Exception Handling

León Jaramillo

softserve

Exception Handling

- Most of programming languages feature **exception handling**. Among them is **Ruby**.
- An **exception** is an unwanted or unexpected event that might stop the normal execution of a software (therefore, Ruby) program.
- Sometimes, an error or exception will just stop the normal execution of a program. Exception handling helps us to avoid this, allowing us to **take appropriate actions** instead.
- As in other languages, an exception is a **subclass** of the `Exception` class.
- Typical **sources** of an exception are:
  - Trying to read an inexistent file.
  - Trying to divide by zero.
  - Trying to access an array element outside its bounds.

Basic Syntax

- Managed code will be within a `begin`/`end` block.
- `rescue` statement allows us to define what happens when a specific exception arises.
- `else` statement allows us to define what happens when an unspecified exception arises.
- Code after the `ensure` statement will always be executed.
- We can use `rescue` with no specific exceptions.

```
begin
# Normal execution flow
rescue OneTypeOfException
# Managing one type of exception
rescue AnotherTypeOfException
# Managing other one
else
# Managing any other exception
ensure
# Always will be executed
end
```

softserve

Using `retry` statement

- By default, once an exception is raised, it might be managed, and that’s it.
- But we can try again the “normal” execution flow using `retry`.

```
begin
    # Exceptions raised by this code will
    # be caught by the following rescue clause
rescue
    # This block will capture all types of exceptions
    retry  # This will move control to the beginning of begin
end
```

Using `raise` statement

We can use `raise` statement to raise and exception.

```
raise
```

OR

```
raise "Error Message"
```

OR

```
raise ExceptionType, "Error Message"
```

OR

```
raise ExceptionType, "Error Message" [condition]
```

softserve

Using an exception’s variable

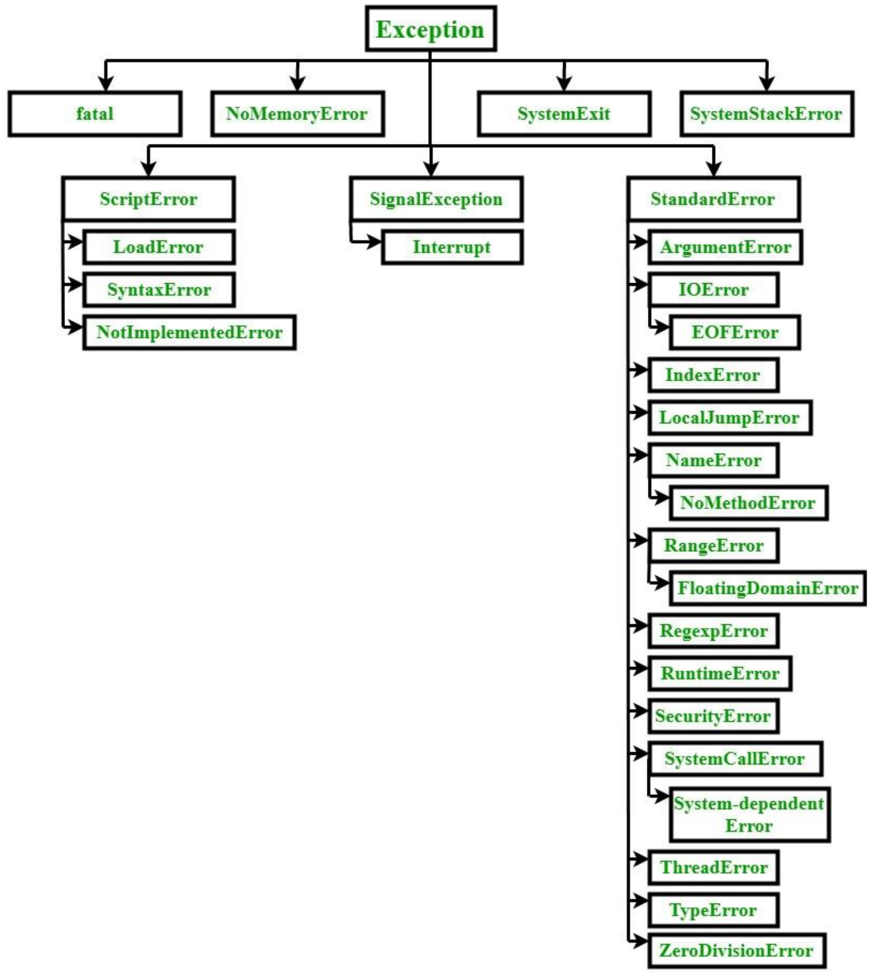
- As everything in Ruby, exceptions are objects. So, we might want to access their methods.
- Default methods can be seen here: https://docs.ruby-lang.org/en/3.2/Exception.html
- If we want to access those methods, we should set the variable name explicitly.

```
begin
   raise 'A test exception.'
rescue Exception => e
   puts e.message
   puts e.backtrace.inspect
end
```

softserve

Ruby Exceptions Hierarchy

- Most of programming languages which deal with exceptions have an **exceptions hierarchy**.
- Note that when you "`rescue`" an exception class, you also "`rescue`" its **subclasses**.
- `RuntimeError` class is raised by default.
- `StandardError` class is "rescued" by default.



From: geeksforgeeks.org

softserve

Useful Resources

- Exception class documentation: https://docs.ruby-lang.org/en/3.2/Exception.html
- Ruby Exceptions: https://www.tutorialspoint.com/ruby/ruby_exceptions.htm
- Ruby exception handling: https://www.geeksforgeeks.org/ruby-exception-handling/
- How to handle errors in Ruby: https://www.youtube.com/watch?v=FJDRNYu25AQ&ab_channel=SimonSomlai

softserve

COLOMBIA DC

Thanks! Any question?

softserve