

Part 9: Exceptions and Validation

Solution Code

So, a few points:

- The `NumberFormatter` : feels better doesn't it (rather than the hacky string manipulation! This just goes to show - the more you find out about the Java API, the less code you'll have to write (as you'll know about some library or function which does what you want it to do without you having to write it!) So - lesson there - keep looking at code and studying examples: this is a great way to accumulate that knowledge and experience superfast!
- Double inlining - maybe that's a bit overkill - but I do like things to be tidy (Note a disadvantage of this is that you can't 'see' the variable after it's being computed unless you put a breakpoint on the `return` from the method in which the inlined value is being calculated).

The main point though is that you created your own exception and used it. Also, because it was a *checked* exception (by extending `java.lang.Exception` and not `java.lang.RuntimeException`) you forced the programmer (yourself in this case!) to deal with the possibility of error, resulting in the `try` / `catch` block in the `App`'s `main` method.

Finally, I just want to say - it's absolutely amazing that you've come this far! Java is not an easy thing to quickly pick up like other languages might be and there's a lot of let's say conceptually-heavy stuff in there which you have to know, but after completing this challenge you should now feel like you've learned a ton and have a pretty good handle on introductory Java programming.

Thanks for staying with us and we hope you enjoyed this challenge part of the course!

Code Listings

App.java

```

package com.javaeasily.demos;

import java.text.DecimalFormat;
import java.text.NumberFormat;

public class App {
    public static void main(String[] args) {
        System.out.println("Loan Calculator Application".toUpperCase());
        System.out.println();

        int amount = 100;
        int years = 5;
        double interestRate = 10;

        try {
            printInputs(amount, years, interestRate);
            printResult(new LoanCalculator(amount, years, interestRate).calculateR
epaymentAmount());
        } catch (LoanCalculationException e) {
            System.out.println(e.getMessage());
        }
    }

    private static void printInputs(int amount, int years, double interestRate) {
        System.out.println("Calculating loan based on:");
        System.out.println("Principal:      " + amount);
        System.out.println("Loan Term:      " + years + " year" + ((years > 0) ?
"s" : ""));
        System.out.println("Interest Rate:  " + interestRate + "%");
    }

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

```

LoanCalculator.java

```

package com.javaeasily.demos;

public class LoanCalculator {
    private int amount;
    private int years;
    private double interestRate;

    public LoanCalculator(int amount, int years, double interestRate) throws LoanCalculationException {
        if (amount <= 0 || years <= 0 || interestRate <= 0.0) {
            throw new LoanCalculationException("Invalid values - cannot calculate repayment amount.");
        }

        this.amount = amount;
        this.years = years;
        this.interestRate = interestRate;
    }

    public double calculateRepaymentAmount() {
        double interestRateMultiplier = 1 + interestRate / 100;

        double currentAmountPayable = amount;
        int currentYear = 1;
        while (currentYear <= years) {
            currentAmountPayable = currentAmountPayable * interestRateMultiplier;
            currentYear++;
        }
        return currentAmountPayable;
    }

    public int getAmount() {
        return amount;
    }

    public int getYears() {
        return years;
    }

    public double getInterestRate() {
        return interestRate;
    }
}

```

LoanCalculationException.java

```
package com.javaeasily.demos;

public class LoanCalculationException extends Exception {
    public LoanCalculationException(String message) {
        super(message);
    }
}
```