

# Part 9: Exceptions and Validation

## Exercise Instructions

---

The last part of the challenge then - let's get some exception handling in so we can control what happens when things go wrong in a bit more of a best practice kind of way.

We'll also get you to use a better way of formatting numbers too as a bonus!

Oh and also, we'll give you less directions now as you should hopefully have a good idea of how to work with the IDE at this point to get stuff done.

### Step 1: Update the Code

Let's do some tidying up first, then we'll do the exceptions stuff afterwards. So first, so minor tidyup:

- Ok, let's do some inlining - it's the right thing to do after all! So ensure you change the code so this is present instead of the non-inlined way:

```
printResult(new LoanCalculator(amount, years, interestRate).calculateRepaymentAmount());
```

*Nobody else is using the `LoanCalculator` so that's ok to do this, plus nobody else is using the result - so we're kind of double inlining it here.*

- Lose the formatting of the result by replacing it with this code (just remember to do the imports when the types turn red in IDEA - ok, use ALT+ENTER for this one to bring up the quick fix options!):

```
private static void printResult(double currentAmountPayable) {  
    NumberFormat formatter = new DecimalFormat("#0.00");  
    System.out.println("Total Amount Due: " + formatter.format(currentAmountPayable));  
}
```

Right, now we can turn our attention to the exception handling:

- Create a new class for the exception we'll throw, `LoanCalculationException`, create it as a regular class as we did when previously creating the other class.
- Ensure that `LoanCalculationException` subclasses `java.lang.Exception`.
- Add a constructor which takes a message. (Hint: you need to call the superconstructor too).
- In the constructor for `LoanCalculator`, before you assign the fields - first check that they're greater than 0, and if they're not throw the exception.

- Apply a `try / catch` around the code in `App.main()` now to be able to catch the exception, and just get the message from the exception and print that out if you get an error.

## Step 2: Run the Code

- No surprises here, run the code and ensure it works as expected (e.g. test by changing the input values to be negative and ensure the exception is thrown in any error cases) - don't forget to check that it works when the input is good too though, in the happy path!