**1.**

Пришло время немного порефакторить.

**Википедия говорит:** "***Рефакторинг****(англ. refactoring) или реорганизация кода - процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение) преобразований. Поскольку каждое преобразование маленькое, программисту легче проследить за его правильностью, и в то же время вся последовательность может привести к существенной перестройке программы и улучшению её согласованности и четкости." Такими маленькими преобразованиями мы и будем заниматься. После каждого изменения следи за сохранностью работоспособности кода. Многие методы рефакторинга не однократно применены и проверены сообществом программистов. Получили свои названия. Каждое задание будет сопровождено названием рефакторинга. Ты всегда можешь прочитать более подробно о нем в книге Мартина Фаулера "****Рефакторинг: Улучшение существующего кода***".

Начнем. Внимательно ознакомься с кодом пакета **human** (человек).

**Задания:**  
1.1. Подъем поля. Подними поле children в базовый класс.  
1.2. Подъем метода. Подними **сеттер** и **геттер** для children в базовый класс.  
1.3. Инкапсуляция коллекции.  
1.3.1. Метод getChildren должен возвращать не модифицируемое представление списка children.  
1.3.2. Убери сеттер для children.  
1.3.3. Добавь методы addChild(**Human**) и removeChild(**Human**). Реализуй их логику.

**Требования:**

* •

Поле children должно быть расположено в классе Human, и не должно быть расположено в классах Teacher и Student.

* •

Сеттер и геттер для поля children должны быть расположены в классе Human, и не должны быть расположены в классах Teacher и Student.

* •

Метод getChildren в классе Human должен возвращать Collections.unmodifiableList(children).

* •

Необходимо удалить метод setChildren из класса Human.

* •

Необходимо добавить методы addChild (Human) и removeChild (Human) в класс Human, и реализовать их.

**2.**

2.1. Извлечение подкласса.  
2.1.1. Добавь класс **Soldier** в пакет **human**.  
2.1.2. Избавься от поля isSoldier.  
2.1.3. Перенеси в **Soldier** необходимые методы из **Human**.  
2.1.4. Обнови сигнатуру конструктора **Human**.  
2.2. Подъем тела конструктора.  
2.2.1. Перенеси инициализацию полей name и age в подходящее место, добавь необходимые параметры в конструктор **Human**.  
2.2.2. Добавь конструктор в класс **Soldier**.

**Требования:**

* •

Класс Soldier должен существовать в пакете human в отдельном файле. Класс Soldier должен наследоваться от класса Human.

* •

В классе Human не должно быть поля isSoldier.

* •

В класс Soldier из класса Human требуется перенести необходимые методы.

* •

Конструктор класса Human должен принимать два параметра: String name и int age, и инициализировать соответствующие поля.

* •

В классе Soldier должен быть конструктор, который принимает два параметра: String name и int age, и вызывает соответствующий конструктор суперкласса.

**3.**

3.1. Спуск поля. Спусти поле course в соответствующий класс. Сделай его **приватным**.  
3.2. Спуск метода. Спусти **геттер** для поля course в соответствующий класс.  
3.3. Извлечение интерфейса.  
3.3.1. Создай интерфейс **Alive** (**живой**) в пакете **human**.  
3.3.2. Интерфейс должен содержать метод жить live().  
3.3.3. Добавь интерфейс нужному классу.  
3.4. Свертывание иерархии. Избавься от класса **Professor**.

**Требования:**

* •

Необходимо спустить поле course в нужный класс и сделать его приватным.

* •

Необходимо спустить геттер для поля course в нужный класс.

* •

Интерфейс Alive должен существовать в пакете human.

* •

Интерфейс Alive должен содержать объявление метода жить live().

* •

Класс Human должен поддерживать интерфейс Alive.

* •

Класс Professor нужно удалить

**4.**

4.1. Замена наследования делегированием.  
4.1.1. Класс **University** не должен наследоваться от **Student**.  
4.1.2. Класс **University** должен содержать список **students**. Не забудь его инициализировать.  
4.1.3. Добавь **сеттер** и **геттер** для **students**.  
4.1.4. Университет имеет название (name) и возраст (age). Добавь необходимые поля, сеттеры и геттеры для них.  
4.2. Извлечение суперкласса.  
4.2.1. Создай класс **UniversityPerson** в пакете **human**.  
4.2.2. Перенеси в него поле university.  
4.2.3. Перенеси **сеттер** и **геттер** для поля university.  
4.2.4. Унаследуй необходимые классы от **UniversityPerson**.  
4.3. Замена простого поля объектом. Измени тип поля university на **University**.

