**"Employee Task Tracker System"**

**🎯 Objective: Design a console-based system to track tasks assigned to employees in a company using core Java principles.**

**🔧 Topics Covered**

| **Java Concept** | **Application in Assignment** |
| --- | --- |
| Java Fundamentals | Classes, methods, OOP |
| Exception Handling and Generics | Custom exceptions, generic task manager |
| Java Collections | Lists, Maps |
| Java Multithreading | Background status updater |
| Java Advanced Concepts | Comparable, Stream API, lambda expressions |

**💼 Use Case**

The system helps an admin assign tasks to employees and track their completion. Tasks can have due dates, priorities, and statuses. A background process checks for overdue tasks and flags them.

**📌 Assignment Requirements and Guided Questions**

**🔹 1. Java Fundamentals**

**Question:**  
Create these main classes:

* Employee: id, name, department
* Task: id, description, status (Pending, In Progress, Completed), dueDate, priority
* TaskManager: manages tasks for all employees

**Hint:**  
Use object-oriented principles to encapsulate data. Include toString() methods for displaying details.

**🔹 2. Exception Handling and Generics**

**Question:**

* Create a generic TaskRepository<T> class that can store any kind of task.
* Create a custom exception TaskNotFoundException that is thrown if someone tries to update a non-existing task.

**Hint:**  
Use Java Generics to make the repository reusable:

public class TaskRepository<T> {

private List<T> taskList = new ArrayList<>();

// add, remove, find methods

}

**🔹 3. Java Collections**

**Question:**

* Use a HashMap<Employee, List<Task>> to store assigned tasks per employee.
* Allow sorting tasks by priority or due date.
* Implement search to filter tasks by keyword in description.

**Hint:**  
Use Collections.sort() or Stream API:

tasks.stream()

.filter(t -> t.getDescription().contains("urgent"))

.forEach(System.out::println);

**🔹 4. Java Multithreading**

**Question:**  
Create a background thread called TaskMonitor that runs every minute. It checks:

* Which tasks are overdue (based on dueDate)
* Logs the list of overdue tasks to the console

**Hint:**

class TaskMonitor extends Thread {

public void run() {

while (true) {

// check tasks

Thread.sleep(60000); // 1 min

}

}

}

**🔹 5. Java Advanced Concepts**

**Question:**

* Implement Comparable<Task> to sort by priority.
* Use lambdas and streams for:
  + Finding all tasks due tomorrow
  + Finding employees with more than 3 pending tasks

**Hint:**

taskList.stream()

.filter(task -> task.getDueDate().equals(LocalDate.now().plusDays(1)))

.forEach(System.out::println);

**✅ Expected Deliverables**

1. Java source code organized in packages:
   * model, service, exception, util
2. README.md with:
   * Features implemented
   * How to compile and run
   * Sample input/output
   * Explanation of how each Java concept is used