**ООП Описание проекта - Информационная система исследовательского университета**

Это большой проект, помните, у вас в команде 8 человек. Вы не единственный лидер команды :)

Проект включает в себя различные виды задач, ответственность лидера команды - эффективно распределять задания после тщательного изучения человеческих ресурсов.

Это часть вашей удивительной специальности! Удачи!

**Я выберу 3 ЛУЧШИХ проекта. Призы будут объявлены позже.**

Ваш проект - система исследовательского университета. Вам следует создать классы (суперклассы, подклассы, абстрактные классы), интерфейсы, перечисления, собственные исключения, шаблоны (которые будут изучены на следующей неделе) и т. д. - все техники, которые мы изучили. Прежде чем приступить к кодированию, вам нужно разработать систему - создать архитектуру с использованием **Use Case** и **UML class Diagrams**.

Ваш проект оценивается в 30 баллов - 10 баллов на АТТ2 **и** (если вы успешно защитите лабораторную работу 3 с оценкой 4,0 или выше) 20 баллов на итоговом экзамене (дополнительные 20 баллов - устный экзамен. Лидеры команд автоматически получают 20 баллов за эту часть).

**Распределение баллов по Аттестации 2:**

Lab3 - 5 баллов

Очки за участие - 5 баллов

Итоговый - 10 баллов

Часть проекта A (Use Case & UML) - 5 баллов

Часть проекта B (часть реализованных классов, не менее 10) - 5 баллов

**3 части проекта:**

⚒ **A. Диаграммы**

используйте диаграммы прецедентов и классов, сдача: 14 неделя Используйте TopCoderUmlTool (<https://github.com/topcoder-platform/topcoder-UML-Tool>) или GenMyModel (<https://www.genmymodel.com/products/>) или любой другой инструмент, который вам нравится, но единственное требование - быть способным преобразовать вашу диаграмму классов в код. Лучше использовать один из этих вариантов, они бесплатны и имеют все, что нам нужно. Для установки Topcoder некоторым пользователям Mac может потребоваться отключить настройки безопасности, инструкции можно найти здесь: https://www.youtube.com/watch?v=ZH8\_XHzkKD4&ab\_channel=kaboratech или здесь: https://www.youtube.com/watch?v=ZH8\_XHzkKD4&ab\_channel=kaboratech.

После завершения создайте исходный код из вашей диаграммы классов и заполните классы реализацией методов.

Не забывайте о обратной инженерии - отражении изменений в ваших диаграммах.

B**. Модели (Классы)**

Покажите черновик ваших классов в течение 15 недели (время практики), окончательную версию - на итоговом экзамене.

**С. Демо (тестовый класс в консоли)**

Должно быть представлено во время итогового экзамена вместе с законченными и скорректированными частями A, B.

**На итоговом экзамене:**

ПРЕДОСТАВЬТЕ СВОЙ ОТЧЕТ (только pdf) и ПРОЕКТ (только zip, с документацией) и ПРЕЗЕНТАЦИЮ (pdf) В КОММЕНТАРИЯХ ПОД ПОСТОМ ТЕЛЕГРАММ (вы увидите его позже).

Укажите фамилии и имена участников команды в сообщении. Лидер команды должен отправить мне в личном сообщении ссылку на чат в телеграмме/тимс, использованный в течение всего периода между участниками команды для обсуждения и реализации проекта.

**Отчет** должен быть полным, детальным и хорошо структурированным. Включите подробное описание ваших классов, интерфейсов и т. д. Добавьте фрагменты кода и окончательные версии диаграмм UML в ваш отчет. Кроме того, предоставьте информацию о возникших проблемах и вопросах управления проектом, скриншоты команды или телеграмм-группы.

**Презентация** должна содержать не более 3-4 слайдов (что работает/что не работает).

Для части B вам не обязательно моделировать систему в работе, вам просто нужно создать модели - классы, интерфейсы, перечисления и т. д.

