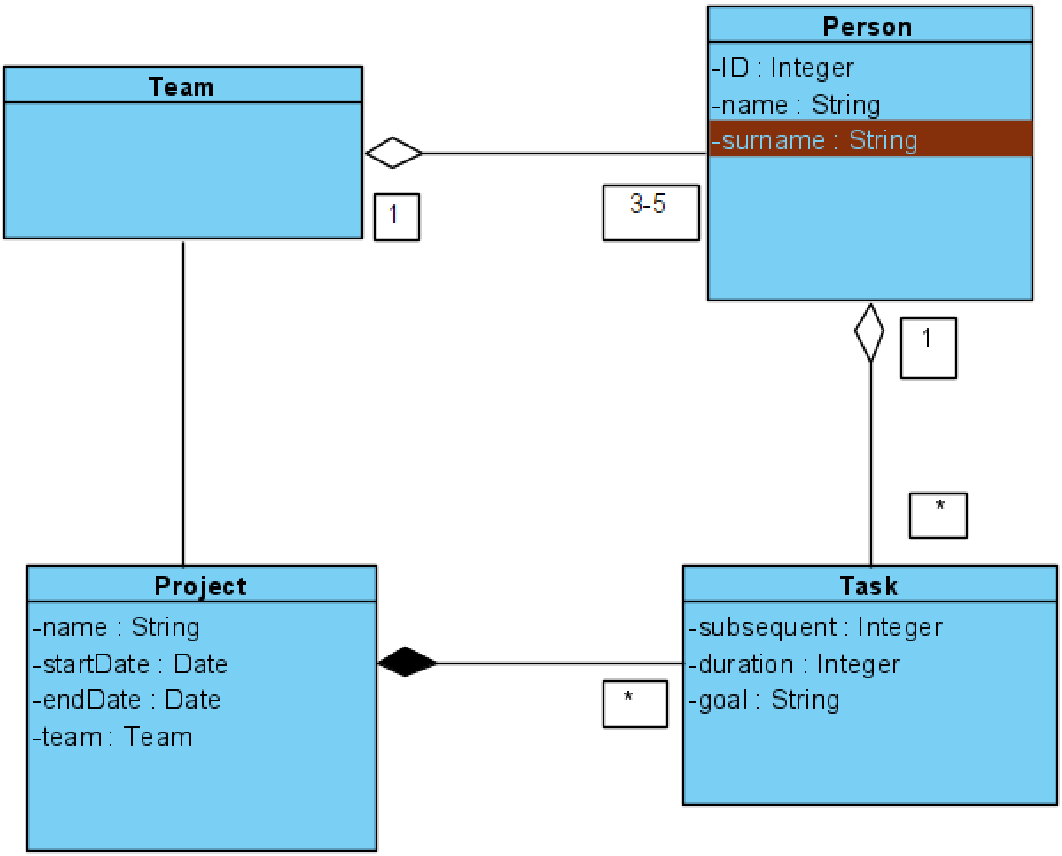
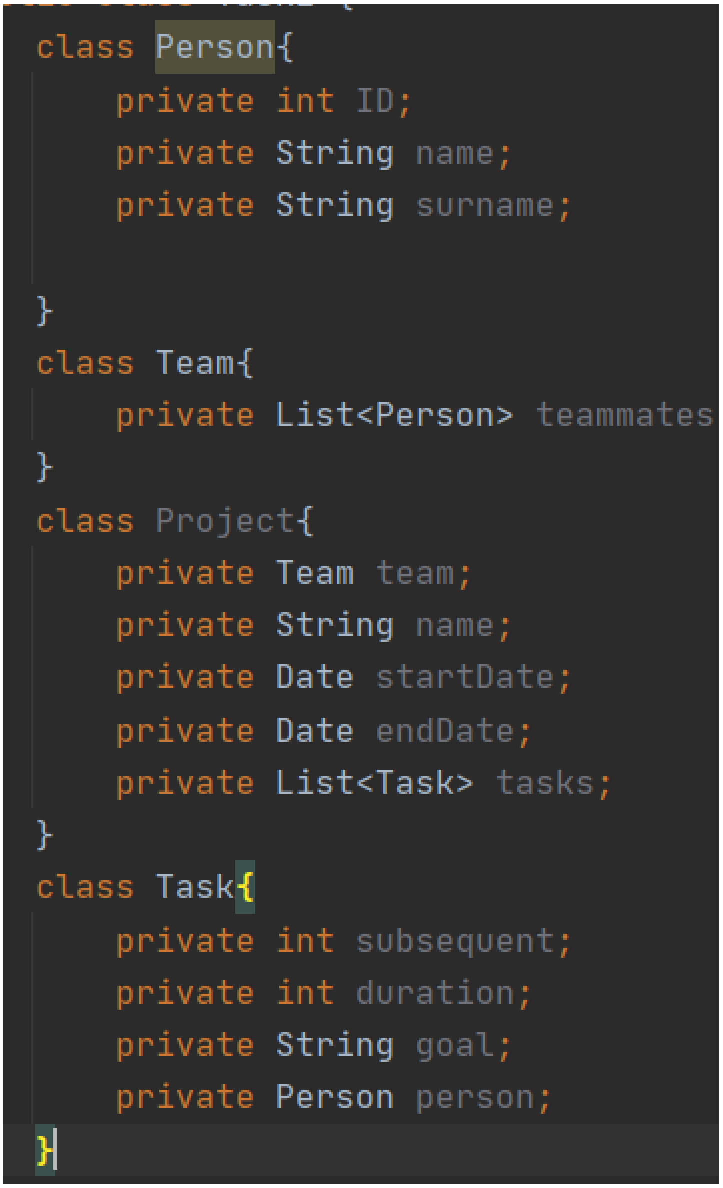
List 7

