

```

classDiagram
    class Company {
        <<singleton>>
        +name: String
        +Company()
        +getCompany(): Company
        +getBoss(): Boss
        +getManagementDepartment(): ManagementDepartment
        +getVant(): String
        +getNbEmployees(): int
        +getNbStandardDepartments(): int
        +getEmployee(id: int): Employee
        +getStandardDepartment(id: int): StandardDepartment
        +setName(name: String): Company
        +addEmployee(employee: Employee): Company
        +removeEmployee(employee: Employee): Company
        +addStandardDepartment(department: StandardDepartment): Company
        +removeStandardDepartment(department: StandardDepartment): Company
        +fire(employee: Employee): Company
        +fire(manager: Manager): Company
        +addManager(firstName: String, lastName: String): Manager
        +addEmployee(firstName: String, lastName: String): Employee
        +addStandardDepartment(name: String, activitySector: String): StandardDepartment
        +addStandardDepartment(name: String, activitySector: String, manager: Manager): StandardDepartment
        +searchEmployee(firstName: String, lastName: String): Employee[]
        +searchStandardDepartmentByActivitySector(name: String, activitySector: String): StandardDepartment[]
        +employeesToString(): String
        +toString(): String
        +save(): void
        +saveAll(): void
        +toJson(): JSONObject
        +load(): void
        +loadEmployeesAndManagers(): void
        +loadManagementDepartment(): void
        +loadStandardDepartments(): void
        +loadCompany(): void
        +loadBoss(): void
    }

    class VirtualDepartment {
        <<abstract>>
        +name: String
        +activitySector: String
        +VirtualDepartment(name: String, activitySector: String): VirtualDepartment
        +getActivitySector(): String
        +setActivitySector(activitySector: String): VirtualDepartment
        +getName(): String
        +setName(name: String): VirtualDepartment
        +toString(): String
        +toJson(): JSONObject
    }

    class ManagementDepartment {
        <<singleton>>
        +ManagementDepartment(): ManagementDepartment
        +getManagementDepartment(): ManagementDepartment
        +addManager(manager: Manager): ManagementDepartment
        +removeManager(manager: Manager): ManagementDepartment
        +getNbManagers(): int
        +getManager(id: int): Manager
        +toStringWithManagers(): String
        +toString(): String
        +save(): void
    }

    class StandardDepartment {
        +NEXT_ID: int
        +id: int
        +StandardDepartment(name: String, activitySector: String, manager: Manager): StandardDepartment
        +StandardDepartment(name: String, activitySector: String): StandardDepartment
        +StandardDepartment(name: String, activitySector: String, id: int): StandardDepartment
        +getId(): int
        +getNbEmployees(): int
        +addEmployee(employee: Employee): void
        +addEmployee(manager: Manager): StandardDepartment
        +removeEmployee(employee: Employee): void
        +setManager(manager: Manager): void
        +getManager(): Manager
        +remove(): void
        +toStringWithEmployees(): String
        +toString(): String
        +toJson(): JSONObject
        +save(): void
    }

    class Manager {
        +Manager(firstName: String, lastName: String): Manager
        +Manager(firstName: String, lastName: String, department: StandardDepartment): Manager
        +getManagementDepartment(): StandardDepartment
        +isManagerOf(department: StandardDepartment): boolean
        +setManagementDepartment(department: StandardDepartment): Manager
        +fire(): void
        +toJson(): JSONObject
    }

    class Employee {
        +NEXT_ID: int
        +id: int
        +startingHour: LocalTime
        +endingHour: LocalTime
        +Employee(firstName: String, lastName: String): Employee
        +Employee(firstName: String, lastName: String, id: int): Employee