**ТРЕБОВАНИЯ**

Я опускаю детали описания, потому что это не только задача программирования, но и задача ДИЗАЙНА, поэтому это зависит от вас, какие методы/поля будут у ваших классов. И в целом какие классы вы создадите :) Обязательны: User, Employee, Teacher, Manager, Student, GraduateStudent (может быть магистром или доктором), Admin, Course, Mark, Lesson, TechSupportSpecialist, Researcher, ResearchPaper, ResearchProject, News, Message. Другие классы - на ваш выбор.

**Ваша система должна поддерживать:**

• Тип урока - лекция/практика

• 3 языка - KZ, EN, RU

• Студент - бакалавр. Аспирант - магистр/доктор

• У учителя должен быть метод отправки жалобы о студенте(ах) декану с уровнем срочности LOW, MEDIUM, HIGH

• У всех магистров и докторов есть научный руководитель, который является исследователем. Если человек с h-индексом < 3 назначен научным руководителем, необходимо вызвать исключение.

• Может быть несколько инструкторов на курсе

• В системе учителя и студенты МОГУТ быть исследователями. Эти учителя, которые профессора и аспиранты, а также студенты МОГУТ быть исследователями. Однако студенты-бакалавры и учителя, которые не являются профессорами (например, репетиторы, старшие лекторы и т. д.), также могут быть исследователями. И может быть сотрудник, который не является ни учителем, ни студентом, но он исследователь. Исследователь имеет проект(ы) по исследованию, исследовательские статьи (также объект!) и т. д.

• Поля для ResearchPaper могут быть выбраны здесь - https://ieeexplore.ieee.org/document/9766691. Возьмите 5-10 важных, цитаты, название, авторы, журнал, страницы, дата, doi и т. д.

• Исследователь должен иметь метод PrintPapers(Comparator c), который печатает его исследовательские статьи в отсортированном порядке, заданном компаратором - по дате публикации или по цитированию или по длине статьи (используйте страницы).

• Исследователь должен иметь метод для расчета h-индекса.

• Система должна поддерживать печать исследовательских статей всех исследователей университета, также отсортированных по дате публикации, цитированию или длине статьи.

• Система должна поддерживать печать наиболее цитируемого исследователя школы, года (среди всех школ).

• У ResearchPaper должен быть метод String getCitation(Format f), где формат может быть либо 'Plain Text', либо 'Bibtex'.

• Чтобы увидеть их, пожалуйста, перейдите по ссылке выше и нажмите кнопку 'Cite this'.

• ResearchProject имеет тему, опубликованные статьи и участника проекта. Если кто-то, кто не является исследователем, пытается присоединиться к исследовательскому проекту, должно быть вызвано пользовательское исключение.

• Генерация отчетов (оценки, простая статистика)

• Служебные записки (рабочие сообщения)

• Для техподдержки сотрудники технической поддержки должны иметь возможность видеть новые заказы, принимать/отклонять их.

• После принятия заказ больше не является "новым", можно просматривать выполненные и принятые (но еще не выполненные) заказы (например, починить проектор или принтер).

• Дипломные проекты. Дипломные проекты магистров и докторов должны иметь список опубликованных исследовательских статей.

• Основные, дополнительные, факультативные курсы. Обратите внимание, что для студента SITE какой-то основной курс из школы нефти и газа может быть факультативным.

• Новости с комментариями. Новости с темой "Исследование" должны иметь приоритет в порядке (закреплены). Когда какой-то исследователь публикует статью, должно быть объявление. Также не забывайте автоматически генерировать новости о самом цитируемом исследователе в университете.

----Поскольку это исследовательский университет, у него есть собственные университетские исследовательские журналы. Все

пользователи в университете (не только исследователи) могут подписаться на какой-либо университет

журналы. Система должна уведомлять читателей, когда новая статья публикуется в

журнал, на который они подписаны. Время от времени появляются новые журналы.

Что это за шаблон?

• 4 или более шаблонов проектирования.

Данная ситуация описывает несколько дизайн-паттернов, которые могут быть использованы для разработки системы университетских журналов. Вот несколько возможных дизайн-паттернов:

Наблюдатель (Observer) паттерн:

Описывает отношение "один ко многим" между объектами, так что при изменении состояния одного объекта все зависящие от него объекты уведомляются и обновляются автоматически. В данном случае, читатели могут быть подписаны на определенные журналы, и когда в журнале появляется новая статья, все подписчики (читатели) автоматически уведомляются.

