deep se
dependable evolvable pervasive software engineering group

Exercise Session – Schedule Evaluation

Federica Filippini

Politecnico di Milano

federica.filippini@polimi.it

POLITECNICO
DI MILANO

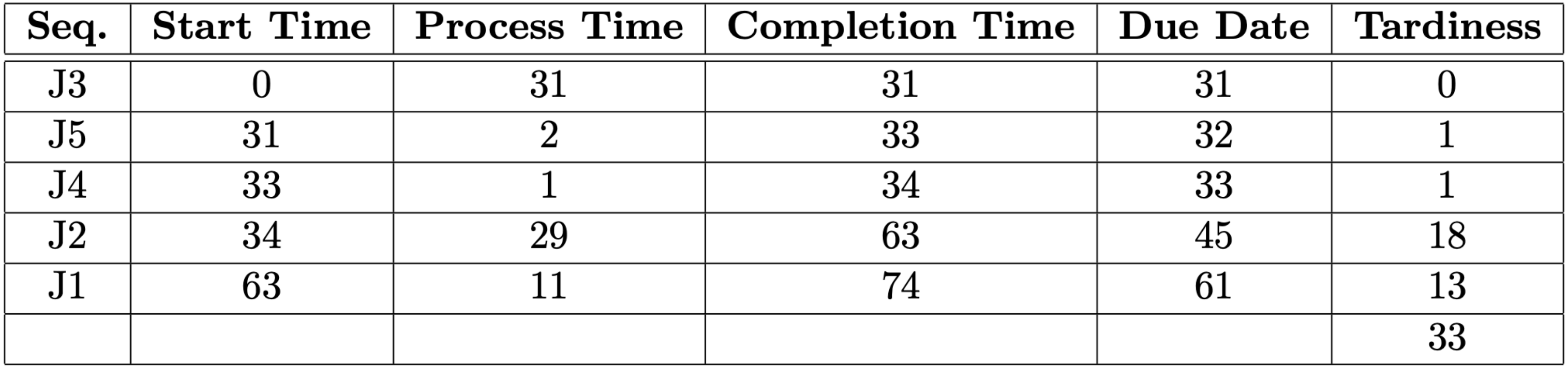
Goal

- Implement a program that, given the **scheduling** of a fixed number of jobs on a single machine, computes its **weighted tardiness**.

Job Id	Submission Time	Process Time	Due Date
J1	0	11	61
J2	0	29	45
J3	0	31	31
J4	0	1	33
J5	0	2	32

- $$\texttt{tardiness} = \begin{cases} \texttt{0}, \text{ if } \texttt{completion\_time < due\_date} \\ \\ \texttt{completion\_time - due\_date}, \text{ otherwise} \end{cases}$$

