Common Errors

Errors are a type of Throwable that typically represent serious, unrecoverable problems that can lead to abnormal program termination. Errors are usually caused by external factors or severe system issues, and they are not meant to be caught and handled by the application code. Instead, they signal critical issues that often require manual intervention or system-level fixes.

Here are some common types of errors in Java:

- OutOfMemoryError: This error occurs when the Java Virtual Machine (JVM) cannot allocate enough memory to fulfill an allocation request, usually due to excessive memory usage by the application.
- StackOverflowError: This error is thrown when the call stack becomes full, often due to excessive recursion. It indicates that the depth of method calls has exceeded the JVM's stack size limit.
- NoClassDefFoundError: This error occurs when the JVM cannot find the definition of a class at runtime, even though the class was available during compilation. This could be due to issues with classpath configuration or missing class files.
- ClassFormatError: This error occurs when the JVM encounters an invalid class file format. It might indicate a corrupted or mismatched class file.

- UnsatisfiedLinkError: This error is thrown when the JVM cannot find a native library required by the application. It's often related to issues with loading native libraries.
- InternalError: This error indicates internal JVM errors that should not normally occur. It might be caused by bugs or inconsistencies within the JVM itself.
- UnknownError: This is a subclass of Error that can be used for unexpected errors that don't fit into other error categories.

Array ⟹

$\boxed{\text{int [] arr = new int[N];}}$ — 10

↳ size is fix

Arraylist ⟶ size is not fix

⟹ ArrayList list = new ArrayList() ;

⟹ ArrayList<Integer> list1 = new ArrayList<>();

↳ add

↳ get

↳ - - - .

```
class ErrorDemo {

        psvm (  ) {

            Arraylist<Byte[]> list= new AL<>();
              while(true){
               byte() b = new byte[1024*1024]
                  list.add(b);


       }

  }
```

```
public class OutOfMemoryDemo {
    public static void main(String[] args) {
        ArrayList<byte [] > list = new ArrayList<>() ;
         while(true){
             byte [] b = new byte[1024*1024] ;
             list.add(b) ;
         }
    }
}
```