How to Use the Task Management System Artifact

By following these instructions, you can set up, compile, and run the Task management system artifact. The provided test data will demonstrate the enhancements effectively, showcasing how tasks are managed, sorted, and searched within the system.

Prerequisites: Set up

1. Java Development Kit (JDK): Ensure you have the JDK installed.

2. Environment Variables: Set up your environment variables to include JAVA\_HOME and add JAVA\_HOME/bin to your system's PATH.

Step-by-Step Instructions

Step 1: Set Up Directory Structure

1. Open a Terminal or Command Prompt:

- On Windows, open Command Prompt or PowerShell.

- On macOS and Linux, open the Terminal application.

2. Navigate to Your Working Directory:

- Use the `cd` command to change to the directory where you want to create your project.

- Create the main directory and navigate into it:

Step 2: Create Java Files and Add Code

Create and write the following Java files inside the `taskservice` directory:

1. Task.java:

- Create a new file named `Task.java`.

- Open the file and add the following code:

package taskservice;

import java.util.Comparator;

public class Task {

private String id;

private String name;

private String description;

// Constructor to initialize the Task object

public Task(String id, String name, String description) {

this.id = id;

this.name = name;

this.description = description;

}

// Getter for the 'id' field

public String getId() {

return id;

}

// Setter for the 'id' field

public void setId(String id) {

this.id = id;

}

// Getter for the 'name' field

public String getName() {

return name;

}

// Setter for the 'name' field

public void setName(String name) {

this.name = name;

}

// Getter for the 'description' field

public String getDescription() {

return description;

}

// Setter for the 'description' field

public void setDescription(String description) {

this.description = description;

}

// Overriding the 'equals' method for Task comparison

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (this.getClass() != obj.getClass())

return false;

Task t = (Task) obj;

return getId().equals(t.getId());

}

// Comparator to compare Task objects by 'id'

public static Comparator<Task> compareById = new Comparator<Task>() {

@Override

public int compare(Task t1, Task t2) {

return t1.getId().compareTo(t2.getId());

}

};

// Comparator to compare Task objects by 'name'

public static Comparator<Task> compareByName = new Comparator<Task>() {

@Override

public int compare(Task t1, Task t2) {

return t1.getName().compareTo(t2.getName());

}

};

// Overriding the 'toString' method to provide a string representation of Task

@Override

public String toString() {

return "Task ID: " + getId() + "\nName: " + getName() + "\nDescription: " + getDescription() + "\n";

}

}

2. TaskLinkedList.java:

- Create a new file named `TaskLinkedList.java`.

- Open the file and add the following code:

package taskservice;

import java.util.ArrayList;

import java.util.List;

class TaskLinkedList {

private Node head;

private class Node {

Task task;

Node next;

Node(Task task) {

this.task = task;

}

}

// Add task to the linked list

public void add(Task task) {

Node newNode = new Node(task);

if (head == null) {

head = newNode;

} else {

Node current = head;

while (current.next != null) {

current = current.next;

}

current.next = newNode;

}

}

// Delete task by ID

public boolean delete(String id) {

if (head == null) {

return false;

}

if (head.task.getId().equals(id)) {

head = head.next;

return true;

}

Node current = head;

while (current.next != null && !current.next.task.getId().equals(id)) {

current = current.next;

}

if (current.next != null) {

current.next = current.next.next;

return true;

}

return false;

}

// Find task by ID

public Task findById(String id) {

Node current = head;

while (current != null) {

if (current.task.getId().equals(id)) {

return current.task;

}

current = current.next;

}

return null;

}

// Convert linked list to list

public List<Task> toList() {

List<Task> list = new ArrayList<>();

Node current = head;

while (current != null) {

list.add(current.task);

current = current.next;

}

return list;

}

}

3. TaskBinarySearch.java:

- Create a new file named `TaskBinarySearch.java`.

- Open the file and add the following code:

package taskservice;

import java.util.List;

class TaskBinarySearch {

public static int binarySearch(List<Task> tasks, String id) {

int low = 0;

int high = tasks.size() - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

Task midTask = tasks.get(mid);

if (midTask.getId().compareTo(id) == 0) {

return mid;

} else if (midTask.getId().compareTo(id) < 0) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1; // ID not found

}

}

4. TaskSorter.java:

- Create a new file named `TaskSorter.java`.

