Java Threads Tutorial

2020 Spring COSI 131A

Java Threads

- Motivation

    - Multiprocessing is the way to go

        - Run multiple tasks in parallel

    - Creating multiple processes is costly

        - Lots of copying

- Multithreading

    - A process can create multiple threads

        - Threads contain a piece of code that can run concurrently

        - Handled by the JVM

    - Thread is a "lightweight process"

        - Thread creation is simpler – less copying

        - Threads share memory

How to declare Java Threads

- Extend the Thread class

```
class MyThread extends Thread {

    public void run() { … }

}
```

- **run()** is the method executed by a thread
- It is declared as abstract in the superclass

OR

- Implement the Runnable interface

```
class MyThread implements Runnable {

    public void run() { … }

}
```

How to start Java Threads

```
public static void main(String args[]){

    MyThread mt = new MyThread();

    Thread t = new Thread(mt);

    t.start();

    // do something else

    t.join();

    // done

}
```

Create an instance of MyThread that implements Runnable

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

How to start Java Threads

```
public static void main(String args[]){
```

```
    MyThread mt = new MyThread();
```

Create an instance of the Thread class with parameter our class MyThread. That's our thread t.

```
    Thread t = new Thread(mt);
```

```
    t.start();

    // do something else

    t.join();

    // done

}
```

How to start Java Threads

```
public static void main(String args[]){

    MyThread mt = new MyThread();

    Thread t = new Thread(mt);

    t.start();

    // do something else

    t.join();

    // done

}
```

Start thread t
a) spawn a new thread and
b) call the run() method

How to start Java Threads

```
public static void main(String args[]){

    MyThread mt = new MyThread();

    Thread t = new Thread(mt);

    t.start();

    // do something else

    t.join();

    // done

}
```

Perform some other tasks in the main thread. Main thread and thread t run in parallel at this point

How to start Java Threads

```
public static void main(String args[]){

    MyThread mt = new MyThread();

    Thread t = new Thread(mt);

    t.start();

    // do something else

    t.join();

    // done

}
```

Wait for thread t to finish running before executing next command (optional). If t is interrupted it will throw an Interrupted Exception.

If t is an independent thread then the join() is not necessary

How to start Java Threads

```
public static void main(String args[]){

    MyThread mt = new MyThread();

    Thread t = new Thread(mt);

    t.start();

    // do something else

    t.join();

    // done

}
```

Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs

Thread t finished and the main method terminates

How to send interrupts to threads

- Threads can be interrupted

  - An event occurred and we want to let the threads know

How to send interrupts to threads

- Threads can be interrupted

  - An event occurred and we want to let the threads know

- `t.interrupt()`

  - a) sets the interrupted flag in target thread t

  - b) sends `InterruptedException` to target thread t

  - *It does not terminate thread t!!*

How to interrupt threads

- Threads can be interrupted

  - An event occurred and we want to let the threads know

- `t.interrupt()`

  - a) sets the interrupted flag in target thread t

  - b) sends `InterruptedException` to target thread t

  - *It does not terminate thread t!!*

- **`t.isInterrupted()`**

  - Returns whether thread t has received an interrupt or not (interrupted flag)

Handling an interrupt

- An interrupt asks a thread to stop what it's doing in an orderly fashion.

- However, the exact response to the interrupt depends on the thread's state, and how the thread is implemented

    - If you do not handle the interrupt, the thread will continue running

    - Ideally an interrupt should not shutdown a thread immediately, in the middle of a computation

Handling an interrupt – example 1

Thread t

```
while(!Thread.currentThread.isInterrupted()){
        System.out.println("Still running...");
}
System.out.println("Interrupted!");
// terminate
```

Main thread

```
t.start();
// do something
t.interrupt();
```

Handling an interrupt – example 2

Thread t

```
try {

    Thread.sleep(5000);

}

catch(InterruptedException ie){

    System.out.println("I got interrupted!");

}
```

