Для всех задач необходимо дополнительно создавать класс Application, который будет содержать статический метод main.

**Задача 1**

Необходимо создать класс Human, содержащий следующие поля: lastName (String), firstName (String), birthDate (тип – Date). Конструктор оставить по умолчанию (без параметров). Реализовать геттеры и сеттеры для данных полей.

При установке полей необходимо проверить значения: Фамилия и имя должны состоять из кириллицы, первая буква – заглавная, остальные – строчные, общая длина не менее 3 символов. Дата рождения должна быть не позднее сегодняшнего дня. В случае, если поля имеют некорректные значения, их не устанавливать. Вывести соответствующее сообщение в консоль.

При получении значений поля необходимо помимо возвращения полей вывести сообщение со значением в консоль.

Для вывода сообщения в консоль реализовать отдельные методы getPrint, setCorrectPrint, setIncorrectPrint.

В методе main необходимо протестировать реализованные методы на различных входных данных.

Формат сообщений (дата должна выводиться в формате dd.MM.yyyy):

|  |  |  |
| --- | --- | --- |
| Операция | Формат | Пример |
| Получение значения lastName | Получено значение фамилии: %s | Получено значение фамилии: Иванов |
| Получение значения firstName | Получено значение имени: %s | Получено значение имени: Петр |
| Получение значения birthDate | Получено значение даты рождения: %s | Получено значение даты рождения: 12.04.1990 |
| Установка корректного значения | Установлено корректное значение фамилии|имени|даты рождения: %s | Установлено корректное значение даты рождения: 15.05.1990 |
| Установка некорректного значения | Введенное значение некорректно: %s | Введенное значение некорректно: 15.04.2025 |

**Задача 2**

Дано: В консоль вводятся строки в формате

%s %s %s

Необходимо создать класс Validator, в котором будут 4 метода. Первый метод должен валидировать формат строки (валидна, если в ней ровно 2 пробела). Если строка не соответствует формату, вывести в консоль соответствующее сообщение. Второй метод из исходной строки должен вернуть массив строк, полученный разделением по пробелам. Третий метод должен валидировать объект типа Human (валиден, если все поля инициализированы). Четвертый метод должен валидировать массив объектов типа Human (массив не валиден, если он полностью заполнен). Методы должны выводить в консоль результаты операций

Класс Human должен быть аналогичен предыдущей задаче.

В методе main создать массив типа Human размером 3. Также создать цикл, который будет считывать строки до тех пор, пока данный массив не будет полностью заполнен валидными значениями.

**Задача 3**

Необходимо создать 2 пакета (package) с именами workers и application. Класс Application поместить в пакет application. В пакете workers создать несколько классов по профессиям работников: Accountant (бухгалтер), Logistician (логист), SysAdmin (системный администратор), Cleaner (уборщик), ServiceDesk (система распределения заявок). Все классы, кроме ServiceDesk должны быть ограничены по доступу в пределах своего пакета и содержать единственный статический метод takeTicket(int ticketNumber). Класс ServiceDesk должен быть публичным и предоставлять метод createTicket(String description), где description – это текст заявки. Данный метод должен выводить на экран «Заявка <номер> была создана: <текст заявки>» и искать специалиста для данной заявки.

В каждом из классов Accountant, Logistician, SysAdmin, Cleaner в методе takeTicket вывести на экран строку вида «Заявка <номер> была принята <название профессии>». В классе ServiceDesk реализовать логику, которая по тексту заявки (нахождению ключевых слов) распределяет заявки между специалистами. Если система распределения заявок не может определить специалиста по заявке, на экран должен быть выведен текст «Специалист по выполнению данной заявки не найден». Ключевые слова, по которым происходит распределение:

|  |  |
| --- | --- |
| Специалист | Ключевые слова |
| Бухгалтер | «справка», «отчет», «бюджет» |
| Логист | «доставка», «курьер» |
| Системный администратор | «компьютер», «принтер», «сервер» |
| Уборщик | «уборка» |

В методе main проверить работу системы. Убедиться, что из метода main возможно вызвать только метод createTicket класса ServiceDesk, вызвать методы takeTicket() у специалистов напрямую из метода main невозможно. Примеры входных данных:

|  |  |
| --- | --- |
| Текст заявки | Ожидаемый результат |
| Прошу предоставить справку 2-ндфл за текущий год | Заявка 1 была создана <текст>  Заявка 1 была принята бухгалтером |
| Прошу предоставить курьера 2 сентября 2020 года | Заявка 2 была создана <текст>  Заявка 2 была принята логистом |
| Необходимо починить принтер в комнате 207 | Заявка 3 была создана <текст>  Заявка 3 была принята системным администратором |
| Прошу произвести внеплановую уборку в переговорной №1 | Заявка 4 была создана <текст>  Заявка 4 была принята уборщиком |
| Прошу выдать канцелярские принадлежности сотруднику Баранову А.С. | Заявка 5 была создана <текст>  Специалист по выполнению заявки 5 не найден |

**Задача 4**

Реализовать класс Human, который будет содержать следующие поля: String lastName, String firstName, int age, Passport passport. Реализовать класс Passport, который будет содержать два поля: String series, String number. Конструктор должен оставить без параметров. Реализовать для данных классов сеттеры и геттеры, используя паттерн Builder.

В сеттерах класса Passport необходимо провести валидацию: серия – число из 4 цифр, номер из 6 цифр. Серия и номер создаваемого паспорта должны быть уникальными (одновременно). Валидацию уникальности проводить только для тех паспортов, где заданы и серия, и номер (например, может существовать 2 паспорта с серией «1234» и номером null)

В сеттере поля passport в классе Human провести валидацию паспорта (серия и номер должны быть заполнены).

В случае валидных данных в сеттере вывести соответствующее сообщение в консоль, в случае невалидных – сообщение об ошибке.

В методе main необходимо протестировать полученное решение.