- Open the file add the following code:

package taskservice;

import java.util.ArrayList;

import java.util.List;

class TaskSorter {

public static void mergeSort(List<Task> tasks) {

if (tasks.size() <= 1) {

return;

}

List<Task> left = new ArrayList<>();

List<Task> right = new ArrayList<>();

int middle = tasks.size() / 2;

for (int i = 0; i < middle; i++) {

left.add(tasks.get(i));

}

for (int i = middle; i < tasks.size(); i++) {

right.add(tasks.get(i));

}

mergeSort(left);

mergeSort(right);

merge(tasks, left, right);

}

private static void merge(List<Task> tasks, List<Task> left, List<Task> right) {

int leftIndex = 0;

int rightIndex = 0;

int tasksIndex = 0;

while (leftIndex < left.size() && rightIndex < right.size()) {

if (left.get(leftIndex).getName().compareTo(right.get(rightIndex).getName()) <= 0) {

tasks.set(tasksIndex++, left.get(leftIndex++));

} else {

tasks.set(tasksIndex++, right.get(rightIndex++));

}

}

while (leftIndex < left.size()) {

tasks.set(tasksIndex++, left.get(leftIndex++));

}

while (rightIndex < right.size()) {

tasks.set(tasksIndex++, right.get(rightIndex++));

}

}

}

5. TaskPriorityQueue.java:

- Create a new file named `TaskPriorityQueue.java`.

- Open the file and add the following code:

package taskservice;

import java.util.PriorityQueue;

class TaskPriorityQueue {

private PriorityQueue<Task> taskQueue;

public TaskPriorityQueue() {

taskQueue = new PriorityQueue<>(Task.compareById);

}

public void addTask(Task task) {

taskQueue.add(task);

}

public Task getHighestPriorityTask() {

return taskQueue.poll();

}

public boolean removeTask(Task task) {

return taskQueue.remove(task);

}

}

6. TaskServiceCapStone.java:

- Create a new file named `TaskServiceCapStone.java`.

- Open the file and add the following code:

package taskservice;

import java.util.List;

class TaskServiceCapStone {

private TaskLinkedList taskList;

private TaskPriorityQueue taskQueue;

public TaskServiceCapStone() {

taskList = new TaskLinkedList();

taskQueue = new TaskPriorityQueue();

}

public void addTask(Task task) {

taskList.add(task);

taskQueue.addTask(task);

}

public boolean delete

Task(String id) {

Task task = taskList.findById(id);

if (task != null) {

taskQueue.removeTask(task);

return taskList.delete(id);

}

return false;

}

public Task findTaskById(String id) {

return taskList.findById(id);

}

public Task getHighestPriorityTask() {

return taskQueue.getHighestPriorityTask();

}

public List<Task> getAllTasksSortedByName() {

List<Task> tasks = taskList.toList();

TaskSorter.mergeSort(tasks);

return tasks;

}

public int findTaskIndexById(String id) {

List<Task> tasks = taskList.toList();

TaskSorter.mergeSort(tasks); // Ensure the list is sorted before binary search

return TaskBinarySearch.binarySearch(tasks, id);

}

}

7. Main.java:

- Create a new file named `Main.java`.

- Open the file and add the following code:

package taskservice;

import java.util.List;

public class Main {

public static void main(String[] args) {

TaskServiceCapStone taskService = new TaskServiceCapStone();

// Generate and add 100 tasks

for (int i = 1; i <= 100; i++) {

String id = String.format("%03d", i);

String name = "Task " + i;

String description = "Description of Task " + i;

taskService.addTask(new Task(id, name, description));

}

// Find task by ID

Task task = taskService.findTaskById("050");

System.out.println("Find Task by ID 050:");

System.out.println(task);

// Get all tasks sorted by name

List<Task> sortedTasks = taskService.getAllTasksSortedByName();

System.out.println("All Tasks Sorted by Name:");

for (Task t : sortedTasks) {

System.out.println(t);

}

// Get highest priority task

Task highestPriorityTask = taskService.getHighestPriorityTask();

System.out.println("Highest Priority Task:");

System.out.println(highestPriorityTask);

// Delete a task

boolean isDeleted = taskService.deleteTask("001");

System.out.println("Task with ID 001 deleted: " + isDeleted);

// Find task index by ID

int index = taskService.findTaskIndexById("075");

System.out.println("Index of Task with ID 075: " + index);

}

}

Step 3: Compile the Code

1. Ensure you are in the `taskservice` directory:

2. Compile all Java files:

Step 4: Run the Application

1. Run the `Main` class to see the output:

Enhancements

1. Implement a Custom Linked List for Task Storage:

- The `TaskLinkedList` class manages a dynamic list of tasks, allowing efficient insertion and deletion operations.

2. Implement Binary Search for Efficient Searching:

- The `TaskBinarySearch` class uses binary search to efficiently find tasks by their ID, given that the tasks are sorted.

3. Implement Sorting Algorithm:

- The `TaskSorter` class implements the merge sort algorithm to sort tasks by their names, ensuring efficient and stable sorting.

4. Use a Priority Queue for Task Management:

- The `TaskPriorityQueue` class manages tasks based on their priority (task ID), ensuring that tasks with the highest priority are processed first.

Below is the output:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Test data to demonstrate the enhancements. 50 – 100 records**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated