

## Program Write-Up Document

Student Name: Justin Akwuba

Date: 5/2/2024

Course: CIDS 235

Assignment: Todo List Application with Inheritance, Aggregation, and Composition in Java

<https://codehs.com/sandbox/id/java-main-mhMqmf>

### **Introduction**

This document shows a basic To-Do List program written in Java that shows object-oriented ideas like composition, inheritance, and aggregation. Users can make different kinds of tasks (RegularTask, RecurringTask, and ImportantTask), add them to a to-do list, delete tasks, and see the list of tasks.

### **UML Diagram**

@startuml

```
class Task {  
    - name: String  
    - description: String  
    - dueDate: Date  
    - taskStatus: boolean  
}
```

```
class RegularTask {  
    - priority: int  
}
```

```
class RecurringTask {  
    - recurrencePattern: String  
}
```

```
class ImportantTask {  
    - isUrgent: boolean  
}
```

```
}
```

```
class TodoList {  
  - tasks: ArrayList<Task>  
  + addTask(Task): void  
  + deleteTask(Task): void  
  + displayTasks(): void  
}
```

```
class User {  
  - todoList: TodoList  
  + addTask(Task): void  
  + deleteTask(Task): void  
  + displayTodoList(): void  
}
```

```
Task <|-- RegularTask  
Task <|-- RecurringTask  
Task <|-- ImportantTask
```

```
User "1" -- "1" TodoList
```

## **Implementation Details**

Include snippets demonstrating key features of the implemented Todo List application:

Making different kinds of tasks: RegularTask, RecurringTask, and ImportantTask objects are created with their own name, description, due date, and other properties that are unique to that type of task (like priority, recurrence pattern, and urgency).

Adding, removing, and showing tasks in the to-do list:

The addTask method of the TodoList class is used to add tasks to the list.

The deleteTask method of the TodoList class is used to remove tasks from the list of things to do.

The TodoList class's displayTasks method shows the list of things that need to be done.

Showing the user's list of things to do before and after actions:

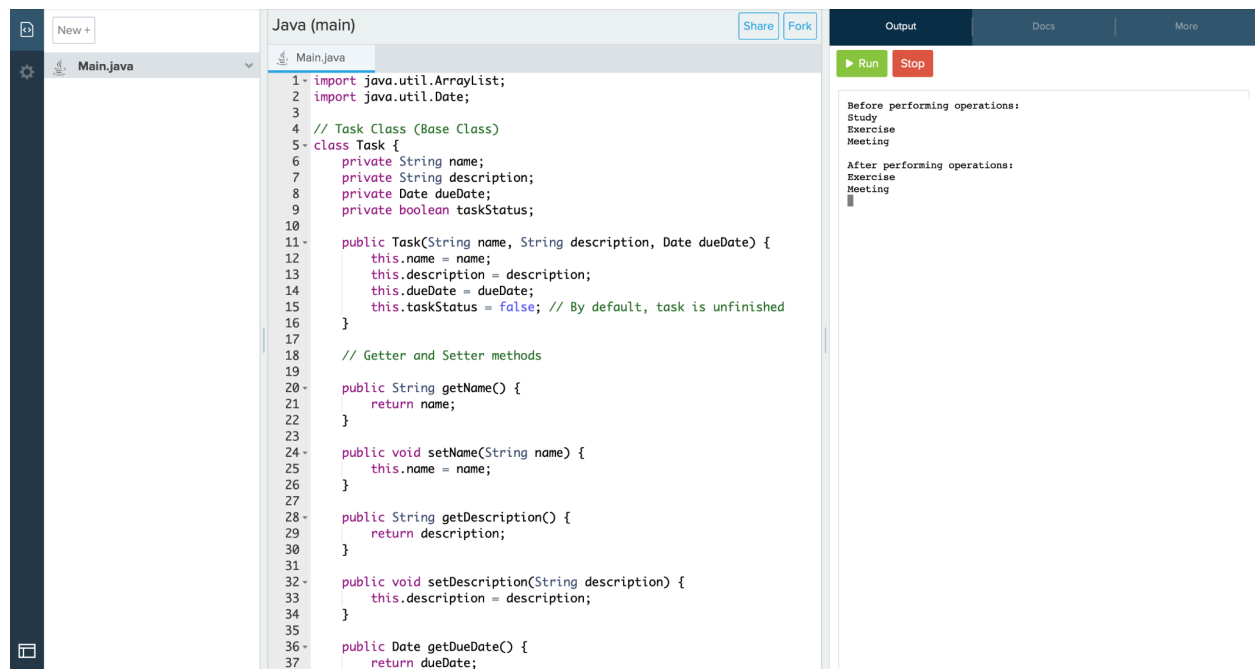
Before any actions are taken, the user's to-do list is shown.

