Week 21 Lecture

File handling in Java means that how to read from and write to file in Java. Java provides the basic I/O package for reading and writing streams. java.io package allows to do all Input and Output tasks in Java.

File Writing:

```
import java.io.FileWriter;

public class WriteInFile {
    public static void main(String[] args) {
        try {
            FileWriter myWriter = new FileWriter("filename.txt");
            myWriter.write("Files in Java might be tricky, but it is fun 
enough!");
            myWriter.write("\nSecond line!");
            myWriter.close();
            System.out.println("Successfully wrote to the file.");
        } catch (Exception e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }

}
```

File Reading:

```
import java.io.FileReader;

public class ReadFromFile {

    public static void main(String args[]) throws Exception {
        FileReader filereadObj = new FileReader("filename.txt");
        int iterator;
        while ((iterator = filereadObj.read()) != -1) {
            System.out.print((char) iterator);
        }
        filereadObj.close();
    }
}
```

read() method is used to return a character in ASCII form. It returns -1 at the end of file.

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

Java Throw Exception

- In Java we have already defined exception classes such as ArithmeticException, NullPointerException, ArrayIndexOutOfBounds exception etc.

- These exceptions are set to trigger on different conditions. For example when we divide a number by zero, this triggers ArithmeticException, when we try to access the array element out of its bounds then we get ArrayIndexOutOfBoundsException.

We can define our own set of conditions or rules and throw an exception explicitly using throw keyword. For example, we can throw ArithmeticException when we divide number by 5, or any other numbers, what we need to do is just set the condition and throw any exception using throw keyword.

The Java throw keyword is used to explicitly throw an exception.

The syntax of java throw keyword is given below.

```
throw exception;
```

Let's see the example of throw IOException.

```
throw new IOException("sorry device error);
```

```
public class Test {
	static void validate(int age) {
		if (age < 18) {
			throw new ArithmeticException("You are not eligible to vote");
		} else {
			System.out.println("welcome to voting system");
		}
	}

	public static void main(String args[]) {
		validate(13);
		System.out.println("rest of the code...");
	}
}
```

```
try{
	validate(13);
}catch(ArithmeticException ex){
	ex.printStackTrace();
}
System.out.println("rest of the code...");
```

```
Exception in thread "main" java.lang.ArithmeticException: You are not eligible to vote
	at cw2.Test.validate(Test.java:6)
	at cw2.Test.main(Test.java:13)
```

Throws Exception

Any method that is capable of causing exceptions must list all the exceptions possible during its execution, so that anyone calling that method gets a prior knowledge about which exceptions are to be handled. A method can do so by using the **throws** keyword.

```
import java.io.*;
class Test{
    public static void main(String[] args) throws IOException{
      FileWriter file = new FileWriter("c:\\Data1.txt");
      file.write("This is file testing");
      file.close();
    }
}
```

Difference between throw and throws

throw	throws
throw keyword is used to throw an exception explicitly.	throws keyword is used to declare an exception possible during its execution. (Risky methods)
throw keyword is declared inside a method body.	throws keyword is used with method signature (method declaration).
We cannot throw multiple exceptions using throw keyword.	We can declare multiple exceptions (separated by commas) using throws keyword.

Static method

Static methods are utility methods that we want to expose to be used by other classes without the need of creating an instance. For example, Integer class.

```
// Integer class
int a = Integer.parseInt(“10”);

//Math class
int a = Math.pow(10, 5);
int b = Math.round(10.1);
```

```
class MathUtility{

      /*
       * To declare static method use static keyword.
       * Static methods are class level methods and can not access any instance
       * member directly. However, it can access members of a particular object
       * using its reference.
       *
       * Static methods are generally written as a utility method or it performs
       * task for all objects of the class.
       *
       */

      public static int add(int first, int second)
      {
            return first + second;
      }

      public static int subtract(int first, int second)
      {
            return first - second;
      }

}
```

```java
public class StaticMethodExample {

	public static void main(String[] args) {

		int result = MathUtility.add(1, 2);
		System.out.println("(1+2) is : " + result);

		int result = MathUtility.subtract(1, 2);
		System.out.println("(1-2) is : " + result);
	}

}
```