Group members

Scott Riccardelli
Thomas Robinson
Quan Nguyen

Assignment 3 – Exceptions

- Provide a comprehensive and detailed description of the difference between exception handling in Java and Python.
- Provide code examples illustrating such differences.
- In case none of the team members is familiar with Python, a different language could be used, e.g., JavaScript, Rust or C#

Please look in the embedded code files `Assignment3.java` and `Assignment3.py` for code examples below.

There are a few major differences.

One difference is that Python contains two fields to store a reference to the Exception that caused the current Exception (if such a cause exists). These fields are `__context__` and `__cause__`. The `__context__` is set automatically, while the `__cause__` is set by using the `from` clause after the `raise` statement. In java, there is only a cause, accessed by using `getCause()`.

Python does not distinguish between "checked" and "uncheck" exceptions. This essentially means that all exceptions in Python are unchecked, and when raised cause the program to exit unless handled.

Python has a `try except else finally` block which includes an `else` showing that python does not discourage mingling exceptions with control flow. Java instead has more simply a `try catch finally` block.

In python, there are exceptions to use for control flow whereas in java no such exceptions exist and using generic or user defined exceptions to achieve this is discouraged. For example, python ships with a `StopIteration` exception which is raised when a generator is exhausted.

```
numbers = [1, 2, 3]
# a simple generator example
square_generator = (n**2 for n in numbers)

while True:
    try:
        # this will eventually raise StopIteration
        val = next(square_generator)
        print(f"Got value: {val}")
```

```
    except StopIteration:
        print("Caught StopIteration: The generator is empty. Breaking the loop.")
        break

print("Loop finished.")
```

In java, there is not any kind of similar exception.

Assignment3.java

```
public class Assignment3 {

    public static void secondFunc() throws Exception {
        throw new Exception("Test error in secondFunc");
    }

    public static void firstFunc() throws Exception {
        try {
            secondFunc();
        } catch (Exception e) {
            throw new Exception("Test error in firstFunc");
        }
    }

    public static void modFirstFunc() throws Exception {
        try {
            secondFunc();
        } catch (Exception e) {
            throw new Exception("Test error in modFirstFunc", e);
        }
    }

    public static void main(String[] args) {
        System.out.println("--- Starting firstFunc error handling ---");
        try {
            firstFunc();
        } catch (Exception e) {
            System.out.println("Error: " + e);
            System.out.println("Cause: " + e.getCause());
            System.out.println("Stack Trace:");
            e.printStackTrace(System.out);
        }
        System.out.println("--- End of firstFunc error handling ---\n");

        System.out.println("--- Starting modFirstFunc error handling ---");
        try {
            modFirstFunc();
```

```
        } catch (Exception e) {
            System.out.println("Error: " + e);
            System.out.println("Cause: " + e.getCause());
            System.out.println("Stack Trace:");
            e.printStackTrace(System.out);
        }
        System.out.println("--- End of modFirstFunc error handling ---");
    }
}
```

Assignment3.py

```
import traceback

def secondFunc():
    raise Exception("Test error in secondFunc")

def firstFunc():
    try:
        secondFunc()
    except Exception:
        raise Exception("Test error in firstFunc")

def modFirstFunc():
    try:
        secondFunc()
    except Exception as e:
        raise Exception("Test error in modFirstFunc") from e

def main():
    print('--- Starting firstFunc error handling ---')
    try:
        firstFunc()
    except Exception as e:
        print(f"Error: {e}")
        print(f"Cause: {e.__cause__}")
        print(f"Context: {e.__context__}")
        print("Stack Trace:")
        traceback.print_exc()
    print('--- End of firstFunc error handling ---\n')

    print('--- Starting modFirstFunc error handling ---')
    try:
        modFirstFunc()
    except Exception as e:
        print(f"Error: {e}")
        print(f"Cause: {e.__cause__}")
```

```
            print(f"Context: {e.__context__}")
            print("Stack Trace:")
            traceback.print_exc()
        print('--- End of modFirstFunc error handling ---')

if __name__ == "__main__":
    main()
```