EXEPTION HANDLING

EXEPTION HANDLING

Exception handling in a complex concurrent and distributed system (e.g. one involving cooperating rather than just competing activities) is often a necessary, but a very difficult, task. No widely accepted models or approaches exist in this area. The object-oriented paradigm, for all its structuring benefits, and real-time requirements each add further difficulties to the design and implementation of exception handling in such systems.

In this paper, we develop a general structuring framework based on the coordinated atomic (CA) action concept for handling exceptions in an object-oriented distributed system, in which exceptions in both the value and the time domain are taken into account. In particular, we attempt to attack several difficult problems related to real-time system design and error recovery, including action-level timing constraints, time-triggered CA actions, and time-dependent exception handling. The proposed framework is then demonstrated and assessed using an industrial real-time application-the Production Cell III case study

exception handling is the process of responding to the occurrence of exceptions – anomalous or exceptional conditions requiring special processing – during the

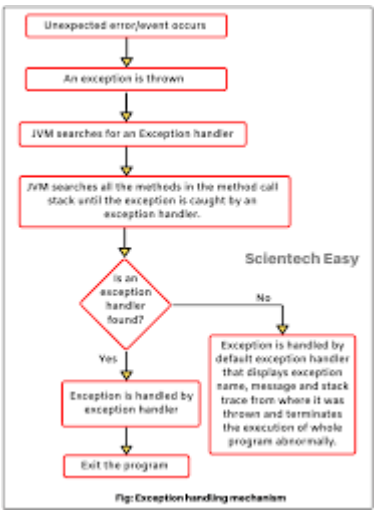
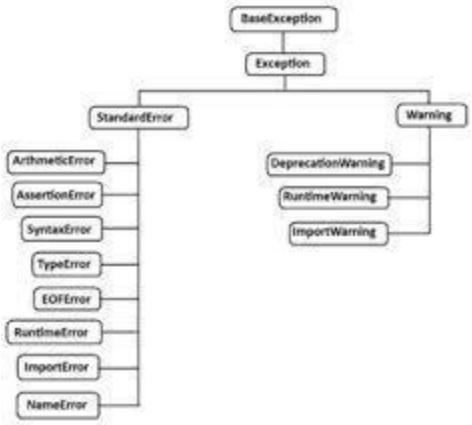


Fig: Exception handling mechanism

execution of a program. In general, an exception breaks the normal flow of execution and executes a pre-registered exception handler; the details of how this is done depend on whether.

it is a hardware or software exception and how the software exception is implemented. Exception handling, if provided, is facilitated by specialized programming language constructs, hardware mechanisms like interrupts, or operating system (OS) inter-process communication (IPC) facilities like signals. Some exceptions, especially hardware ones, may be handled so gracefully that execution can resume where it was interrupted.



An exception is a runtime error which can be handled by the programmer. That means if the programmer can guess an error in the program and he can do something to eliminate the harm caused by that error, then it is called an 'exception'. If the programmer cannot do anything in case of an error, then it is called an 'error' and not an exception.

All exceptions are represented as classes in Python. The exceptions which are already available in Python are called 'built-in' exceptions. The base class for all built-in exceptions is '**Base Exception**' class. From Base Exception class, the sub class '**Exception**' is derived. From Exception class, the sub classes '**Standard Error**' and '**Warning**' are derived.

Exception Handling

The purpose of handling errors is to make the program robust. The word ‘robust’ means ‘strong’. A robust program does not terminate in the middle. Also, when there is an error in the program, it will display an appropriate message to the user and continue execution. Designing such programs is needed in any software development. For this purpose, the programmer should handle the errors. When the errors can be handled, they are called exceptions.