**Требования:**

* •

Класс University не должен наследоваться от Student.

* •

В классе University необходимо создать поле name, сеттер и геттер для него.

* •

В классе University необходимо создать поле age, сеттер и геттер для него.

* •

В классе University необходимо создать приватное поле List<Student> students.

* •

В классе University необходимо создать сеттер и геттер для поля students.

* •

Необходимо создать публичный класс UniversityPerson в пакете human и унаследовать его от класса Human.

* •

Необходимо перенести поле university из классов Teacher и Student в класс UniversityPerson.

* •

Необходимо перенести сеттер и геттер поля university из классов Teacher и Student в класс UniversityPerson.

* •

Необходимо унаследовать классы Teacher и Student от класса UniversityPerson.

* •

Необходимо изменить тип поля university на University.

**5.**

5.1. Создание шаблонного метода.  
5.1.1. Добавь в класс **Human** метод String getPosition(), который должен возвращать строку "***Человек***".  
5.1.2. Переопредели этот метод в классе **Student** и **Teacher**. Метод должен возвращать "***Студент***" и "***Преподаватель***" соответственно.  
5.1.3. Замени метод printData в подклассах шаблонным методом в базовом классе, использующим getPosition().  
5.2. Замена делегирования наследованием. Класс **Worker** должен наследоваться от **Human**, а не содержать его.  
5.3. Переименование метода. Переименуй метод setSlr, чтобы было понятно сеттером чего является этот метод.

**Требования:**

* •

В классе Human должен существовать метод String getPosition(), который возвращает строку "Человек".

* •

В классе Student переопредели метод String getPosition(), чтобы он возвращал строку "Студент".

* •

В классе Teacher переопредели метод String getPosition(), чтобы он возвращал строку "Преподаватель".

* •

Необходимо заменить метод printData() в классе Student на метод printData() в классе Human. Используй getPosition().

* •

Необходимо заменить метод printData() в классе Teacher на метод printData() в классе Human. Используй getPosition().

* •

Класс Worker должен наследоваться от класса Human, а не содержать его.

* •

В классе Worker необходимо переименовать метод setSlr(double) на setSalary(double).

**6.**

6.1. Замена параметра набором специализированных методов. Замени метод setValue() класса **Student** специализированными методами setCourse и setAverageGrade.  
6.2. Добавление параметра. Добавить параметр с типом double в метод getStudentWithAverageGrade(), чтобы было понятно с каким средним балом нужен студент.  
Реализуй метод getStudentWithAverageGrade().  
6.3. Удаление параметра. Убери параметр из метода getStudentWithMaxAverageGrade().  
Реализуй этот метод, он должен возвращать студента с максимальным средним балом.  
6.4. Разделение запроса и модификатора. Раздели метод getStudentWithMinAverageGradeAndExpel на Student getStudentWithMinAverageGrade() и void expel(Student **student**). Первый метод должен возвратить студента с минимальным средним балом, а второй - отчислить переданного студента (*удалять из списка* **students**).

**Требования:**

* •

Необходимо заменить метод setValue класса Student специализированными методами setCourse и setAverageGrade.

* •

Необходимо добавить параметр с типом double в метод getStudentWithAverageGrade() класса University и реализовать метод.

* •

Необходимо удалить параметр из метода getStudentWithMaxAverageGrade(double) класса University и реализовать метод.

* •

Необходимо разделить метод getStudentWithMinAverageGradeAndExpel на Student getStudentWithMinAverageGrade() и void expel(Student student) и реализовать эти два метода.

**7.**

7.1. Параметризация метода. Замени методы incAverageGradeBy01() и incAverageGradeBy02() одним методом incAverageGrade(double **delta**).  
7.2. Передача всего объекта. Перепиши метод addInfoAboutStudent(), чтобы он в качестве параметра принимал объект типа **Student**.  
7.3. Замена параметра вызовом метода. Перепиши метод printInfoAboutStudent(), чтобы он не требовал в качестве параметра имя студента, а получал его, вызвав соответствующий метод у переданного объекта.  
7.4. Замена параметров объектом. Перепиши методы setBeginningOfSession и setEndOfSession,чтобы они вместо набора параметров принимали по одному объекту даты.

**Требования:**

* •

Необходимо заменить методы incAverageGradeBy01() и incAverageGradeBy02() класса Student одним методом incAverageGrade(double delta).

