~\OneDrive - VietNam National University - HCM INTERNATIONAL UNIVERSITY\Desktop\DSA\DSA LAB NEW\Lab 3 Stacks & Queues\ITITSB22029_DoMinhDuy_Lab3\Task Scheduling (Queue)\TaskSchedulerApp.java

```java
// In a task scheduling system, tasks arrive at a processing unit in the order they were created, and they must be
// processed in the same order (First In, First Out). Some tasks take longer than others to complete, and
// occasionally, new high-priority tasks arrive that need to be processed before regular tasks. High-priority tasks
// are always processed immediately, but regular tasks continue in the original order after the high-priority tasks
// are handled.
// Your task is to simulate this task scheduling system using a queue and priority queue.

// Problem Description:
// You need to implement a task scheduling system with the following operations:
// 1. add_task(task_name, is_priority): Add a task to the queue. If is_priority is True, it’s a high-priority task
// and should be processed before regular tasks.
// 2. process_task(): Process the next task in the queue (priority tasks first, followed by regular tasks).
// Output the task being processed.

// Input:
//  A series of operations (e.g., add_task("task1", False), process_task()).

// Output:
//  The task that is processed after each process_task() operation.

// Example:
// Input:
// add_task("task1", False)
// add_task("task2", False)
// add_task("urgent_task", True)
// process_task()
// process_task()
// process_task()
// Output:
// Process urgent_task
// Process task1
// Process task2

// Key Challenges:
// 1. Queue Operations: Implementing task scheduling with a regular queue for normal tasks and a priority
// queue for urgent tasks.
// 2. Priority Management: Students must handle two different types of tasks and process them in the right
// order.
// 3. Edge Cases: Consider cases where all tasks are high-priority or no tasks are available to process.
```

```java

import java.util.LinkedList;
import java.util.Queue;
import java.util.PriorityQueue;
import java.util.Comparator;
import java.util.Scanner;

// Class representing a task
class Task {
    String name;
    boolean isPriority;

    public Task(String name, boolean isPriority) {
        this.name = name;
        this.isPriority = isPriority;
    }
}

// Task Scheduler Class
class TaskScheduler {
    private Queue<Task> regularQueue; // Queue for regular tasks
    private PriorityQueue<Task> priorityQueue; // Priority queue for high-priority tasks

    public TaskScheduler() {
        regularQueue = new LinkedList<>();
        priorityQueue = new PriorityQueue<>(Comparator.comparingInt(task -> task.isPriority ? 0 : 1));
    }

    // Add a task to the appropriate queue
    public void add_task(String taskName, boolean isPriority) {
        Task newTask = new Task(taskName, isPriority);
        if (isPriority) {
            priorityQueue.offer(newTask);
        } else {
            regularQueue.offer(newTask);
        }
    }

    // Process the next task
    public String process_task() {
        // Check if there are high-priority tasks first
        if (!priorityQueue.isEmpty()) {
            Task task = priorityQueue.poll(); // Get and remove the highest priority task
            return "Process " + task.name;
        } else if (!regularQueue.isEmpty()) {
            Task task = regularQueue.poll(); // Get and remove the next regular task
            return "Process " + task.name;
        } else {
            return "No tasks to process";
        }
    }
}

// Main Class to Test the Task Scheduler
public class TaskSchedulerApp {
    public static void main(String[] args) {
        TaskScheduler scheduler = new TaskScheduler();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.print("Enter command (add_task(taskName, isPriority) or process_task()) or 'exit' to quit: ");
            String command = scanner.nextLine().trim();

            if (command.equals("exit")) {
                break;
            } else if (command.startsWith("add_task")) {
                // Extract the parameters from the command
                String parameters = command.substring(command.indexOf("(") + 1, command.indexOf(")"));
                String[] parts = parameters.split(","); // Split by comma
                String taskName = parts[0].trim().replace("\"", ""); // Remove quotes from task name
                boolean isPriority = Boolean.parseBoolean(parts[1].trim()); // Convert to boolean
                scheduler.add_task(taskName, isPriority);
            } else if (command.equals("process_task()")) {
                String result = scheduler.process_task();
                System.out.println(result); // Output the processed task
            } else {
                System.out.println("Invalid command.");
            }
        }

        scanner.close();
    }
}

```