Multithreading in Java – Simple Notes

1. What is Multithreading?

Multithreading is the ability of a program to run multiple parts (threads) at the same time. A thread is a small, independent path of execution inside a program.

2. Why do we use Multithreading?

It helps in performing multiple tasks at once, using the CPU efficiently, making programs faster, and keeping applications responsive.

Examples of multithreading in real software:

• Downloading files while listening to music
• Video players
• Games
• Servers handling many users

3. Ways to Create Threads in Java:

```
There are two main ways – extending Thread class and implementing Runnable interface.
```

Method 1: Extending Thread class

```
class MyThread extends Thread { public void run() { System.out.println("Thread is
running"); } } public class Main { public static void main(String[] args) { MyThread t =
new MyThread(); t.start(); } }
```

Method 2: Implementing Runnable interface

```
class MyTask implements Runnable { public void run() { System.out.println("Thread is
running using Runnable"); } } public class Main { public static void main(String[] args)
{ Thread t = new Thread(new MyTask()); t.start(); } }
```

4. run() vs start():

run(): behaves like a normal method and does not start a new thread.
start(): creates a new thread and internally calls run(). Always use start().

5. Thread States in Java:

NEW – object is created
RUNNABLE – ready to run or running
BLOCKED – waiting for a lock
WAITING – waiting without time limit
TIMED_WAITING – waiting for fixed time (sleep, wait(ms))
TERMINATED – thread finished

6. Example of two threads running together:

```
class A extends Thread { public void run() { for(int i=1; i<5; i++) {
System.out.println("Hello hi"); try { Thread.sleep(10); } catch(Exception e) {
e.printStackTrace(); } } } } class B extends Thread { public void run() { for(int i=1;
i<5; i++) { System.out.println("Ok Bye"); try { Thread.sleep(10); } catch(Exception e) {
e.printStackTrace(); } } } } public class Main { public static void main(String[] args)
{ A a1 = new A(); B b1 = new B(); a1.start(); b1.start(); } }
```

7. Thread.sleep():

```
Thread.sleep(ms) pauses the thread for a given time and puts it into TIMED_WAITING
state.
```

8. Thread.join():

```
join() makes the current thread wait until another thread finishes execution.
```

9. What is e.printStackTrace()?

It prints details of the exception such as type, message, and line numbers to help debugging.

10. Advantages of Multithreading:

• Faster execution
• Better CPU usage
• Responsive programs
• Resource sharing

11. Disadvantages of Multithreading:

• Complex to program
• Hard to debug
• Risk of data inconsistency
• Requires synchronization

12. One-line Summary:

Multithreading allows multiple threads to run at the same time to make programs efficient and fast.