It is possible to do things like add and delete tasks.

The user's list of things to do is shown again after actions are taken to show changes.

.

## Screenshots



The screenshot displays a Java IDE interface. The central pane shows the code for `Main.java` under the `Java (main)` project. The code defines a `Task` class with attributes `name`, `description`, `dueDate`, and `taskStatus`. It includes a constructor and getter/setter methods. The right-hand pane shows the `Output` window, which contains the following text:

```
Before performing operations:
Study
Exercise
Meeting

After performing operations:
Exercise
Meeting
```

## Extra

[Discuss any additional features or enhancements implemented beyond the assignment requirements.]

## **Discussion**

Object-oriented ideas like inheritance, aggregation, and composition help make software systems like the Todo List app modular, flexible, and easy to maintain. They encourage code reuse, encapsulation, and abstraction, which makes the application better designed, scalable, and easy to maintain.

## **Conclusion**

Creating the To-Do List app taught me a lot about object-oriented design concepts like composition, inheritance, and aggregation. We learned how to use these ideas to make software solutions that are modular, flexible, and easy to maintain, while also dealing with problems that come up when making design decisions and keeping track of dependencies.

## **Appendix**

```
import java.util.ArrayList;
import java.util.Date;

// Task Class (Base Class)
class Task {
    private String name;
    private String description;
    private Date dueDate;
    private boolean taskStatus;

    public Task(String name, String description, Date dueDate) {
        this.name = name;
        this.description = description;
        this.dueDate = dueDate;
        this.taskStatus = false; // By default, task is unfinished
    }

    // Getter and Setter methods

    public String getName() {
        return name;
    }

    public void setName(String name) {
```

```

        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Date getDueDate() {
        return dueDate;
    }

    public void setDueDate(Date dueDate) {
        this.dueDate = dueDate;
    }

    public boolean isTaskStatus() {
        return taskStatus;
    }

    public void setTaskStatus(boolean taskStatus) {
        this.taskStatus = taskStatus;
    }

    // Other methods can be added as needed
}

// RegularTask Class (Subclass of Task)
class RegularTask extends Task {
    private int priority;

    public RegularTask(String name, String description, Date dueDate, int priority) {
        super(name, description, dueDate);
        this.priority = priority;
    }
}

```

```

    // Getter and Setter for priority
}

// RecurringTask Class (Subclass of Task)
class RecurringTask extends Task {
    private String recurrencePattern;

    public RecurringTask(String name, String description, Date dueDate, String
recurrencePattern) {
        super(name, description, dueDate);
        this.recurrencePattern = recurrencePattern;
    }

    // Getter and Setter for recurrencePattern
}

// ImportantTask Class (Subclass of Task)
class ImportantTask extends Task {
    private boolean isUrgent;

    public ImportantTask(String name, String description, Date dueDate, boolean
isUrgent) {
        super(name, description, dueDate);
        this.isUrgent = isUrgent;
    }

    // Getter and Setter for isUrgent
}

// TodoList Class (Using Aggregation)
class TodoList {
    private ArrayList<Task> tasks;

    public TodoList() {
        tasks = new ArrayList<>();
    }

    public void addTask(Task task) {
        tasks.add(task);
    }
}

```

```

    }

    public void deleteTask(Task task) {
        tasks.remove(task);
    }

    public void displayTasks() {
        for (Task task : tasks) {
            System.out.println(task.getName());
        }
    }
}

// User Class (Associated with TodoList using Composition)
class User {
    private TodoList todoList;

    public User() {
        todoList = new TodoList();
    }

    public void addTask(Task task) {
        todoList.addTask(task);
    }

    public void deleteTask(Task task) {
        todoList.deleteTask(task);
    }

    public void displayTodoList() {
        todoList.displayTasks();
    }
}

// Main Class
public class Main {
    public static void main(String[] args) {
        // Creating instances of different task types

```

```
        RegularTask regularTask = new RegularTask("Study", "Study for exam", new
Date(), 2);
        RecurringTask recurringTask = new RecurringTask("Exercise", "Daily exercise
routine", new Date(), "Daily");
        ImportantTask importantTask = new ImportantTask("Meeting", "Client
meeting", new Date(), true);

        // Adding tasks to user's todo list
        User user = new User();
        user.addTask(regularTask);
        user.addTask(recurringTask);
        user.addTask(importantTask);

        // Displaying user's todo list
        System.out.println("Before performing operations:");
        user.displayTodoList();

        // Performing operations
        user.deleteTask(regularTask);

        // Displaying user's todo list after operations
        System.out.println("\nAfter performing operations:");
        user.displayTodoList();
    }
}
```