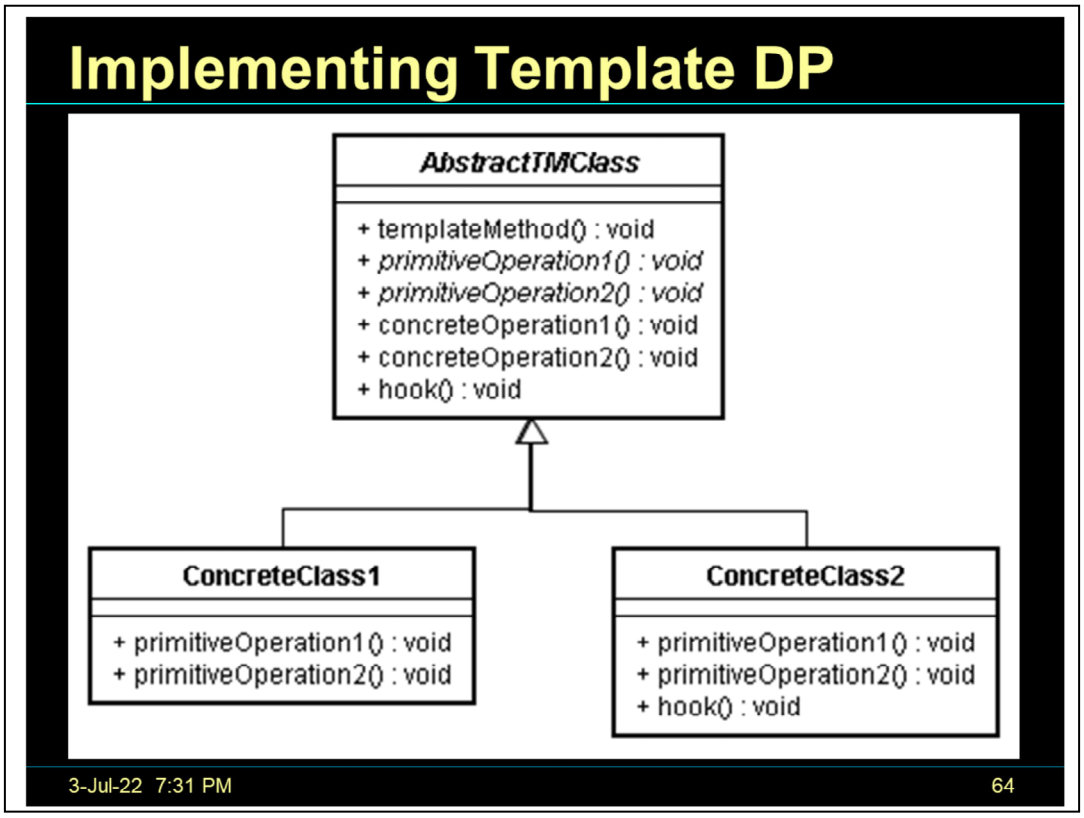
The draw function of Circle and Square will need to use Color via some getter function. This breaks encapsulation principle for Shape.



GraphicProperties can have color, line thickness, etc.

Based on the values in graphicProperties, different instance varialbes in Shape1 can be set.

doDrawing function will exist for each derived class now



Concrete methods carry out some basic operation. They are final.

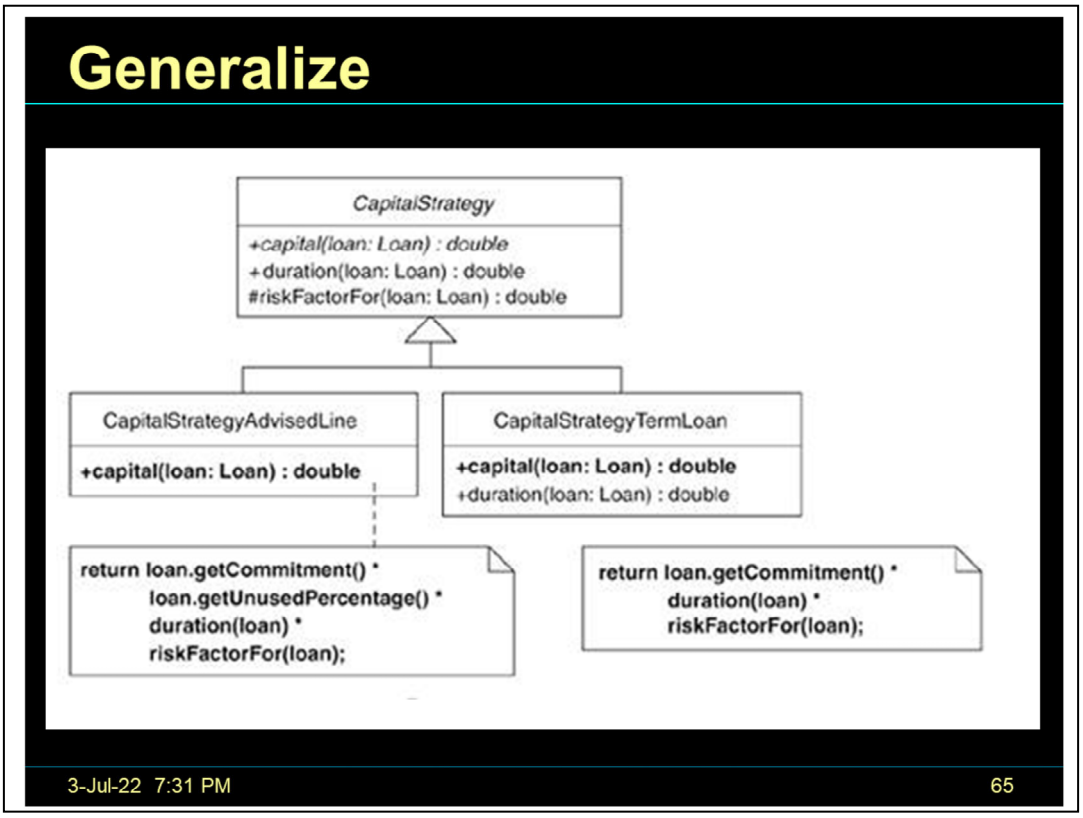
Primitive methods whose implementation must be provided in the subclasses. They are abstract.

