**Chain Constructors**

**Motivation**

Multiple constructors in a class can contain duplicate code each time a variable is initialised. This makes code harder to read as there are a lot of repeated statements, making it tedious for future developers to work with. This issue may also cause functional problems as multiple constructors would each need to be updated when a new variable is added, which is a problem if there are many constructors. The solution is to implement a chain of constructors, otherwise known as a series of constructors that call each other. This removes duplicate code and ensures all constructors are up to date.

**Methods**

1. Identify two constructors that contain duplicate code.
2. Determine which constructor can call the other.
3. Remove the duplicate code from the second constructor and call the first constructor.

**Sample Code to refactor**

A bank wants to create multiple types of loans to distribute to customers. They come up with the following classes for each type of loan. However, there is a lot of duplicated code between each class, which can be reduced through the Chain Constructors refactoring.

public Loan(float notional, float outstanding, int rating, Date expiry) {

this.strategy = new TermROC();

this.notional = notional;

this.outstanding = outstanding;

this.rating = rating;

this.expiry = expiry;

}

public Loan(float notional, float outstanding, int rating, Date expiry, Date maturity) {

this.strategy = new RevolvingTermROC();

this.notional = notional;

this.outstanding = outstanding;

this.rating = rating;

this.expiry = expiry;

this.maturity = maturity;

}

public Loan(CapitalStrategy strategy, float notional, float outstanding, int rating,

Date expiry, Date maturity) {

this.strategy = strategy;

this.notional = notional;

this.outstanding = outstanding;

this.rating = rating;

this.expiry = expiry;

this.maturity = maturity;

}