* •

Необходимо изменить метод addInfoAboutStudent(String, int, double) класса StudentsDataBase, чтобы он в качестве параметра принимал объект типа Student.

* •

Необходимо изменить метод printInfoAboutStudent(String, Student) класса StudentsDataBase, чтобы он не требовал в качестве параметра имя студента.

* •

Необходимо изменить метод setBeginningOfSession(int, int, int) класса Student, чтобы он принимал один параметр типа Date.

* •

Необходимо изменить метод setEndOfSession(int, int, int) класса Student, чтобы он принимал один параметр типа Date.

**8.**

8.1. Удаление сеттера. Удали метод setId(). Поле id должно устанавливаться только в момент создания объекта.  
8.2. Сокрытие метода (***поля***). Изменить область видимости поля nextId в соответствии с областью его использования.  
8.3. Замена исключения проверкой условия. Перепиши метод removeStudent(int **index**), чтобы он удалял студента из списка студентов только, если он там есть. Метод не должен кидать исключение.  
8.4. Удаление управляющего флага. Перепиши метод findDimaOrSasha(), сохранив логику его работы. В методе не должны использоваться флаги типа found, воспользуйся оператором break.

**Требования:**

* •

Необходимо удалить метод setId(int) класса Human.

* •

Необходимо изменить область видимости поля nextId класса Human с public на private.

* •

Необходимо изменить метод removeStudent(int index) класса StudentsDataBase, чтобы он не бросал исключение.

* •

Необходимо изменить метод findDimaOrSasha() класса StudentsDataBase, сохранив логику его работы. Из метода нужно удалить флаг boolean found и воспользоваться оператором break.

**9.**

9.1. Самоинкапсуляция поля. Перепиши метод incAverageGrade() используя **сеттер** и **геттер** для доступа к averageGrade.  
9.2. Замена поля-массива объектом. Замени массив int[] size объектом нового типа **Size**, содержащим публичные поля: рост int height и вес int weight. Публичный класс **Size** объяви внутри класса **Human**.  
9.3. Инкапсуляция поля. Сокрой поле company в классе **Worker**. Добавь **сеттер** и **геттер** для него.  
9.4. Замена кодирования типа классом.  
9.4.1. Объяви публичный класс группы крови **BloodGroup** внутри пакета **human**.  
9.4.2. Добавь в класс **BloodGroup** приватное константное поле int code, приватный конструктор, принимающий int и инициализирующий поле code, **геттер** для поля класса.  
9.4.3. Добавь в класс **BloodGroup** статические методы first(), second(), third() и fourth(), создающие и возвращающие объекты типа BloodGroup с правильным кодом внутри (**1**, **2**, **3** и **4** соответственно).  
9.4.4. Примени в классе **Human** новый тип **BloodGroup**.

**Требования:**

* •

Необходимо изменить метод incAverageGrade() класса Student с использованием сеттера и геттера для доступа к averageGrade.

* •

Необходимо заменить массив int[] size класса Human объектом нового типа Size.

* •

Необходимо изменить модификатор доступа поля company в классе Worker на приватный. Необходимо добавить сеттер и геттер для этого поля.

* •

Необходимо создать публичный класс BloodGroup внутри пакета human.

* •

Необходимо добавить в класс BloodGroup константное поле int code, приватный конструктор, принимающий int и инициализирующий поле code, геттер для поля класса.

* •

Необходимо добавить в класс BloodGroup статические методы first(), second(), third() и fourth(), создающие и возвращающие объекты типа BloodGroup с правильным кодом внутри (1, 2, 3 и 4 соответственно).

* •

Необходимо изменить тип поля bloodGroup класса Human на BloodGroup, обновить сеттер и геттер.

* •

Необходимо удалить из класса Human константы: FIRST, SECOND, THIRD, FOURTH.

**10.**

Разберись с кодом в пакете **car** (***машина***).  
10.1. Замена конструктора фабричным методом.  
10.1.1. Объяви классы **Truck** (**грузовик**), **Sedan** (**седан**) и **Cabriolet** (**кабриолет**), унаследованные от **Car**.  
10.1.2. Добавь в них конструкторы, принимающие int numberOfPassengers.  
10.1.3. Добавь фабричный статический метод Car create(int type, int **numberOfPassengers**) в класс **Car** и реализуй его.  
10.1.4. Измени область видимости конструктора класса **Car**.

**Требования:**

* •

Нужно создать классы Truck (грузовик), Sedan (седан) и Cabriolet (кабриолет), унаследованные от Car.

* •

Необходимо в созданные классы добавить конструкторы, принимающие int numberOfPassengers.

