# PS Assignment on Error Model

Harish

201401019

The way errors communicate and are dealt with is an ‘Error Model’.

**Reliable System:**

A system that never fails is a reliable one. Practically, creating such a system is impossible. Divide and conquer is the best strategy to create a system which is close to reliable one.

**Error Codes:**

Error codes are basically return values that indicate that there was an error/failure. Even though implementation is simple, there are some issues with this model.

Performance: We need to return extra values as return codes, this uses extra space and registers.

Man made errors such as forgetting to return error codes is also possible.

**Exceptions:**

Reliability is the most important of our requirements above when developing the Error Model. If you can’t react appropriately to failures, your system, by definition, won’t be very reliable. There are different types of exceptions:

Checked Exceptions: Generally we don’t know what all exceptions are possible. Different types are possible. Checked are those which we predict and write code to check.

Unchecked Exceptions: Exceptions that are not checked will go through some default flow and we won’t be able to control them.

**Bugs:**

Bugs are flaws in the code which occurs when code behaves in a way it is not supposed to work. Different classification of bugs are:

1.Plain old bugs

2.Arithmetic overflow

3.Stack overflow

4.Assertions

5.Contracts:

* + Contracts and assertions were proven to be side effect free. It is the biggest area of language innovation.
  + Pre-conditions:
    - pre-conditions are the most basic form of contract. This states that what are the conditions that needs to be satisfied before a method is dispatched.
    - Post conditions are nothing but what the expectation is(What the result should be).
  + There are three levels of contracts they are:
    - Weak, indicated by Contract.Weak.\*, meaning debug-only.
    - Normal, indicated simply by Contract.\*, leaving it as an implementation decision when to check them.
    - Strong, indicated by Contract.Strong.\*, meaning always checked.