Task 1

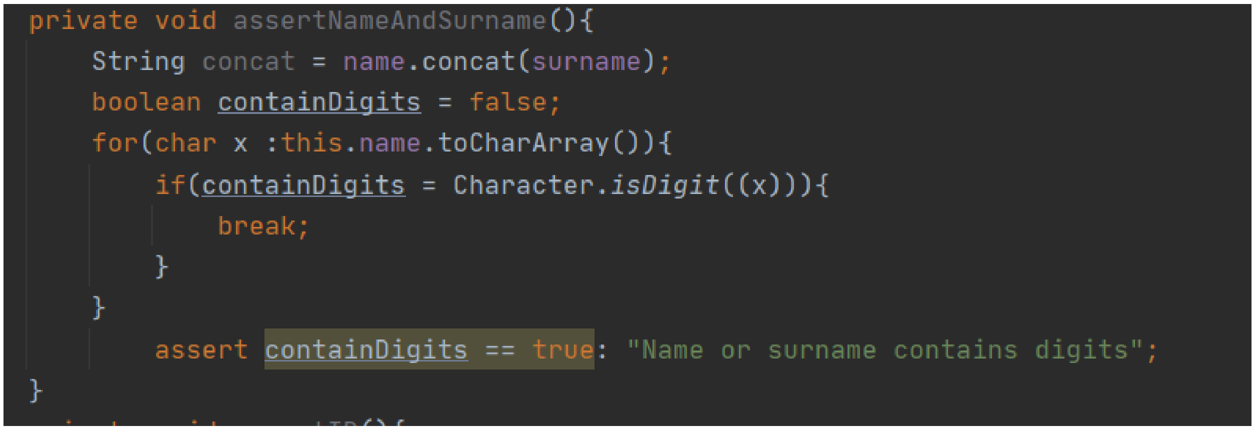
Class Diagram:





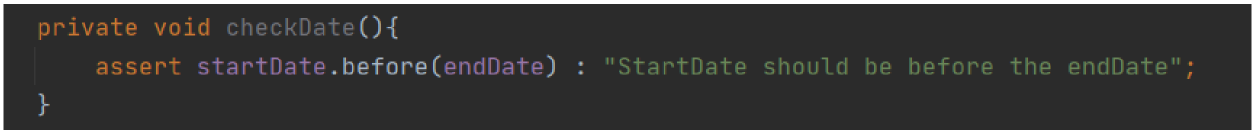
```
class Person{
        private int ID;
        private String name;
        private String surname;

}
class Team{
        private List<Person> teammates
}
class Project{
        private Team team;
        private String name;
        private Date startDate;
        private Date endDate;
        private List<Task> tasks;
}
class Task{
        private int subsequent;
        private int duration;
        private String goal;
        private Person person;
}
```

