BOND CALCULATOR USER MANUAL

Xolani Mazibuko (mazi76erx@gmail.com)

25 September, 2018

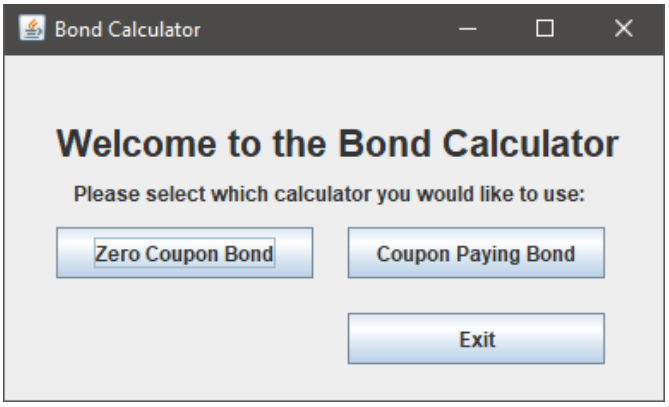
Introduction

A bond calculator which calculates the bond value at maturity using both the Zero Coupon Bond formula as well as the Coupon Paying bond formula shown in Appendix A. This implementation of this program is written using the Java. You will need a Java Runtime Environment (≥1.6) to run it.

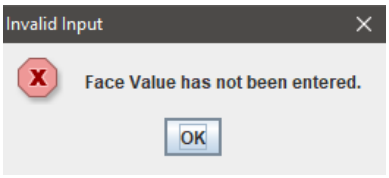
Contents

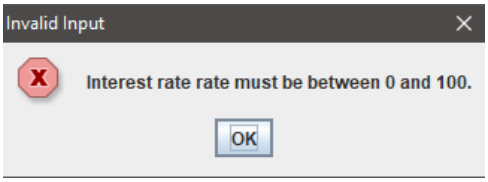
Usage Guide

1. Launch the Bond Calculator by double-clicking on its icon in Windows Explorer.
2. Choose the calculator you would like to use between Zero Coupon Bond and Coupon Paying bond by clicking on the corresponding button.

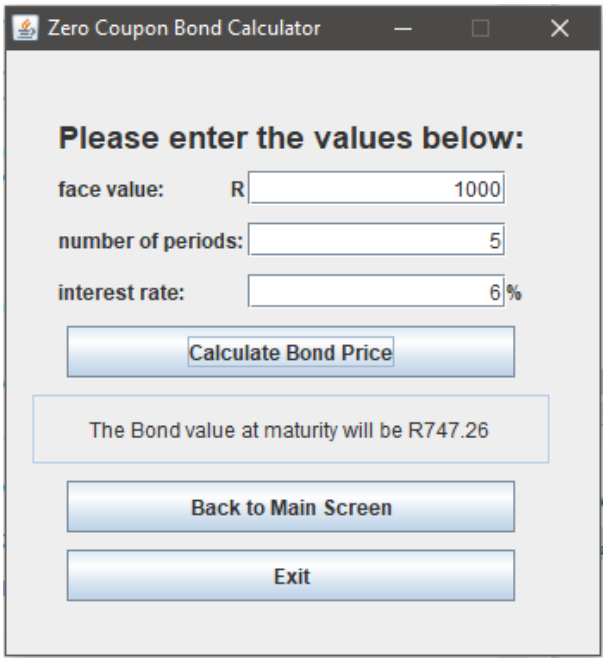


3. Make sure to enter values into each edit box otherwise the program will display an error message. All values must be positive numbers and the interest rate and coupon rates must be between 0 and 100.





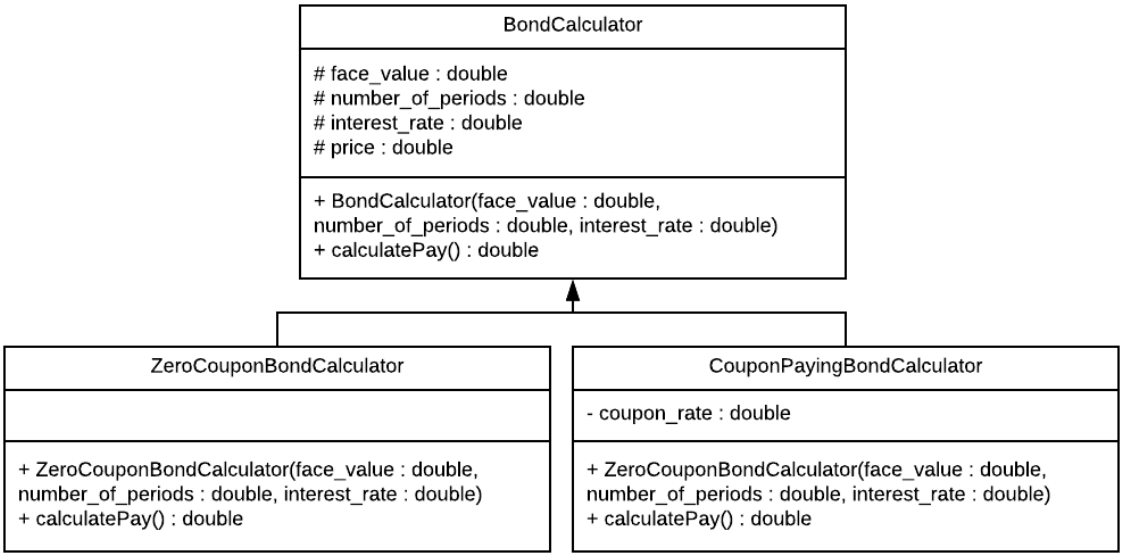
4. Once all the data has been entered click on the “Calculate Bond Price” button. The bond price will be displyed underneath this button.