Main thread

```
t.start();
// do something
t.interrupt();
```

PA2 tasks

- Make Filters run concurrently

- Support background commands

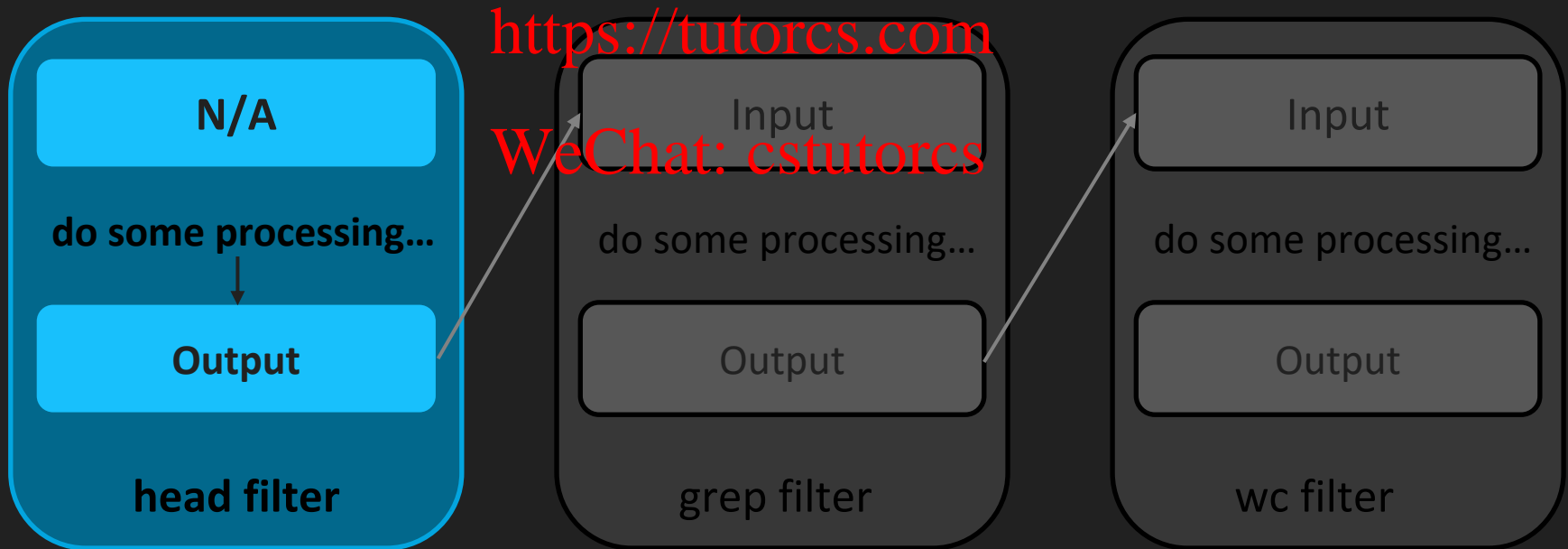- Implement repl_jobs command

- Implement kill command

Make Filters concurrent

```
> head file.txt | grep foo | wc
```

PA1: Filters run sequentially... First head,

N/A

do some processing...