* •

Необходимо создать фабричный статический метод Car create(int type, int numberOfPassengers) в классе Car и реализовать его.

* •

Необходимо изменить область видимости конструктора класса Car с public на protected.

**11.**

11.1. Замена кода ошибки исключением. Перепиши метод заправиться fill(double **numberOfLiters**), чтобы он в случае ошибки кидал исключение Exception.  
11.2. Разбиение условного оператора.  
11.2.1. Добавь и реализуй метод в классе **Car**, определяющий относится ли переданная дата к лету: boolean isSummer(Date **date**, Date **summerStart**, Date **summerEnd**).  
11.2.2. Добавь и реализуй метод, рассчитывающий расход топлива зимой: double getWinterConsumption(int **length**).  
11.2.3. Добавь и реализуй метод, рассчитывающий расход топлива летом: double getSummerConsumption(int **length**).  
11.2.4. Перепиши метод getTripConsumption(), используя новые методы.

**Требования:**

* •

Необходимо изменить метод fill(double numberOfLiters) класса Car, чтобы он в случае ошибки бросал исключение Exception.

* •

Необходимо добавить в класс Car и реализовать публичный метод boolean isSummer(Date date , Date summerStart, Date summerEnd).

* •

Необходимо добавить в класс Car и реализовать публичный метод double getWinterConsumption(int length).

* •

Необходимо добавить в класс Car и реализовать публичный метод double getSummerConsumption(int length).

* •

Необходимо изменить метод getTripConsumption(), используя методы: isSummer, getWinterConsumption, getSummerConsumption.

**12.**

12.1. Объединение условных операторов.  
12.1.1. Добавь внутренний метод, сообщающий, могут ли быть перевезены пассажиры boolean canPassengersBeTransferred() в класс **Car**.  
Метод должен возвращать **true**, если водитель доступен isDriverAvailable и есть топливо fuel.  
12.1.2. Перепиши метод getNumberOfPassengersCanBeTransferred(), объединив условные операторы (*используй метод* canPassengersBeTransferred()).  
12.2. Объединение дублирующихся фрагментов в условных операторах. Перепиши метод startMoving(), чтобы в нем не было повторяющихся вызовов функций.  
12.3. Замена магического числа символьной константой. Замени магические числа в методе getMaxSpeed() на константные переменные метода:  
MAX\_TRUCK\_SPEED, MAX\_SEDAN\_SPEED и MAX\_CABRIOLET\_SPEED.  
12.4. Замена условного оператора полиморфизмом.  
12.4.1. Переопредели метод getMaxSpeed() в подклассах, избавившись от условного оператора.  
12.4.2. Метод getMaxSpeed() в классе **Car** сделай абстрактным.

**Требования:**

* •

Необходимо создать приватный метод boolean canPassengersBeTransferred() в классе Car и реализовать его.

* •

Необходимо изменить метод getNumberOfPassengersCanBeTransferred(), объединив условные операторы (используй метод canPassengersBeTransferred()).

* •

Необходимо изменить метод startMoving(), чтобы в нем не было повторяющихся вызовов метода fastenDriverBelt().

* •

Необходимо переопределить метод getMaxSpeed() в классе Truck. В методе нужно использовать константную переменную метода MAX\_TRUCK\_SPEED, значение которой равно 80.

* •

Необходимо переопределить метод getMaxSpeed() в классе Sedan. В методе нужно использовать константную переменную метода MAX\_SEDAN\_SPEED, значение которой равно 120.

* •

Необходимо переопределить метод getMaxSpeed() в классе Cabriolet. В методе нужно использовать константную переменную метода MAX\_CABRIOLET\_SPEED, значение которой равно 90.

* •

Метод getMaxSpeed() в классе Car необходимо сделать абстрактным.

**13.**

Разберись с кодом пакета **user** (*пользователь*).

13.1. Извлечение метода. Добавь метод printInfo(), который будет выводить имя и фамилию  
в консоль в формате

Имя: Вася  
Фамилия: Пупкин

Замени повторяющийся код метода printUsers() его вызовом.  
13.2. Встраивание метода. Избавься от метода ageLessThan16().  
13.3. Перемещение метода. Перемести методы printInfo() и printAdditionalInfo() в класс **User**.  
13.4. Расщепление переменной. Переменная age в методе calculateAverageAge() используется для разных промежуточных значений. Перепиши метод без использования этой переменной.  
13.5. Удаление присваиваний параметрам. Перепиши метод calculateRate(), чтобы он не пытался менять входные параметры, а просто возвращал рассчитанное значение.