Издатель-Подписчик (Publisher-Subscriber) паттерн:

Этот паттерн также обеспечивает механизм уведомлений, но через централизованный сервис. В данном случае, университетские журналы могут действовать как издатели, а читатели - как подписчики. Когда новая статья публикуется, издатель уведомляет все подписчиков через централизованный сервис.

Фабричный метод (Factory Method) паттерн:

Может быть использован для создания экземпляров журналов. Поскольку время от времени появляются новые журналы, фабричный метод может предоставить интерфейс для создания экземпляров различных типов журналов, без необходимости указания конкретного класса.

Стратегия (Strategy) паттерн:

Если различные типы уведомлений могут быть выбраны или изменены динамически, стратегия может быть использована для определения различных стратегий уведомлений (например, электронная почта, push-уведомления и т.д.), которые могут быть применены в зависимости от предпочтений читателя.

**Общие требования:**

* **ООП стиль**
* **Использование Comparable, Comparators, equals, hashcode, toString и т. д.**
* **Работающая сериализация (думайте о хранении данных и каком-то шаблоне)**
* Любой пользователь должен получить доступ к системе через аутентификацию
* Интуитивное использование
* **Согласованность с UML**
* Не забывайте о правильном использовании перечислений. Вы можете использовать их, например, для представления званий преподавателей - репетитор, лектор, старший лектор, профессор и т. д.
* Правильное и логически последовательное использование коллекций
* Использование Java API (стандартных классов). Не изобретайте велосипед заново.
* **Документация**
* Низкая связность, высокая связанность

**Приблизительная проверочная таблица (для всего, кроме исследования):**

* **Администратор**
  + Управление пользователями (Добавление, удаление, обновление)
  + Просмотр журналов пользовательских действий
* **Преподаватель**
  + Просмотр курсов
  + Управление курсом
  + Просмотр студентов, информация о студентах
  + Выставление оценок
  + Отправка сообщений другим сотрудникам (фактически любой сотрудник может отправить сообщение любому сотруднику)
  + Отправка жалоб
* **Студент**
  + Просмотр курсов, Регистрация на курсы
  + Просмотр информации о преподавателе по конкретному курсу
  + Просмотр оценок
  + Просмотр транскрипта
  + Оценка преподавателей
  + Получение транскрипта
  + Студенческие организации (например, ОСИТ). Студент может быть членом/руководителем.
* **Менеджер**
  + Утверждение регистрации студентов, Добавление курсов для регистрации (указание для какого направления/курса предназначен курс)
  + Назначение курсов преподавателям
  + Типы менеджеров - ОР, Департаменты и т. д. (используйте перечисление).
  + Создание статистических отчетов об успеваемости.
  + Управление новостями
  + Просмотр информации о студентах и преподавателях (различные способы, например, сортировка по среднему баллу, алфавиту и т. д.)
  + Просмотр запросов от сотрудников (они должны быть подписаны деканом/ректором)
* **Исследователь - описано выше. Исследователь - это загадка. Это интерфейс? Абстрактный класс? Создается с использованием паттерна декоратора? Просто сотрудник? Разберитесь с этим. Здесь нет единственного ответа.**

**Важное замечание:**

* **Учтите детали! Например:**
  + Студенты не могут иметь более 21 кредита
  + Студенты не могут провалить более 3 раз
  + Оценка состоит из 1-й, 2-й аттестации и финальной.
  + Многое другое, действительно...

САМАЯ ВАЖНАЯ ФУНКЦИОНАЛЬНОСТЬ - регистрация на курсы, выставление оценок, исследование. Завершите это первым.

Мы будем изучать шаблоны проектирования, потоки коллекций в предстоящих лекциях.

**Бонус:** Учтите как можно больше деталей (за ценные дополнительные функции вы получите дополнительные баллы, например, генерация расписания (учитывая загрузку аудиторий, тип аудитории и т. д.), Посещаемость, Опция генерации отчетов для преподавателя (оценки), расширенный поиск с использованием регулярных выражений, стартапы, рекомендательные письма).