Домашнее задание

Задача 1

Техническое задание на систему "Умный дом"

1. Введение

1.1. Назначение

Разработка объектно-ориентированной системы для управления умными устройствами в доме.

1.2. Область применения

Система предназначена для моделирования работы умных устройств и управления ими в рамках системы "Умный дом".

2. Базовый класс SmartDevice

2.1. Требования к полям

- `deviceId` (String) - уникальный идентификатор устройства
- `deviceName` (String) - название устройства
- `location` (String) - местоположение в доме
- `isPoweredOn` (boolean) - статус включения устройства
- `powerConsumption` (double) - потребление энергии в ваттах

2.2. Статические члены

- `totalDevices` - счетчик созданных устройств
- `MAX_POWER_CONSUMPTION` - final константа = 5000 Вт
- `getTotalDevices()` - статический метод для получения общего количества устройств

2.3. Конструкторы

1. Конструктор по умолчанию
2. Конструктор с основными параметрами (id, название, местоположение)
3. Конструктор со всеми параметрами

2.4. Методы

Общие методы:

- `getDeviceType()` - возвращает тип устройства (базовая реализация)
- `performAction()` - выполнить действие устройства (базовая реализация)
- `turnOn()` - включить устройство

- `turnOff()` - выключить устройство
- `getStatus()` - получить статус устройства

2.5. Валидация в сеттерах

- `powerConsumption` не может быть отрицательным и не больше `MAX_POWER_CONSUMPTION`
- `deviceId` не может быть null или пустым

3. Классы-наследники

3.1. Класс SmartLight (Умная лампа)

Наследование:

- Наследует от `SmartDevice`

Дополнительные поля:

- `brightness` (int) - яркость (0-100%)
- `color` (String) - цвет света
- `isColorChanging` (boolean) - поддерживает ли смену цвета

Переопределяемые методы:

- `getDeviceType()` - возвращает "Умная лампа"
- `performAction()` - включает/выключает свет

Перегруженные методы:

- `setBrightness(int level)` - установить яркость
- `setBrightness(int level, String color)` - установить яркость и цвет

Уникальные методы:

- `changeColor(String newColor)` - изменить цвет
- `dimLights()` - приглушить свет

3.2. Класс SmartThermostat (Умный термостат)

Наследование:

- Наследует от `SmartDevice`

Дополнительные поля:

- `currentTemperature` (double) - текущая температура
- `targetTemperature` (double) - целевая температура
- `mode` (String) - режим (охлаждение/обогрев)

Переопределяемые методы:

- `getDeviceType()` - возвращает "Умный термостат"
- `performAction()` - регулирует температуру

Перегруженные методы:

- `setTemperature(double temp)` - установить температуру
- `setTemperature(double temp, String mode)` - установить температуру и режим

Уникальные методы:

- `getEnergyReport()` - получить отчет по энергии
- `scheduleTemperature(double temp, int hour)` - запланировать температуру

3.3. Класс SmartSecurityCamera (Умная камера)

Наследование:

- Наследует от `SmartDevice`

Дополнительные поля:

- `isRecording` (boolean) - ведется ли запись
- `motionDetection` (boolean) - включено ли обнаружение движения
- `videoQuality` (String) - качество видео

Переопределяемые методы:

- `getDeviceType()` - возвращает "Умная камера"
- `performAction()` - начинает/останавливает запись

Перегруженные методы:

- `startRecording()` - начать запись
- `startRecording(boolean motionDetection)` - начать запись с настройкой обнаружения движения

Уникальные методы:

- `detectMotion()` - обнаружить движение
- `getLiveFeed()` - получить прямую трансляцию

4. Класс SmartHome (Умный дом)

4.1. Поля

- `homeName` (String) - название дома
- `devices` - список умных устройств

4.2. Методы

- `addDevice(SmartDevice device)` - добавить устройство
- `removeDevice(String deviceId)` - удалить устройство по ID
- `turnOnAllDevices()` - включить все устройства
- `turnOffAllDevices()` - выключить все устройства
- `getDevicesByType(String type)` - найти устройства по типу
- `calculateTotalPowerConsumption()` - посчитать общее потребление энергии
- `getDeviceStatus(String deviceId)` - получить статус конкретного устройства

5. Технические требования

5.1. Язык программирования

- Java

5.2. Принципы ООП

- Инкапсуляция
- Наследование
- Полиморфизм
- Перегрузка методов

5.3. Ограничения

- Все поля базового класса должны быть private/protected
- Использование статических переменных и методов
- Валидация входных данных
- Обработка исключительных ситуаций

Задача 2

Техническое задание на систему управления меню ресторана

1. Введение

1.1. Назначение

Разработка объектно-ориентированной системы для управления меню и заказами в ресторане.