**Требования:**

* •

Необходимо создать метод printInfo() в классе User, который будет выводить имя и фамилию в консоль так же, как это делается в методе printUsers() класса UserHelper.

* •

Необходимо заменить в методе printUsers() класса UserHelper повторяющийся код вызовами метода printInfo().

* •

Необходимо избавиться от метода ageLessThan16() класса UserHelper.

* •

Необходимо переместить метод printAdditionalInfo() в класс User. Обрати внимание, что метод printAdditionalInfo теперь не должен получать объект класса User в качестве параметра.

* •

Необходимо переписать метод calculateAverageAge() класса UserHelper без использования переменной age. Переменную age - удалить.

* •

Перепиши метод calculateRate() класса UserHelper, чтобы метод не изменял входные параметры, а просто возвращал рассчитанное значение (метод должен возвращать int).

**14.**

14.1. Перемещение поля. Замени поля isManAnya и isManRoma полем man в нужном классе.  
Добавь **сеттер** и **геттер** для нового поля (*при выборе имен методов учти тип поля*).  
14.2. Извлечение класса.  
14.2.1. Добавь класс **Address** в пакет **user**.  
14.2.2. Перенеси поля country, city и house в новый класс.  
14.2.3. Добавь **сеттеры** и **геттеры** для них.  
14.2.4. Перепиши класс **User**, используя поле класса **Address** address.  
14.3. Встраивание класса. Класс **House** почти ничего не делает, избавься от него.  
14.4. Сокрытие делегирования.  
14.4.1. Добавь в класс **User** метод getBoss().  
14.4.2. Перепиши реализацию метода getBossName(User **user**) класса **UserHelper**.

**Требования:**

* •

Необходимо заменить поля isManAnya и isManRoma класса UserHelper полем boolean man в классе User. Так же добавь сеттер и геттер для нового поля.

* •

Необходимо добавить класс Address в пакет user.

* •

Необходимо перенести поля country, city и house из класса User в класс Address.

* •

Необходимо добавить сеттеры и геттеры для полей country, city, house класса Address.

* •

В классе User необходимо переписать методы: getAddress, getCountry, setCountry, getCity, setCity. И нужно добавить в класс приватное поле Address address.

* •

Необходимо избавиться от класса House. Поле класса String house нужно перенести в класс Address. Необходимо обновить геттер и сеттер поля.

* •

Необходимо добавить в класс User метод getBoss() и реализовать этот метод.

* •

Необходимо изменить реализацию метода getBossName(User user) класса UserHelper (используй метод getBoss() класса User).

**15.**

***Отлично****. Ты освоил основные методы рефакторинга: Подъем поля, Подъем метода, Инкапсуляция коллекции, Извлечение подкласса, Подъем тела конструктора, Спуск метода, Спуск поля, Извлечение интерфейса, Свертывание иерархии, Замена наследования делегированием, Извлечение суперкласса, Замена простого поля объектом, Создание шаблонного метода, Замена делегирования наследованием, Переименование метода, Замена параметра набором специализированных методов, Добавление параметра, Удаление параметра, Разделение запроса и модификатора, Параметризация метода, Передача всего объекта, Замена параметра вызовом метода, Замена параметров объектом, Удаление сеттера, Сокрытие метода, Замена исключения проверкой условия, Удаление управляющего флага, Самоинкапсуляция поля, Замена поля-массива объектом, Инкапсуляция поля, Замена кодирования типа классом, Замена конструктора фабричным методом, Замена кода ошибки исключением, Разбиение условного оператора, Объединение условных операторов, Объединение дублирующихся фрагментов в условных операторах, Замена магического числа символьной константой, Замена условного оператора полиморфизмом, Извлечение метода, Встраивание метода, Перемещение метода, Расщепление переменной, Удаление присваиваний параметрам, Перемещение поля, Извлечение класса, Встраивание класса и Сокрытие делегирования.*

Конечно есть еще и другие, обязательно ознакомься и с ними. Очень важно не только уметь применять **методы рефакторинга**, но и **видеть места**, где их **нужно применить**.  
По мере возрастания опыта, такие места будет все проще и проще находить.  
Мартин Фаулер в своей книге "***Рефакторинг: Улучшение существующего кода***" называет такие места "***код с душком***". Обязательно прочитай про основные варианты "кода с душком". Желаю, чтобы твой код пах хорошо!

**Требования:**

* •

Тест всегда проходит.