

Digital Away Day

Inputs

1. Activities File – contains
 - Activity Name
 - Activity Duration (in minutes)

Constants

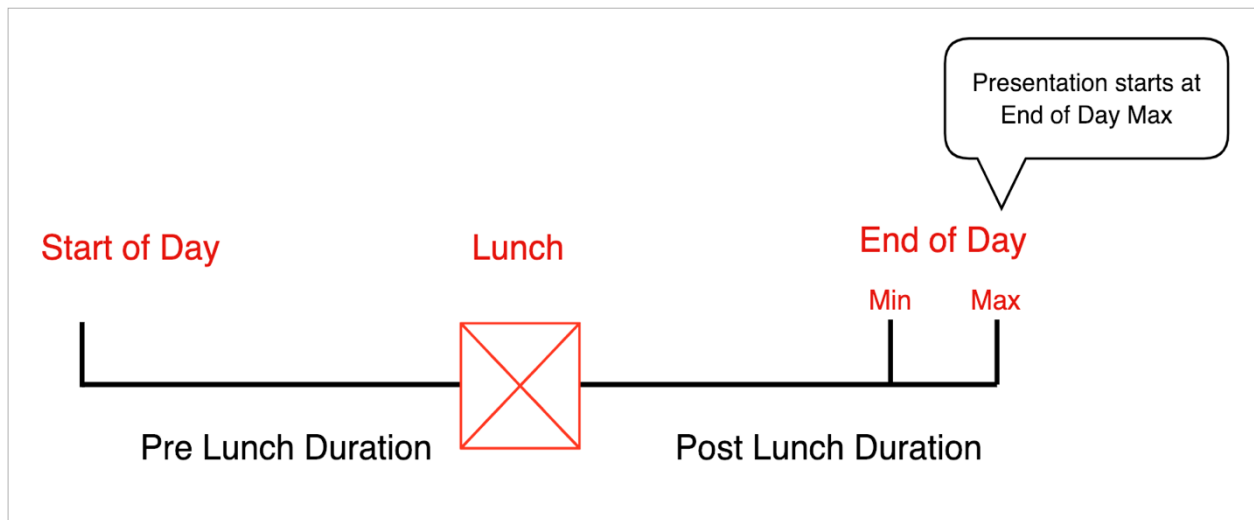
1. Start of Day = 9am
2. End of Day Min = 4pm
3. End of Day Max = 5pm
4. Lunch Start = 12pm
5. Lunch End = 1pm
6. Sprint Time = 15min

Assumptions

1. An activity cannot be performed twice (I.e), if an activity is allocated to one schedule, it cannot be present in another schedule
2. **Pre-Lunch** Activities have to start at SOD and finish exactly at Lunch Start
3. **Post-Lunch Activities** have to start exactly at Lunch End and can finish anywhere between EOD min & EOD max
4. If the activity file has an invalid entry, only that activity is discarded/ignored. Rest of the file is processed.
5. Time is assumed to be in minutes for all activities
6. Activity time has to be at least 1 min

Derived Variables

1. Pre-Lunch Duration = Lunch Start – Start of Day
2. Post Lunch Duration Min = End of Day Min – Lunch End
3. Post Lunch Duration Max = End of Day Max – Lunch End
4. Lunch Duration = Lunch End – Lunch Start
5. Maximum Possible Teams = Total Time for all activities / (Pre Lunch Duration + Post Lunch Duration Min)