        +setStartingHour(startingHour: LocalTime): Employee
        +setEndingHour(endingHour: LocalTime): Employee
        +getStartingHour(): LocalTime
        +getEndingHour(): LocalTime
        +getArrivingTimeAt(date: LocalDate): LocalTime
        +getLeavingTimeAt(date: LocalDate): LocalTime
        +getNbArrivedAt(date: LocalTime): boolean
        +getNbLeftAt(date: LocalTime): boolean
        +isEarlierAt(date: LocalTime): boolean
        +getDepartment(): StandardDepartment
        +setDepartment(department: StandardDepartment): Employee
        +setDepartmentToNull(): Employee
        +fire(): Employee
        +getCheckOutAt(date: LocalDate): CheckOut
        +doCheck(date: LocalDate, time: LocalTime): Employee
        +toString(): String
        +toJson(): JSONObject
        +save(): void
    }

    class Person {
        <<abstract>>
        +firstName: String
        +lastName: String
        +Person(): Person
        +Person(firstName: String, lastName: String): Person
        +getFirstName(): String
        +getLastName(): String
        +setFirstName(firstName: String): Person
        +setLastName(lastName: String): Person
        +toString(): String
        +toJson(): JSONObject
    }

    class Boss {
        <<singleton>>
        +bossInstance: Boss
        +Boss(): Boss
        +getBoss(): Boss
        +toString(): String
        +save(): void
    }

    class CheckOut {
        +totalCheckInPerDay: HashMap<LocalDate, Integer>
        +totalCheckOutPerDay: HashMap<LocalDate, Integer>
        +arrivalAt: LocalTime
        +date: LocalDate
        +CheckOut(employee: Employee, date: LocalDate): CheckOut
        +CheckOut(employee: Employee, date: LocalDate, time: LocalTime): CheckOut
        +CheckOut(employee: Employee): CheckOut
        +getTotalCheckInAt(date: LocalDate): int
        +getTotalCheckOutAt(date: LocalDate): int
        +getTotalCheckInAt(date: LocalDate): int
        +getTotalCheckOutAt(date: LocalDate): int
        +getEmployee(): Employee
        +getArrivalAt(): LocalTime
        +getArrAt(): LocalTime
        +getDate(): LocalDate
        +setArrivalAt(date: LocalTime): void
        +setArrAt(date: LocalTime): void
        +check(time: LocalTime): void
        +check(time: LocalTime): void
        +checkOut(time: LocalTime): void
        +checkOut(time: LocalTime): void
        +roundTimeToNearestQuarter(time: LocalTime): LocalTime
        +toString(): String
        +toJson(): JSONObject
    }

    Company "1" --> "1..*" StandardDepartment
    Company "1" --> "1..*" VirtualDepartment
    Company "1" --> "1..*" ManagementDepartment
    Company "1" --> "1..*" Employee
    VirtualDepartment "1" --> "1..*" StandardDepartment
    ManagementDepartment "1" --> "1" StandardDepartment
    Manager "1" --> "1..*" StandardDepartment
    Manager "1" --> "1..*" Employee : manages
    Employee "1" --> "1..*" Person
    Employee "1" --> "0..*" CheckOut : check in/out
    Boss "1" --> "1..*" Employee
    
```

Explications du diagramme :

Les classes *Company*, *Boss* et *ManagementDepartment* implémentent le patron de conception *Singleton*, permettant d'assurer la cohérence du programme : Il n'y a qu'une seule compagnie, qu'un seul boss et un seul département de management.

Les classes *StandardDepartment*, représentant les départements de travail, et *ManagementDepartment* héritent toutes deux de *VirtualDepartment*. La raison pour laquelle *ManagementDepartment* n'hérite pas de *StandardDepartment* est que l'on ne peut pas voir le département de management comme une les autres départements. Il ne faut pas pouvoir ajouter des employés, en modifier le manager qui est le boss de la compagnie.

La classe *Manager* hérite de *Employee* car se sont des employés, travaillant dans un département, mais qui sont en plus le manager de celui ci.

Chaque instance de la classe *Employee* possède une *HashMap<LocalDate, CheckInOut>*, ce qui permet d'associer une instance de *CheckInOut* à chaque employé par jour.