**Рефакторинг (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.

### Рефакторинг (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.

**Рефакторинг (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:  
1. Необходимо спустить поле course в нужный класс и сделать его приватным.  
2. Необходимо спустить геттер для поля course в нужный класс.  
3. Интерфейс Alive должен существовать в пакете human.  
4. Интерфейс Alive должен содержать объявление метода жить live().  
5. Класс Human должен поддерживать интерфейс Alive.  
6. Класс Professor нужно удалить.

### Рефакторинг (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**.

### Рефакторинг (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:  
1. В классе Human должен существовать метод String getPosition(), который возвращает строку "Человек".  
2. В классе Student переопредели метод String getPosition(), чтобы он возвращал строку "Студент".  
3. В классе Teacher переопредели метод String getPosition(), чтобы он возвращал строку "Преподаватель".  
4. Необходимо заменить метод printData() в классе Student на метод printData() в классе Human. Используй getPosition().  
5. Необходимо заменить метод printData() в классе Teacher на метод printData() в классе Human. Используй getPosition().  
6. Класс Worker должен наследоваться от класса Human, а не содержать его.  
7. В классе Worker необходимо переименовать метод setSlr(double) на setSalary(double).

### Рефакторинг (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).

### Рефакторинг (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 setBeginningOfSessionи setEndOfSession,чтобы они вместо набора параметров принимали по одному объекту даты.

Требования:  
1. Необходимо заменить методы incAverageGradeBy01() и incAverageGradeBy02() класса Student одним методом incAverageGrade(double delta).  
2. Необходимо изменить метод addInfoAboutStudent(String, int, double) класса StudentsDataBase, чтобы он в качестве параметра принимал объект типа Student.  
3. Необходимо изменить метод printInfoAboutStudent(String, Student) класса StudentsDataBase, чтобы он не требовал в качестве параметра имя студента.  
4. Необходимо изменить метод setBeginningOfSession(int, int, int) класса Student, чтобы он принимал один параметр типа Date.  
5. Необходимо изменить метод setEndOfSession(int, int, int) класса Student, чтобы он принимал один параметр типа Date.

### Рефакторинг (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.

### Рефакторинг (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 Sizeобъяви  
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:  
1. Необходимо изменить метод incAverageGrade() класса Student с использованием сеттера и геттера для доступа к averageGrade.  
2. Необходимо заменить массив int[] size класса Human объектом нового типа Size.  
3. Необходимо изменить модификатор доступа поля company в классе Worker на приватный. Необходимо добавить сеттер и геттер для этого поля.  
4. Необходимо создать публичный класс BloodGroup внутри пакета human.  
5. Необходимо добавить в класс BloodGroup константное поле int code, приватный конструктор, принимающий int и инициализирующий поле code, геттер для поля класса.  
6. 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).  
7. Необходимо изменить тип поля bloodGroup класса Human на BloodGroup, обновить сеттер и геттер.  
8. Необходимо удалить из класса Human константы: FIRST, SECOND, THIRD, FOURTH.

### Рефакторинг (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:  
1. Нужно создать классы Truck (грузовик), Sedan (седан) и Cabriolet (кабриолет), унаследованные от Car.  
2. Необходимо в созданные классы добавить конструкторы, принимающие int numberOfPassengers.  
3. Необходимо создать фабричный статический метод Car create(int type, int numberOfPassengers) в классе Car и реализовать его.  
4. Необходимо изменить область видимости конструктора класса Car с public на protected.

### Рефакторинг (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.

### Рефакторинг (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:  
1. Необходимо создать приватный метод boolean canPassengersBeTransferred() в классе Car и реализовать его.  
2. Необходимо изменить метод getNumberOfPassengersCanBeTransferred(), объединив условные операторы (используй метод canPassengersBeTransferred()).  
3. Необходимо изменить метод startMoving(), чтобы в нем не было повторяющихся вызовов метода fastenDriverBelt().  
4. Необходимо переопределить метод getMaxSpeed() в классе Truck. В методе нужно использовать константную переменную метода MAX\_TRUCK\_SPEED, значение которой равно 80.  
5. Необходимо переопределить метод getMaxSpeed() в классе Sedan. В методе нужно использовать константную переменную метода MAX\_SEDAN\_SPEED, значение которой равно 120.  
6. Необходимо переопределить метод getMaxSpeed() в классе Cabriolet. В методе нужно использовать константную переменную метода MAX\_CABRIOLET\_SPEED, значение которой равно 90.  
7. Метод getMaxSpeed() в классе Car необходимо сделать абстрактным.

### Рефакторинг (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:  
1. 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.  
2. Необходимо заменить в методе printUsers() класса UserHelper повторяющийся код вызовами метода printInfo().  
3. Необходимо избавиться от метода ageLessThan16() класса UserHelper.  
4. Необходимо переместить метод printAdditionalInfo() в класс User. Обрати внимание, что метод printAdditionalInfo теперь не должен получать объект класса User в качестве параметра.  
5. Необходимо переписать метод calculateAverageAge() класса UserHelper без использования переменной age.  
6. Перепиши метод calculateRate() класса UserHelper, чтобы метод не изменял входные параметры, а просто возвращал рассчитанное значение (метод должен возвращать int).