Validation

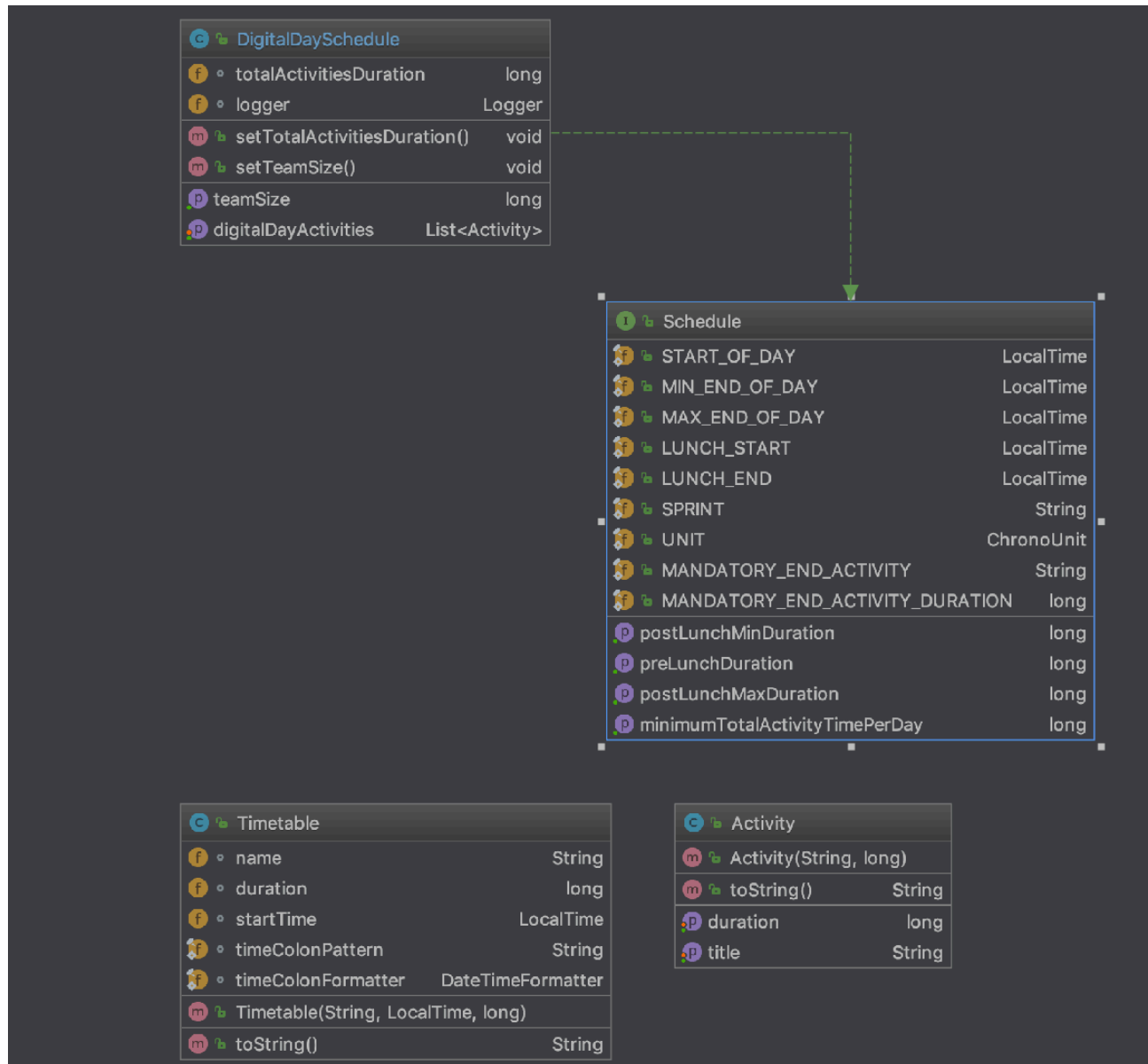
1. Check if a file is empty
2. Validate lines for the correct format using regex
3. Check and eliminate duplicates
4. If Total Time of all Activities is less than (Pre-Lunch + Post-Lunch duration) then not possible to create a schedule

How to run


















1. Git url ::
2. Download the repository 'master' branch
3. Import as a gradle project and run DigitalAwayDay (or)
4. Go to project root folder
5. Run `./gradlew build`
6. And then
7. `java -cp build/resources/main:build/classes/java/main com.deloitte.digital.DigitalAwayDay`











Solution / Implementation Details










Model

























Service

		DigitalDayScheduler	
	◦	digitalDaySchedule	DigitalDaySchedule
	◦	listOfPossibleActivities	List<List<Activity>>
	◦	cachedActivities	List<Activity>
	◦	activityList	List<Activity>
	◦	logger	Logger
		DigitalDayScheduler(DigitalDaySchedule)	
		initialiseDigitalDaySchedule(List<Activity>)	void
		formTeams()	void
		formActivityGroups(List<Activity>, long, long, long, boolean)	List<Activity>
		listOfTimetables	List<List<Timetable>>
		listOfPossibleActivities	List<List<Timetable>>

	◦	ScheduleInitialiserTest	
	◦	scheduleInitialiser	ScheduleInitialiser
		given_wrong_input_file_must_throw_exception()	void
		given_an_empty_file_must_stop_processing()	void
		given_an_incorrect_single_line_must_stop_processing()	void
		should_remove_duplicate_activities()	void

		ScheduleInitialiser	
	◦	SPRINT	String
	◦	activityTitles	Set<String>
	◦	logger	Logger
		getActivitiesFromInput(String, String)	List<Activity>
		addActivity(String)	Activity

Tests

	◦ DigitalAwayDayTest	
	◦ digitalAwayDay	DigitalAwayDay
	◦ scheduleInitialiser	ScheduleInitialiser
	◦ digitalDaySchedule	DigitalDaySchedule
	 given_wrong_input_file_must_throw_exception()	void
	 given_an_empty_file_must_stop_processing()	void
	 given_an_incorrect_single_line_must_stop_processing()	void
	 should_remove_duplicate_activities()	void
	 should_handle_duplicate_activities_with_different_duration()	void
	 should_throw_exception_if_insufficient_activities_in_list()	void
	 should_throw_exception_if_activities_do_not_fit_in_morn_slots()	void
	 should_throw_exception_if_activities_do_not_fit_in_afternoon_slots()	void
	 should_form_two_sets_of_timetables_for_correct_known_file()	void