Hooks are methods that is declared in the base class. They are often empty. Sometimes they have a default implementation. The subclass can override it, if required.

This DP is used

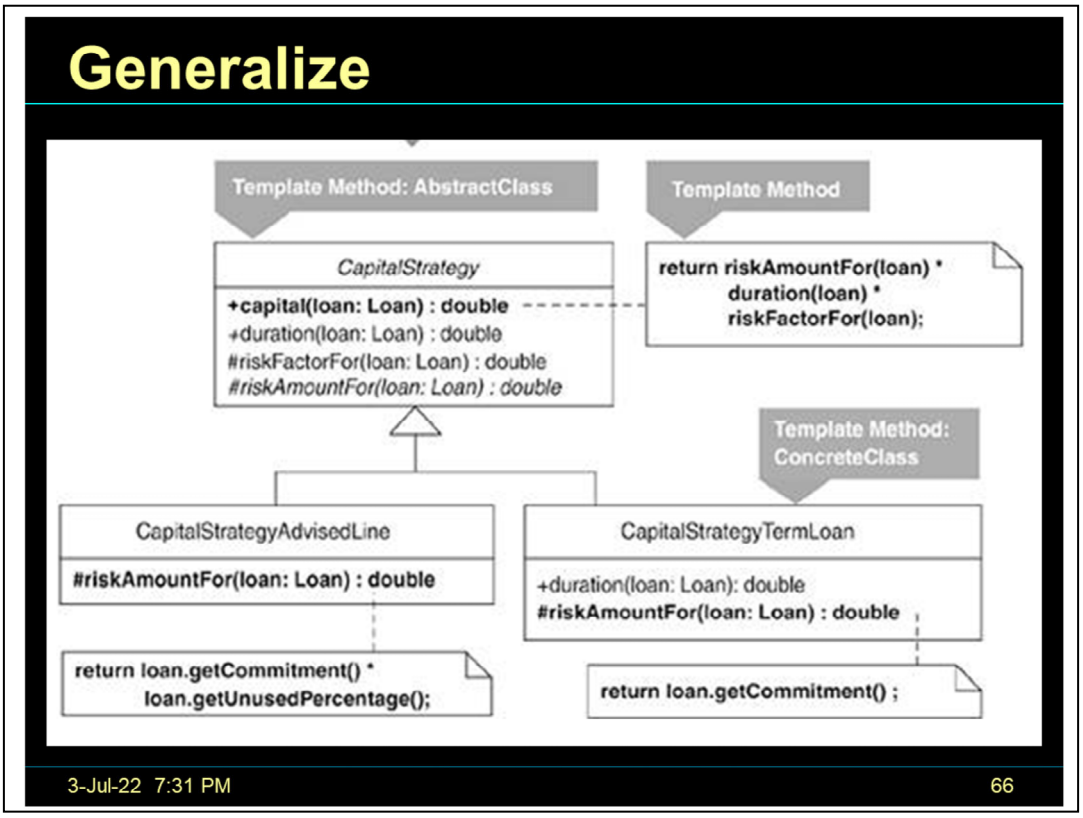
- To implement the invariant parts of an algorithm once and leave it up to subclasses to implement the behavior that can vary.
- When common behavior among subclasses should be factored and localized in a common class to avoid duplication.

This DP is used often in frameworks / libraries.



Two methods in subclasses perform similar steps in the same order, yet the steps are different.

Generalize the methods by extracting their steps into methods with identical signatures, then pull up the generalized methods to form a Template Method.



