Priority of Thread

1.What is Priority of thread?

Each thread has a priority. Priorities are represented by a number between 1 and 10. In most cases, the thread scheduler schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses. Note that not only JVM a Java programmer can also assign the priorities of a thread explicitly in a Java program.

The valid range of thread priorities is 1-10. Where 1 is min priority and 10 is max priority

- Thread scheduler will use priorities while allocating processor.
- The thread which is having highest priority will get the chance first.

For example, imagine you have a user interface thread and a background thread in your application. The user interface thread is responsible for displaying data on the screen and handling user input, while the background thread performs a long-running task, such as downloading a large file from the internet. Without thread priority, both threads would compete for the same resources and the performance of the user interface thread would be affected by the background task. By setting the priority of the user interface thread to be higher, you ensure that it receives more CPU time and the user experience is not affected by the background task.

2. what are Setter & Getter Methods of Thread Priority?

public final int getPriority(): The java.lang.Thread.getPriority() method returns the priority of the given thread.

public final void setPriority(int newPriority): The java.lang.Thread.setPriority() method updates or assign the priority of the thread to newPriority. The method throws IllegalArgumentException if the value newPriority goes out of the range, which is 1 (minimum) to 10 (maximum).

Example

```
public class MyThread extends Thread {
    public void run() {
        // code to run in the thread
    }

    public static void main(String[] args) {
        MyThread thread = new MyThread();
        // set the priority to 7
        thread.setPriority(7);
        // get the priority
        int priority = thread.getPriority();
        System.out.println("Thread priority: " + priority);
    }
```

3.Methods used in Thread Priority?

`currentThread()` is a static method of the Thread class in Java that returns a reference to the currently executing thread. This method can be used to access the properties and methods of the current thread, such as its name and priority.

For example, consider the following code:

```
Thread current = Thread.currentThread();
System.out.println("Name of current thread: " + current.getName());
System.out.println("Priority of current thread: " + current.getPriority());
```

This code will print the name and priority of the currently executing thread.

It's important to note that currentThread() can be used in any thread context, whether it is a main thread or a child thread, it will always refer to the current thread that is running in that context.

`setName(String name)` is a method of the Thread class in Java that sets the name of the thread.

For example, consider the following code:

```
Thread t = new Thread();
t.setName("My Thread");
System.out.println("Thread name: " + t.getName());
```

This code creates a new thread and sets its name to "My Thread" using the setName() method. Then it prints the name of the thread using the **`getName()`** method. The output will be "Thread name: My Thread".

4.What are three constants defined in thread class?

There are several constants defined in the Thread class in Java, but three of the most commonly used are:

MIN_PRIORITY: This constant represents the minimum priority that a thread can have, which is 1.

NORM_PRIORITY: This constant represents the default priority that a thread is given when it is created, which is 5.

MAX_PRIORITY: This constant represents the maximum priority that a thread can have, which is 10.

Here is an example of how to use these constants:

```
Thread t1 = new Thread();
t1.setPriority(Thread.MIN_PRIORITY);

Thread t2 = new Thread();
t2.setPriority(Thread.NORM_PRIORITY);

Thread t3 = new Thread();
t3.setPriority(Thread.MAX_PRIORITY);
```

In this example, we create three threads and set their priorities using the constants MIN_PRIORITY, NORM_PRIORITY, and MAX_PRIORITY respectively.It's worth noting that, the above constants are just a convention and it depends on the underlying operating system to support the thread priorities. Some operating systems may not support priorities at all, and others may support a different range of priorities.

5.Explain IllegalArgumentException in thread priority?

The Thread class in Java throws an IllegalArgumentException when the priority value passed to the setPriority() method is not within the valid range of 1 to 10.

Here is an example of how this exception can be thrown:

```
class MyThread extends Thread {
    public void run() {
        // code to run in the thread
    }
}

MyThread thread = new MyThread();

try {
    // set the priority to 11 (invalid)
    thread.setPriority(11);
} catch (IllegalArgumentException e) {
    System.out.println("Invalid priority: " + e.getMessage());
}
```

In this example, we try to set the priority of the thread to 11, which is outside of the valid range of 1 to 10. As a result, the setPriority() method throws an IllegalArgumentException with the message "Invalid priority: Priority out of range", which is caught by the catch block and printed to the console.