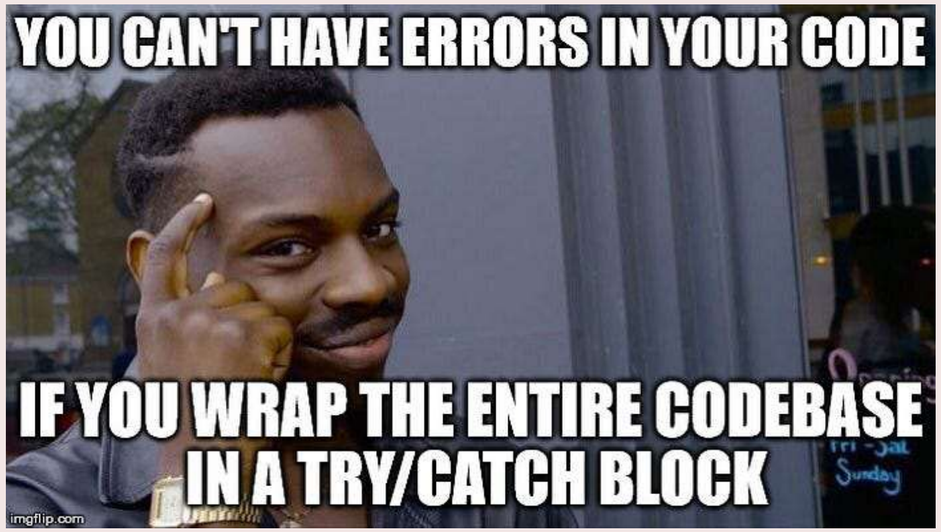
Exceptions

Errors

Errors



- there are many types of errors that can be used exceptions for example hardware problems. bad hard drive or bad data.
- when occurred this type of exceptions Java create a method called Throwable.
- this method have an information about the exception include the type, state for program and the moment than occurred the error

Advantaje of controller errors

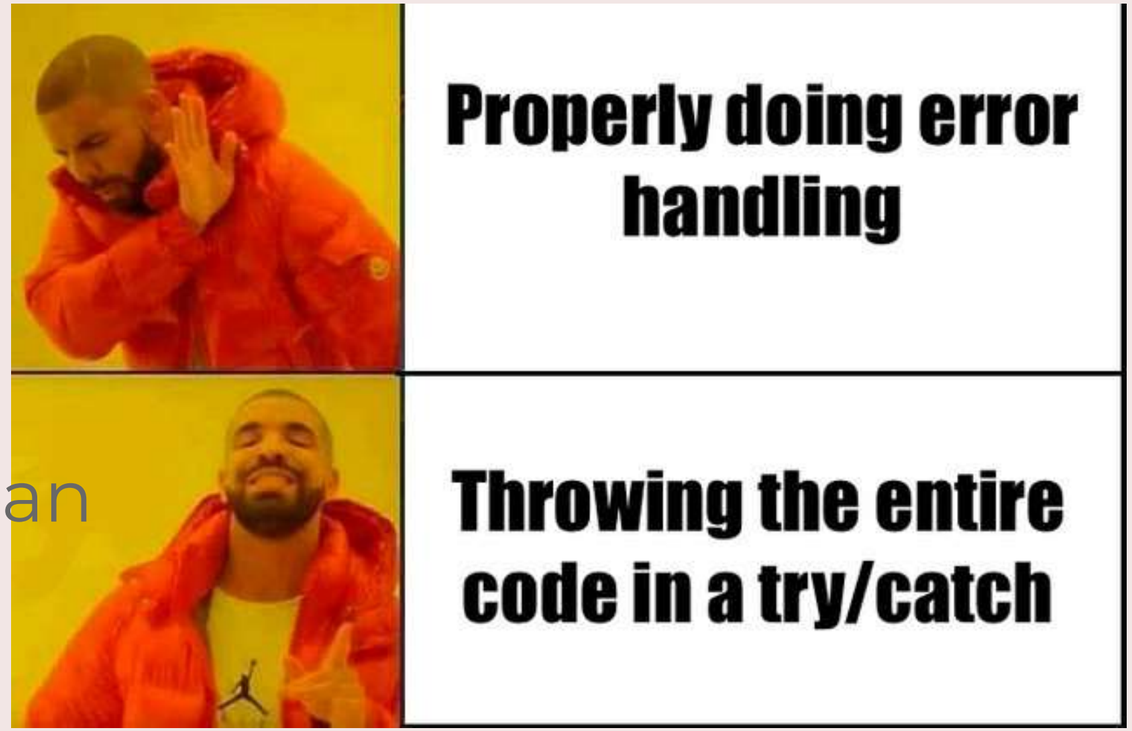
Separation of error handling from "normal" code.