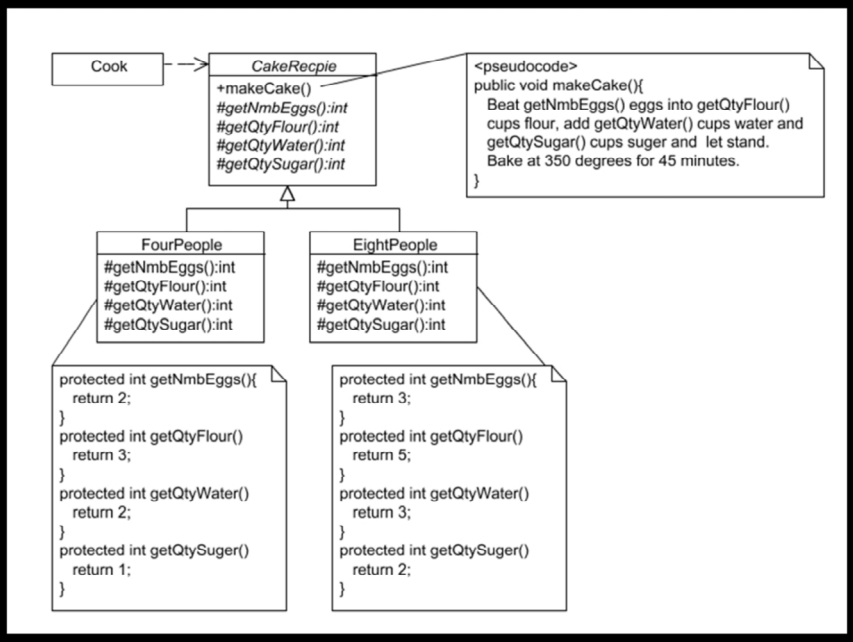
+Removes duplicated code in subclasses by moving invariant behavior to a superclass.

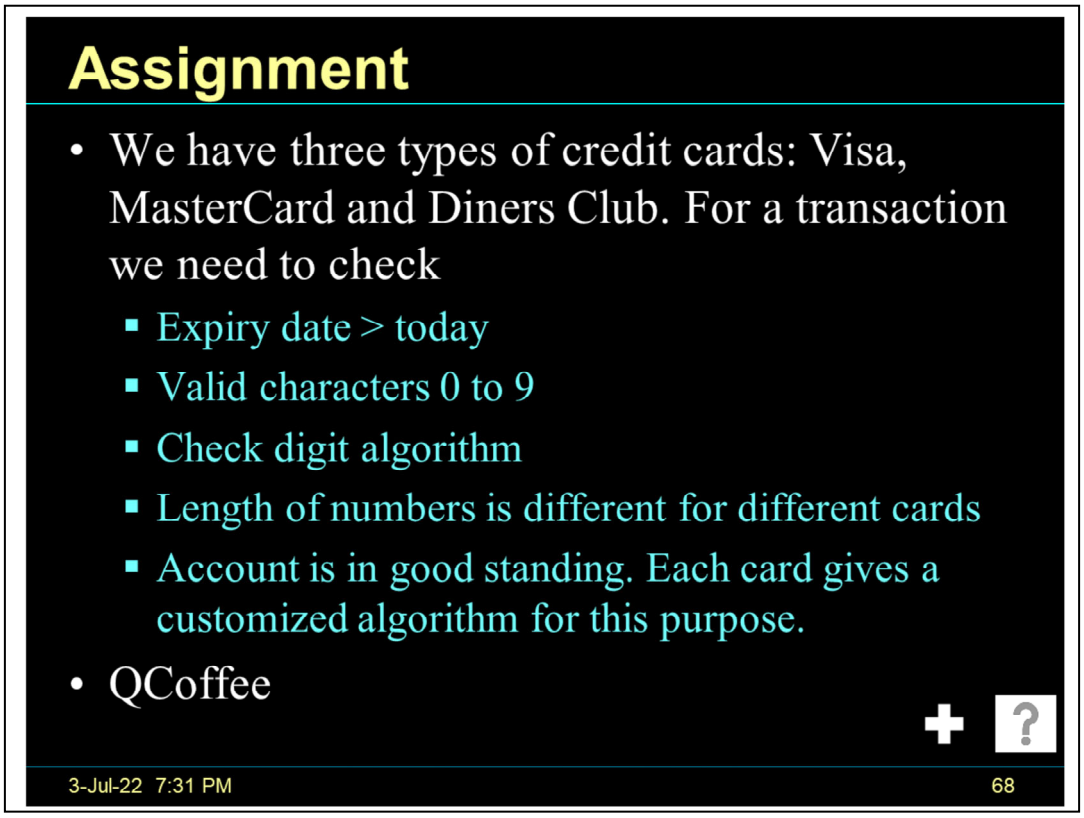
+Simplifies and effectively communicates the steps of a general algorithm.

+Allows subclasses to easily customize an algorithm.

–Complicates a design when subclasses must implement many methods to flesh out the algorithm.

Real Life Analog





Instead use the Tax computation problem in the exercise that has lot of duplication.

OOAD slides: Open Closed Principle, Guidelines (keep things that vary separately),

Inheritance vs. Delegation, Airline Reservation System, Same Person – Multiple roles, A person can change roles Now, Stack is not ArrayList, What is the output, Bat cannot be a bird, Liskov Substitution Principle,

Does Subclass make sense, Avoid Deep inheritance, All Inherited features should make sense in subclasses, Reorganize (Ellipse, circle)