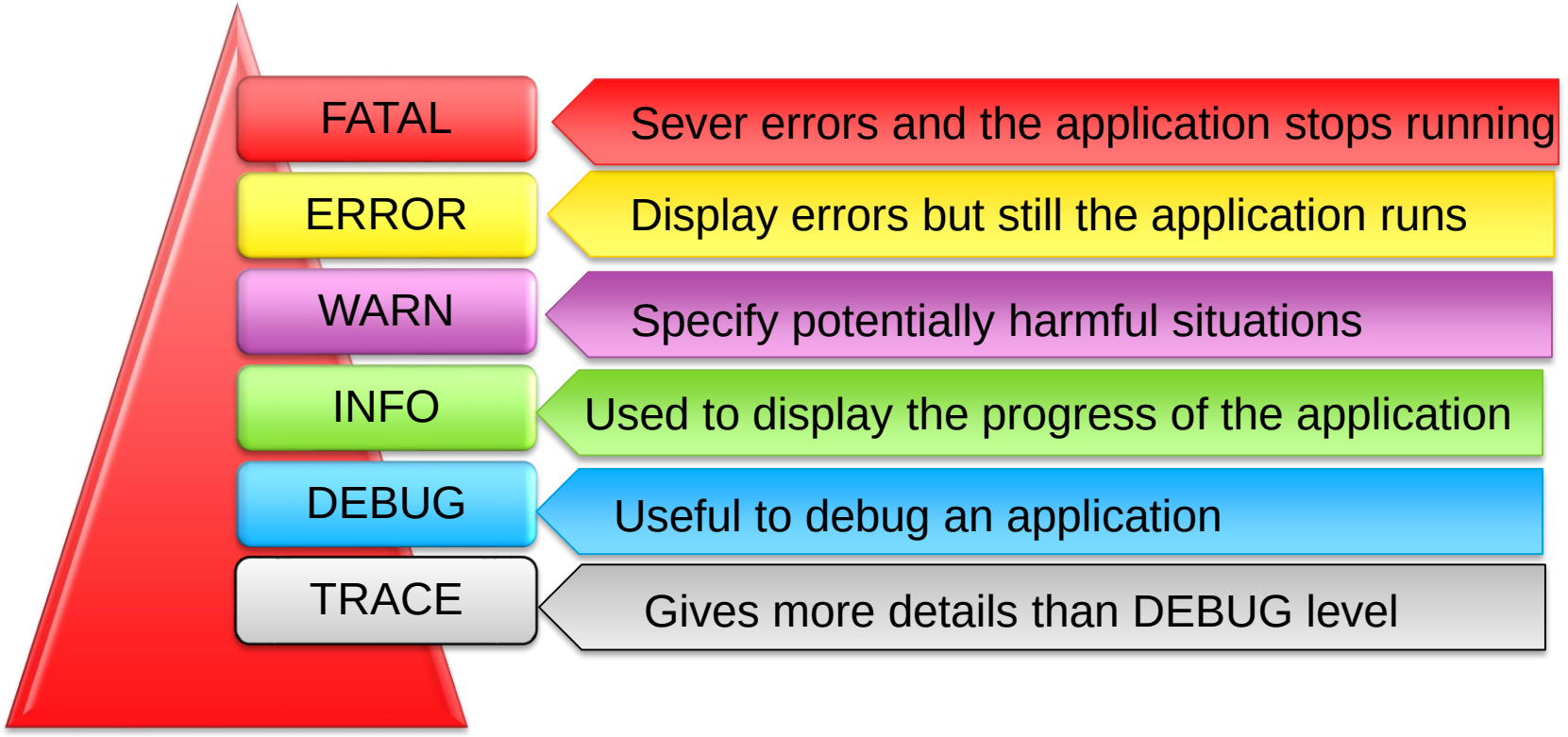
Logging Levels

Logging Levels

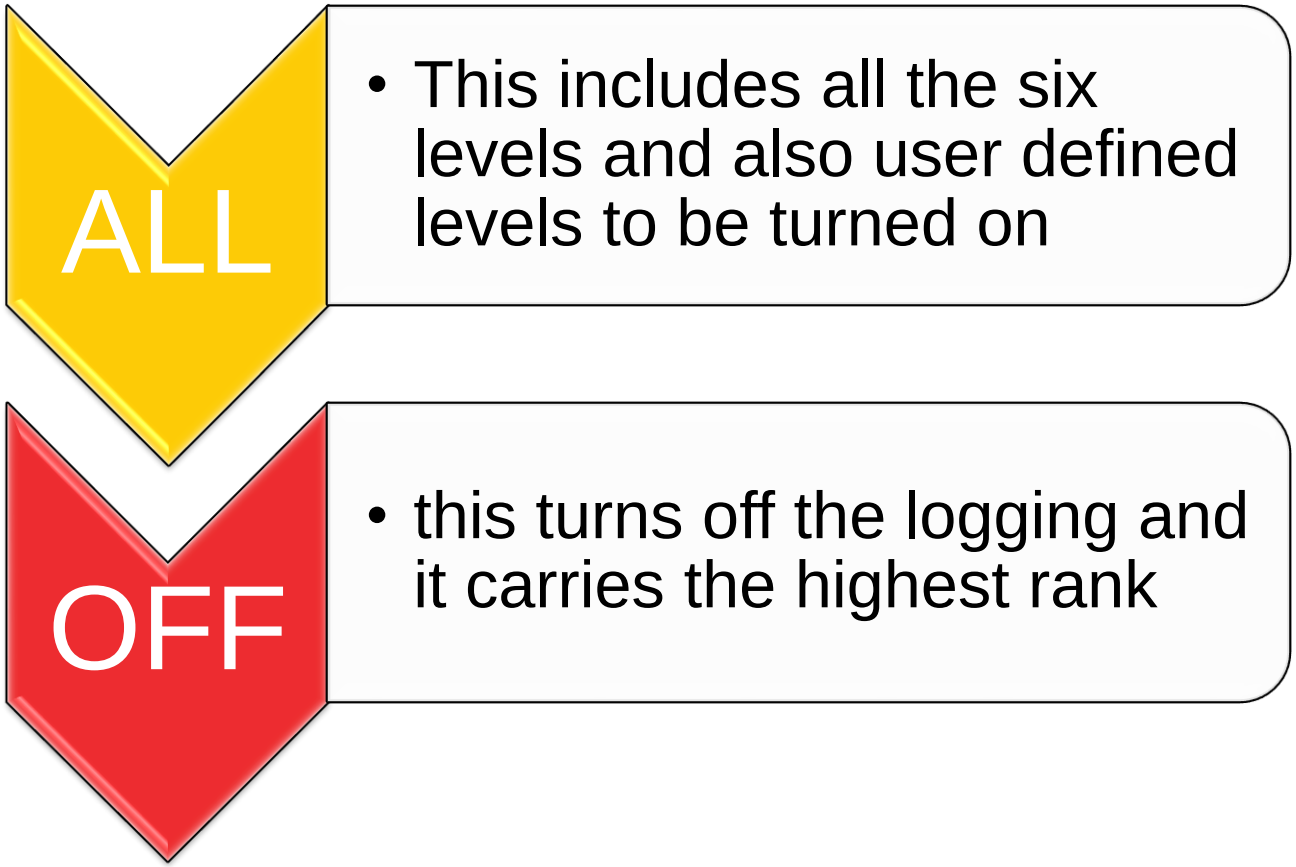
Logging Levels

org.apache.log4j provides 6 concrete levels for messaging which are assigned to Logger



Logging Levels

- **Other than these it also has 2 special levels**



Logging Levels

FATAL

- It points to sever error information which aborts the application execution
- `public void fatal(Object message);`

ERROR

- It points out to error information of the application that are not show stoppers
- `public void error(Object message);`

WARN

- It points to warning messages of exceptional behaviours of the application. These if not handled may lead to ERROR or FATAL later in usage `public void warn(Object message);`

INFO

- It points to messages which provides information on process of the application
- `public void info(Object message);`

DEBUG

- It points out to application's debug related informational events
- `public void debug(Object message);`

TRACE

- It points to more detailed information than DEBUG level
- `public void trace(Object message);`

Working of Levels

- Following is the order of levels starting with ALL at least p riority and OFF at highest priority
- ALL < TRACE < DEBUG < INFO < WARN < ERROR < F ATAL < OFF
- Loggers can be created with a level assigned to it through setLevel(Level lvl) method
- So that level is said to be the base and Logger displays all messages of that level and higher than that

Simple Demo

- Create a Java Project
- Add log4j-1.2.17.jar to the project using Project->Build Path -> Add externals jars
- Create log.properties file in src folder of the project