Propagation of errors throughout the call stack.

Grouping of error types and differentiation between them.

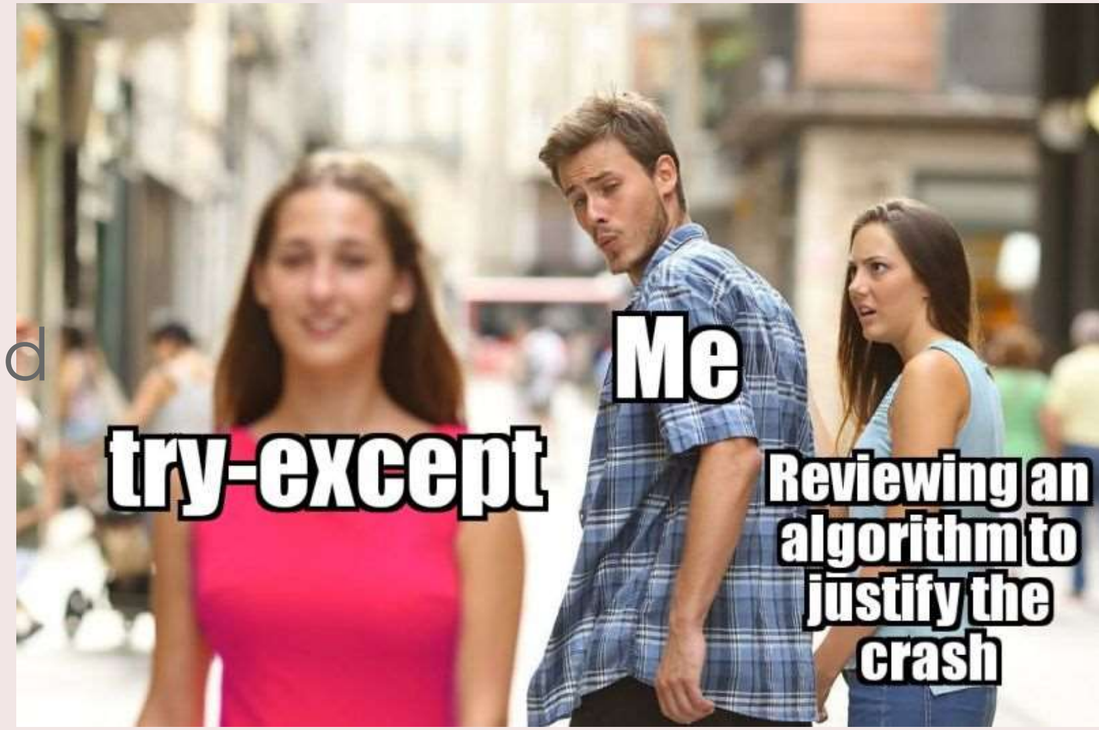
Separation of error handling from "normal" code.