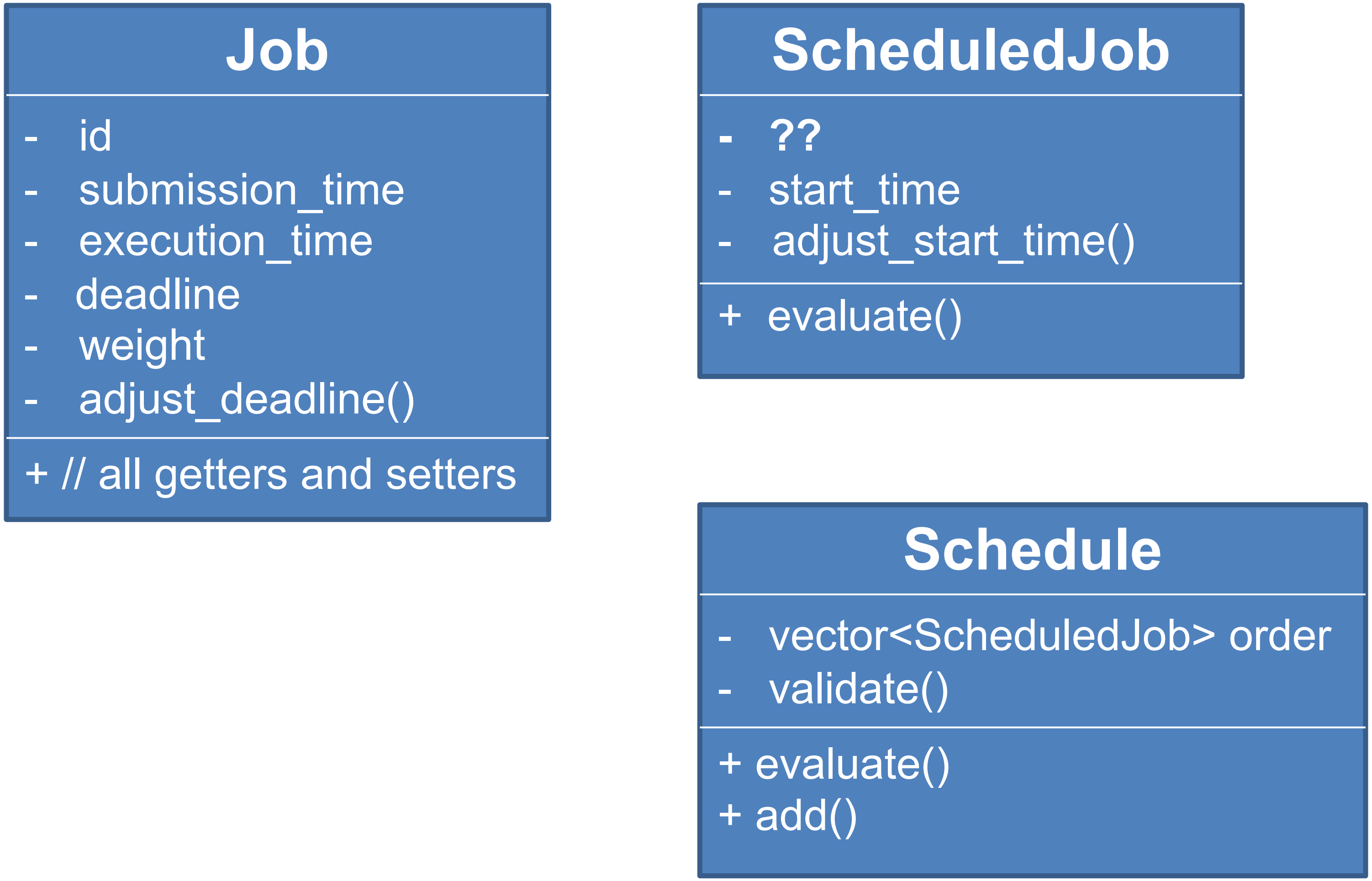
Example:



Seq.	Start Time	Process Time	Completion Time	Due Date	Tardiness
J3	0	31	31	31	0
J5	31	2	33	32	1
J4	33	1	34	33	1
J2	34	29	63	45	18
J1	63	11	74	61	13
					33

- **Assumption (1):** all the tasks must be executed **sequentially**, i.e., it is not possible to start a job until the previous one has been completed.
- **Assumption (2)**: we want to minimize code replication **and memory usage**.

Code Structure



Job

- id
- submission_time
- execution_time
- deadline
- weight
- adjust_deadline()

+ // all getters and setters

ScheduledJob

- ??
- start_time
- adjust_start_time()

+ evaluate()

Schedule

- vector<ScheduledJob> order
- validate()

+ evaluate()
+ add()

Required methods

- **`ScheduledJob::adjust_start_time()`**, that guarantees that `start_time` and `submission_time` of the current job are compatible
- **`ScheduledJob::set_start_time(time)`**, that sets the start time of the current job
- **`ScheduledJob::evaluate()`**, that computes the weighted tardiness of the current job
- **`Schedule::add(???)`**, that adds the given job to the schedule
- **`Schedule::validate()`**, that guarantees that the schedule is valid
- **`Schedule::evaluate()`**, that computes the weighted tardiness of the schedule