log.properties

```
log4j.rootLogger=DEBUG, SIMPLE

log4j.appender.SIMPLE=org.apache.log4j.ConsoleAppender

log4j.appender.SIMPLE.layout=org.apache.log4j.SimpleLayout
```

Simple Demo

- Create a SimpleDemo class

SimpleDemo.java

```
package com.wipro.log4j;


import org.apache.log4j.Logger;

public class SimpleDemo {

static final Logger logger = Logger.getRootLogger();
public static void main(String[] args) {
logger.debug("Sample debug message");
logger.info("Sample info message");
logger.warn("Sample warn message");
logger.error("Sample error message");
logger.fatal("Sample fatal message");
}

}
```

Simple Demo

- Output in the console since we had used a ConsoleAppender:

```
DEBUG - Sample debug message
INFO - Sample info message
WARN - Sample warn message
ERROR - Sample error message
FATAL - Sample fatal message
```

Assignment

1. Create a Hello World program which can display log entries in the console.
2. Create an application which will print from 1 to 100 in the console as well as in a log file.
3. Create an application which will print the elements of an int array. If it encounters an even number in the array, a log entry should be made ‘Even number encountered’ of level ‘WARN’.
4. Create an application which prints numbers infinitely. If the loop counter crosses 1 million, then log a ‘FATAL’ entry in the log file.
5.In the third assignment, add the following modifications:

a. Use a RollingFileAppender and create a log file named ‘even.log’
b. Display the date and time of the log entry in the following format: 2015-05-22 15:05:59

Summary

- Logging Levels
- Working of levels

Thank You