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.

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).

Требования:  
1. Необходимо заменить метод setValue класса Student специализированными методами setCourse и setAverageGrade.  
2. Необходимо добавить параметр с типом double в метод getStudentWithAverageGrade() класса University и реализовать метод.  
3. Необходимо удалить параметр из метода getStudentWithMaxAverageGrade(double) класса University и реализовать метод.  
4. Необходимо разделить метод getStudentWithMinAverageGradeAndExpel на Student getStudentWithMinAverageGrade() и void expel(Student student) и реализовать эти два метода.