Output

head filter

Input

do some processing...
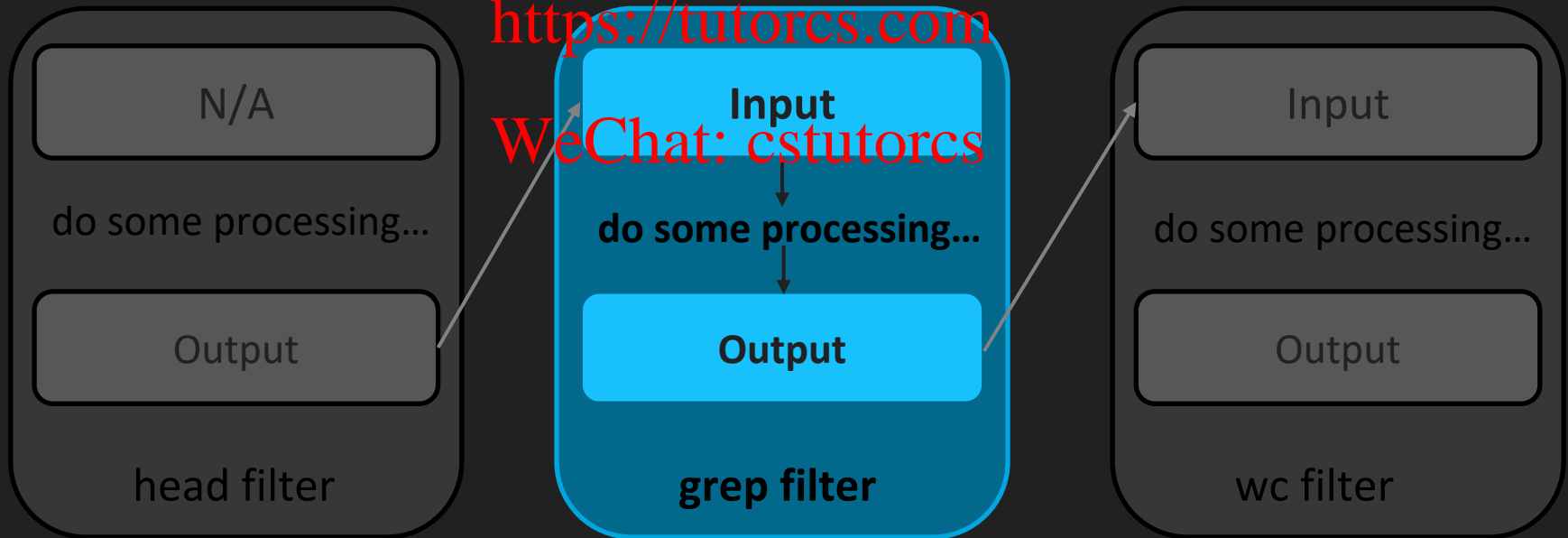
Output

grep filter

Input

do some processing...

Output

wc filter

Make Filters concurrent

```
> head file.txt | grep foo | wc
```

Then grep,   Assignment Project Exam Help

https://tutorcs.com

head filter	grep filter	wc filter
N/A	Input	Input
do some processing...	do some processing...	do some processing...
Output	Output	Output

WeChat: cstutorcs

Make Filters concurrent

```
> head file.txt | grep foo | wc
```

And finally wc

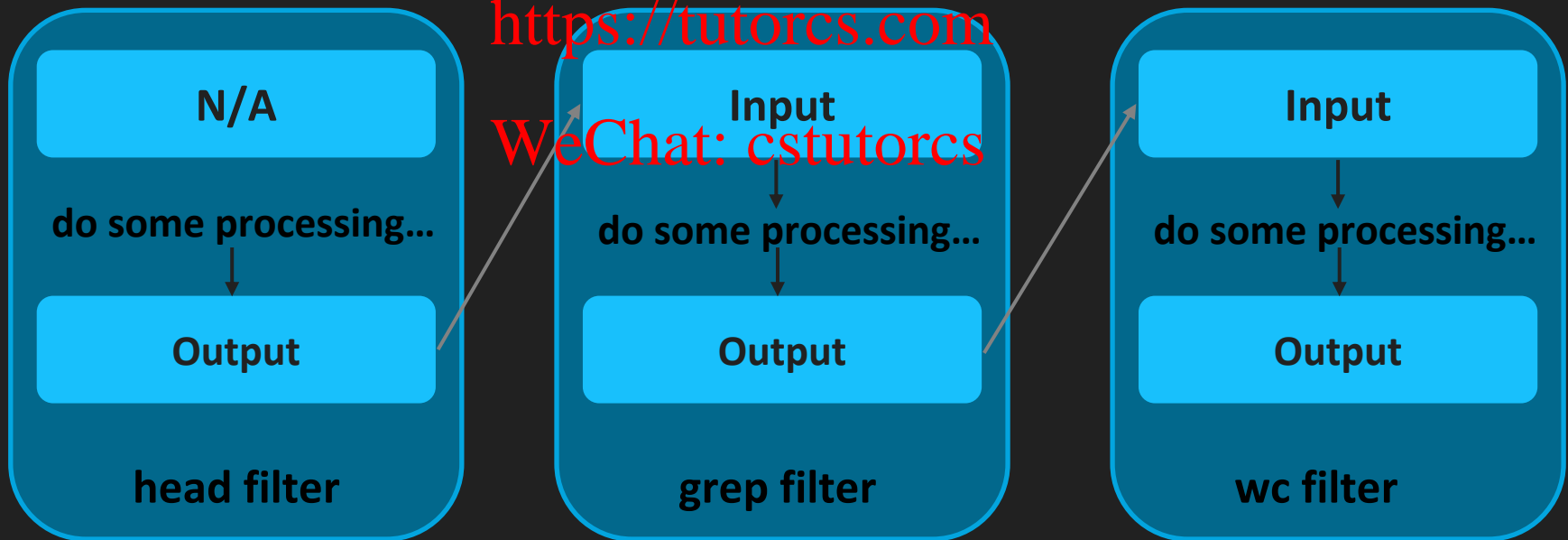| head filter	grep filter	wc filter
N/A	Input	**Input**
do some processing…	do some processing…	**do some processing…**
Output	Output	**Output**