- By using this method of exceptions. the mistakes have identified and controller for separated, this has a result a clean code and readable



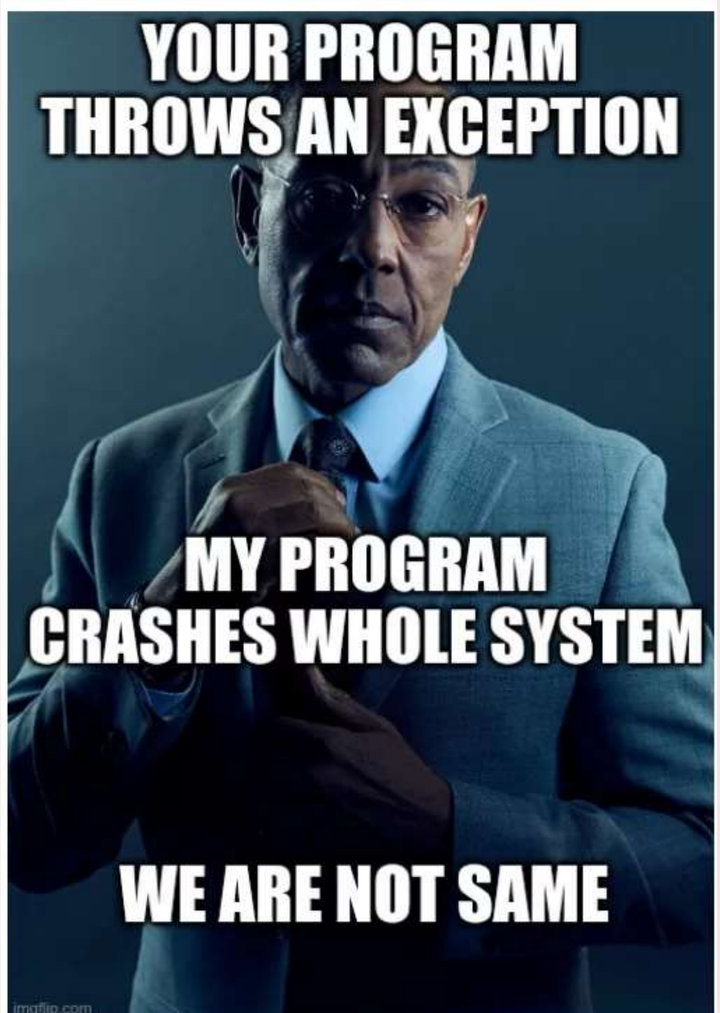
Propagation of errors throughout the call stack.

- Another advantage of the controller exceptions in Java is the capacity of spread mistakes through stack calls.
- when an exceptions occur is a method can be cast and spread a up if stack called until a find a controller of exceptions appropriate



Grouping of error types and differentiation between them.

- In Java, the exceptions organized in a hierarchy of class. this it means we can a defined and catch exceptions specifics for different types of mistakes
- For example, can catch exceptions specific for mistakes of entry and exit, mistakes of connection or calculate, or more



For example

```java
public class ThrowableExample {

    public static void main(String[] args) {
        boolean continuar = true;

        while (continuar) {
            try {
                Scanner scanner = new Scanner(source: System.in);

                System.out.print(s: "Enter the numerator: ");
                int numerator = scanner.nextInt();

                System.out.print(s: "Enter the denominator: ");
                int denominator = scanner.nextInt();

                divideNumbers(numerator, denominator);
                continuar = false;
            } catch (Throwable t) {
                System.out.println(x: "Exception caught: cannot split by those characters" );
                System.out.println(x: "Re-enter the numbers.");
            }
        }
    }

    public static void divideNumbers(int numerator, int denominator) {
        int result = numerator / denominator;
        System.out.println("Result: " + result);
    }
}
```

For example

Output - ThrowableExample (run)

```
run:
Enter the numerator: 3212
Enter the denominator: /
Exception caught: cannot split by those characters
Re-enter the numbers.
Enter the numerator: 32124
Enter the denominator: =
Exception caught: cannot split by those characters
Re-enter the numbers.
Enter the numerator: 3223
Enter the denominator: 0
Exception caught: cannot split by those characters
Re-enter the numbers.
Enter the numerator: 122222
Enter the denominator: 2
Result: 61111
BUILD SUCCESSFUL (total time: 16 seconds)
```