1. Модели:

- **Company**

private String id;

private List<User> users;

- **User** :

private String id;

private String name;

private Title title; ( BOSS, MANAGER, DEVELOPER)

- **Boss**:

private List<String> managers;

private Company company;

private List<String> tasks;

- **Manager**:

private Unit unit; (JAVA, KOTLIN)

private List<String> developers;

private List<Task> tasks;

- **JavaDeveloper**:

private Task task;

- **KotlinDeveloper:**

private Task task;

! Для разработчиков интерфэйс **Developer**:

void develop();

Unit getUnit();

void setTask(Task task);

Task getTask();

- **Task**:

private String id;

private String content;

private TaskStatus status; ( CREATED, IN\_PROGRESS, DONE)

private String path;

1. **!Обратите внимание что бос хранит id мэнэджеров, а мэнэджеры хранят id разработчиков, но компания хранит всех юзеров. При получении определенных юзеров нужно кастить к необходимому типу( например: .map(Developer.class::cast) для stream или ((Developer) user) .doSth() )**
2. Boss должен уметь добавлять:

- users,

- user,

- при создании добавлять себя в новосозданную кампанию как юзера)

- **assignTasks()** – это значит что в конструкторе будет добавлено 2 задачи ("JAVA\_task1.txt", "KOTLIN\_task1.txt") **предварительно созданные в корне проекта с произвольным текстом внутри** в зависимости от префекса таска должна быть добавлена соответствющему мэнеджеру который в свою очередь создаст новую таску, с загруженным из файла контентом, и заосайнит ее на девелопера в своем юните если у девелопера нет задачи, если девелоперов без задач нет обработать exception.

1. Разработчик имитирует работу над задачей 5 секунд
2. В клиентском классе boss говорит мэнэджерам что бы разработчики у которых есть заосайненные таски приступили к работе. **!Каждый мэнэджер выполняет работу по запуску задач в отдельном потоке т.е. найти своих разработчиков и сказать им приступить к работе!**