Make Filters concurrent

```
> head file.txt | grep foo | wc
```

PA2: Each filter will be started as a thread and run concurrently

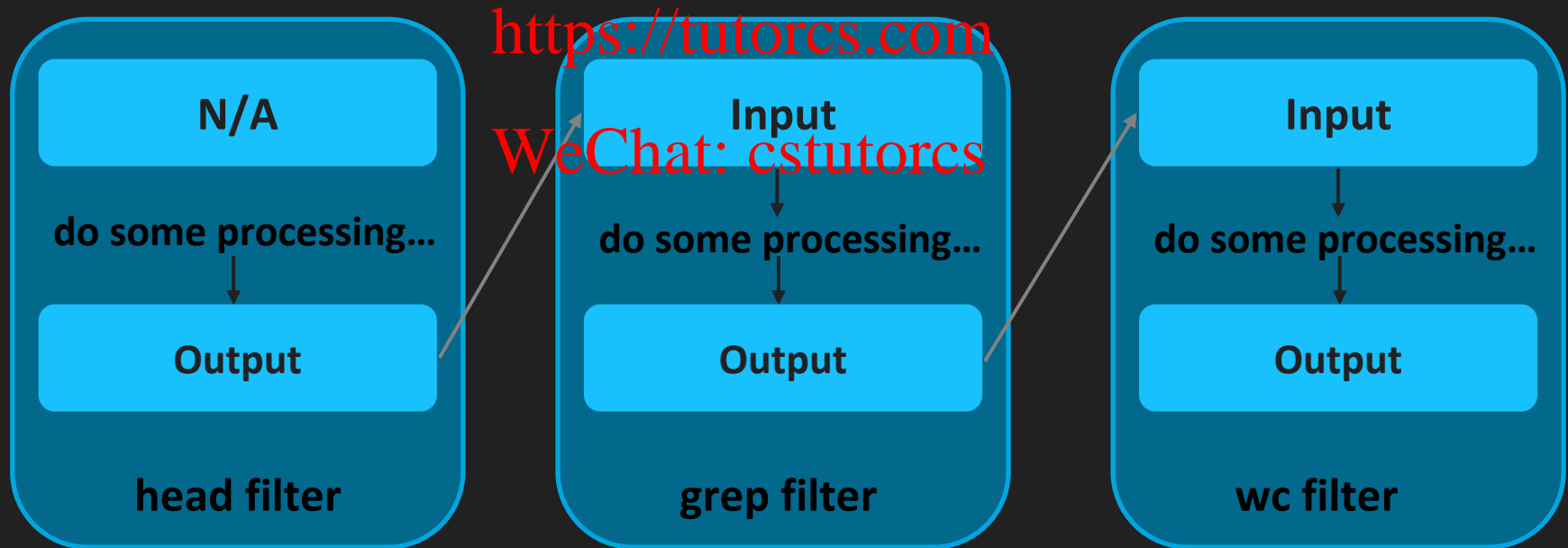| head filter	grep filter	wc filter
N/A	**Input**	**Input**
do some processing…	do some processing…	do some processing…
Output	**Output**	**Output**

Make Filters concurrent

```
> head file.txt | grep foo | wc
```

Sending the output to the next filter as soon as it becomes available

Background commands

- Allow a command to run in the background
    - Don't wait for it to finish
    - Allow the user to enter a new command
    - Use the '&' at the end of the command
    > `head –100000 reviews.txt | grep good > good.txt &`
        - This will let the command run in the background
- Display the background commands with repl_jobs
    > `repl_jobs`
        1. `head –100000 reviews.txt | grep good > good.txt &`

Kill a background command

- Terminate the execution of a command

```
> repl_jobs
    1. head -100000 reviews.txt | grep good > good.txt &
> kill 1
```

- This will terminate the background command with number 1

Assignment Project Exam Help

https://tutorcs.com

Questions?

WeChat: cstutorcs