1.2. Область применения

Система предназначена для автоматизации процессов управления меню, формирования заказов и контроля процесса приготовления блюд.

2. Базовый класс MenuItem

2.1. Требования к полям (все private)

- `itemId` (String) - уникальный идентификатор блюда
- `name` (String) - название блюда
- `price` (double) - цена
- `cookingTime` (int) - время приготовления в минутах
- `isAvailable` (boolean) - доступно ли для заказа

2.2. Статические члены

- `totalMenuItems` - счетчик созданных блюд
- `MAX_COOKING_TIME` - final константа = 120 минут
- `getTotalMenuItems()` - статический метод для получения общего количества блюд

2.3. Конструкторы

1. Конструктор по умолчанию
2. Конструктор с основными параметрами (id, название, цена)
3. Конструктор со всеми параметрами

2.4. Методы

Основные методы:

- `getItemType()` - возвращает тип блюда (базовая реализация)
- `prepare()` - приготовить блюдо (базовая реализация)
- `displayInfo()` - показать информацию о блюде
- `updatePrice(double newPrice)` - обновить цену

2.5. Валидация в сеттерах

- `price` не может быть отрицательным
- `cookingTime` от 1 до MAX_COOKING_TIME
- `itemId` не может быть null или пустым

3. Классы-наследники

3.1. Класс Appetizer (Закуска)

Наследование:

- Наследует от `MenuItem`

Дополнительные поля:

- `servingTemperature` (String) - температура подачи (холодная/горячая)
- `isVegetarian` (boolean) - вегетарианское ли
- `spiceLevel` (int) - уровень остроты (1-5)

Переопределяемые методы:

- `getItemType()` - возвращает "Закуска"
- `prepare()` - готовит закуску

Перегруженные методы:

- `displayInfo()` - базовая информация
- `displayInfo(boolean showDetails)` - подробная информация с деталями

Уникальные методы:

- `changeSpiceLevel(int level)` - изменить уровень остроты
- `getNutritionInfo()` - получить информацию о питательности

3.2. Класс MainCourse (Основное блюдо)

Наследование:

- Наследует от `MenuItem`

Дополнительные поля:

- `mainIngredient` (String) - основной ингредиент
- `sideDish` (String) - гарнир

- `calories` (int) - калории

Переопределяемые методы:

- `getItemType()` - возвращает "Основное блюдо"
- `prepare()` - готовит основное блюдо

Перегруженные методы:

- `updatePrice(double newPrice)` - обновить цену
- `updatePrice(double newPrice, String reason)` - обновить цену с указанием причины

Уникальные методы:

- `changeSideDish(String newSideDish)` - изменить гарнир
- `getCookingInstructions()` - получить инструкции по приготовлению

3.3. Класс Dessert (Десерт)

Наследование:

- Наследует от `MenuItem`

Дополнительные поля:

- `sweetnessLevel` (int) - уровень сладости (1-10)
- `containsNuts` (boolean) - содержит ли орехи
- `isGlutenFree` (boolean) - без глютена

Переопределяемые методы:

- `getItemType()` - возвращает "Десерт"
- `prepare()` - готовит десерт

Перегруженные методы:

- `prepare()` - обычное приготовление
- `prepare(boolean withExtraDecoration)` - приготовление с дополнительным украшением

Уникальные методы:

- `addTopping(String topping)` - добавить топпинг
- `checkAllergies()` - проверить наличие аллергенов

4. Класс RestaurantOrder (Заказ в ресторане)

4.1. Поля

- `orderId` (String) - номер заказа
- `tableNumber` (int) - номер стола
- `orderItems` - список блюд в заказе
- `orderStatus` (String) - статус заказа

4.2. Статические члены

- `totalOrders` - счетчик заказов
- `MAX_TABLE_NUMBER` - final константа = 50

4.3. Методы

- `addItem(MenuItem item)` - добавить блюдо в заказ
- `removeItem(String itemId)` - удалить блюдо из заказа
- `calculateTotal()` - посчитать общую сумму заказа
- `getOrderDetails()` - получить детали заказа
- `updateStatus(String newStatus)` - обновить статус заказа
- `getItemsByType(String type)` - получить блюда по типу
- `estimatePreparationTime()` - оценить общее время приготовления

5. Технические требования

5.1. Язык программирования

- Java

5.2. Принципы ООП

- Инкапсуляция (все поля private)
- Наследование
- Полиморфизм
- Перегрузка методов

5.3. Ограничения и валидация

- Контроль корректности входных данных
- Ограничение времени приготовления
- Проверка уникальности идентификаторов
- Валидация числовых значений (цены, уровни, калории)

5.4. Бизнес-логика

- Управление доступностью блюд
- Расчет общего времени приготовления заказа
- Фильтрация блюд по типам
- Контроль статусов заказов