1. Name and surname shouldn’t contain any digits



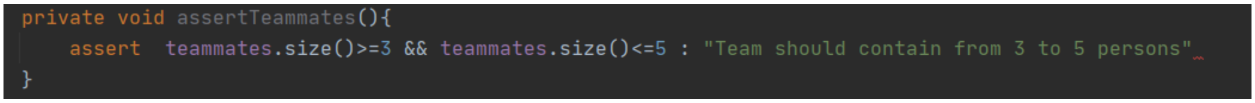
```java
private void assertNameAndSurname(){
    String concat = name.concat(surname);
    boolean containDigits = false;
    for(char x :this.name.toCharArray()){
        if(containDigits = Character.isDigit((x))){
            break;
        }
    }
        assert containDigits == true: "Name or surname contains digits";
}
```

2. EndDate should be later than startDate



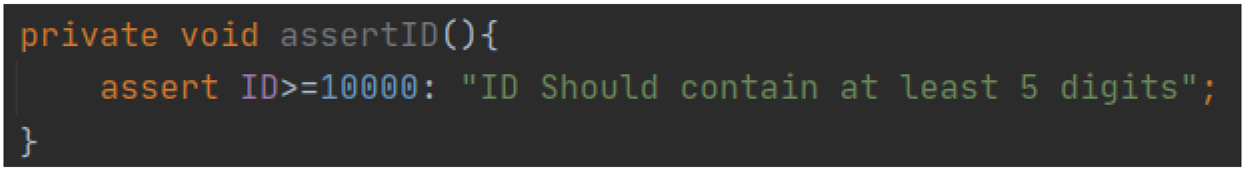
```java
private void checkDate(){
    assert startDate.before(endDate) : "StartDate should be before the endDate";
}
```

3. Number of teammates should be between 3 and 5



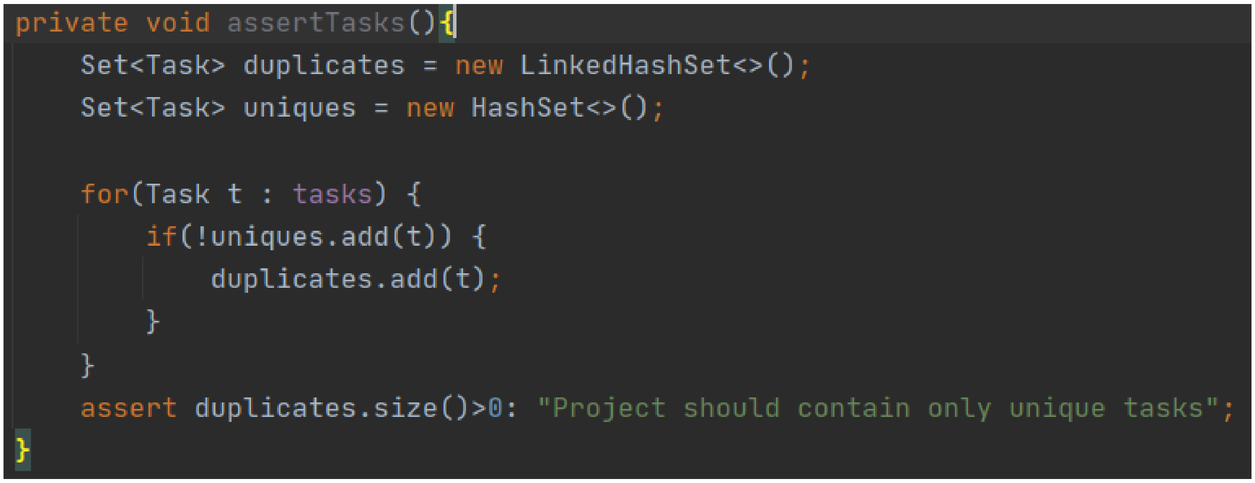
```java
private void assertTeammates(){
    assert  teammates.size()>=3 && teammates.size()<=5 : "Team should contain from 3 to 5 persons"
}
```

4. Unique identifier should be positive and contain at least 5 digits



```java
private void assertID(){
    assert ID>=10000: "ID Should contain at least 5 digits";
}
```

5. Project cannot contain similar tasks



```java
private void assertTasks(){
    Set<Task> duplicates = new LinkedHashSet<>();
    Set<Task> uniques = new HashSet<>();

    for(Task t : tasks) {
        if(!uniques.add(t)) {
            duplicates.add(t);
        }
    }
    assert duplicates.size()>0: "Project should contain only unique tasks";
}
```