7.1 Error

Errors are wrongs or mistakes that can make a program go wrong. Errors may be logical or may be typing mistakes.

An error may produce an erroneous or incorrect output or may terminate the execution of program shortly or even may cause the system crash.

So it is important to detect and manage all errors condition in the program so that the program will not terminate or crash during execution.

7.1.1 Types of Errors

The errors are classified into two categories:

1. Compile time errors
2. Run time errors.

7.1.1.1 Compile Time Errors

Java compilers detect all Syntax error and display it.

So these errors are known as compile-time errors when compiler displays errors, or suppose a program contains error, the compiler does not create .class file.

So it is necessary to fix all the errors before successful compilation and execution of program.

Example:-

```
// Following code contains errors.
class Error1
{
public static void main(string args[])
	{
```

```
System.out.println(“Hello Java !”)
            // ; missing
      }
}
```

The common compile time errors are:

1. Missing semicolon.
2. Missing in brackets in classes and methods.
3. Misspelling of identifier and keywords.
4. Missing a double quote in strings.
5. Use of undeclared variable.
6. Incompatible types of assignment/initialization.
7. Bad reference objects.

7.1.1.2 Run Time Errors

After successfully compiling a program .class file is created but may not run properly like programs may produce wrong result due to wrong logic or may terminate due to errors such as stack overflow.

❑**Commonly occurred runtime errors are:**

1. Dividing an integer by zero.
2. Accessing an element that is out of the bounds of an array.
3. Trying to a storing a value into an array of an incompatible class or type.
4. Trying to cast an instance of class one of its subclass.
5. Passing a parameter that is not in a valid range or value for a method.
6. Trying to illegally change the state of a thread.

7. Attempting to use a negative size for an array.
8. Using a null object reference as a legitimate object reference to access method or variable.
9. Converting invalid string to a number.
10. Accessing a character that is out of bounds of a string.