5. The bond price can be re-calucated by editting the data in the editboxes and clicking on the “Calculate Bond Price” button.
6. Once are done you may click on the “Back to Main Screen” button to use the other calculator or “Exit” button to close the program.

Object Orientation Design Principles

UML of Bond Calculator



Design Decisions

Inheritance

ZeroBondCalculator and CouponPayingBondCalculator inherit from the base class BondCalculator. Inheritance is used here because of the both calculators use the same variables (face_value, number_of_periods, interest_rate), constructor and method (calculatePrice). CouponPayingCalculator has an additional variable (coupon_rate) and both derived classes have different implementations of the method calculatePrice.

Abstraction

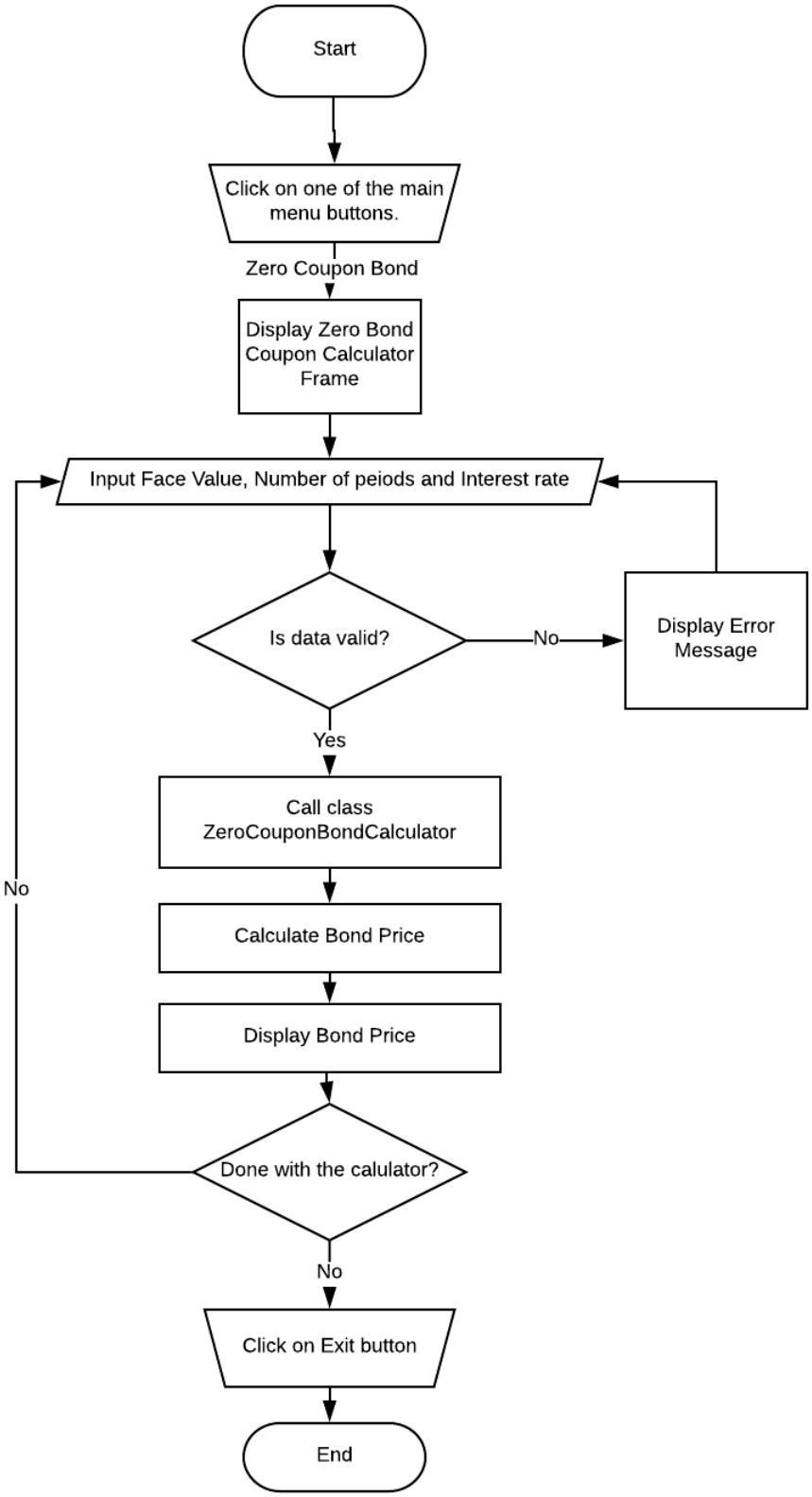
BondCalculator is a abstract class which means that every derived class of BondCalculator is must have the same variables, constructors and methods. Abstraction is used because the users of BondCalculator do not need to see the implementation of the class, they are only worried about the input and output.

Encapsulation

BondCalculator has protected variables (face_value, number_of_periods, interest_rate) and CouponPayingBondCalculator has private variable (coupon_rate). These variables are not accessible to the user who uses these class's as one would not want them to change the implementation of the class which would cause an error. This is an example of encapsulation.

Operational Flow of Code

Zero Coupon Bond Calculator



Appendix A

Zero Coupon Bond Formula

$$Bond\ Price = \frac{F}{(1+i)^n}$$

F – Face Value

i - Interest rate

n – number of periods

Coupon Paying Bond Formula

Coupon paying Bond Formula

$$Bond\ Price = C\left(\frac{1-(1+i)^{-n}}{i}\right) + \frac{F}{(1+i)^n}$$

C – Coupon Price (Coupon rate x FaceValue)

F – Face Value

i - Interest rate

n – number of periods