Laboratory Work #24

Java Object-Oriented Programming. Code Reuse. Inheritance

LEARN. GROW. SUCCEED.

Faculty of Information Technology and Robotics
Belarusian National Technical University
by Viktor Ivanchenko / ivanvikvik@bntu.by / Minsk

ЛАБОРАТОРНАЯ РАБОТА #24

ООП в Java. Повторное использование кода. Наследование

Цель работы 

Углубить свои фундаментальные знания в использовании методологии ООП, а также научиться практически применять инструменты языка Java для повторного использования кода в виде ассоциации, наследования, агрегации, композиции и делегирования.

Требования 

1) Необходимо скорректировать UML-диаграмму взаимодействия классов и объектов программной системы с учётом вносимых дополнений и изменений.
2) При корректировки программной системы необходимо полностью использовать своё ООП-воображение и по максимум использовать возможности, которые предоставляет язык Java для программирования с использованием методологии ООП.
3) Каждый пользовательский тип должен иметь адекватное осмысленное имя и информативный состав (соответствующие конструкторы: по умолчанию, с параметрами, конструктор-копирования; ***get***- и ***set***-методы для доступа к состоянию объекта; корректно переопределённые методы базового класса *Object*: ***toString()***, ***equals()***, ***hashCode()*** и др.).
4) Также рекомендуется придерживаться ***Single Responsibility Principle***, ***SRP*** (принципа единственной ответственности): у каждого пакета, класса или метода должна быть только одна ответственность (цель), т.е. должна быть только одна причина изменить в дальнейшем соответствующий блок кода.
5) Добавляемые классы необходимо грамотно разложить по соответствующим пакетам, которые должны иметь «адекватные» названия и быть вложены в указанные стартовые пакеты: ***by.bntu.fitr.poisit.nameofstudent.nameofproject***.

6) В соответствующих важных компонентах программной системы необходимо предусмотреть «защиту от дурака».
7) В качестве хранилище данных использовать Java-массивы!!! Все контейнерный класс должны поддерживать расширяемость, т.е. они не ограничены в объёме хранимых данных и динамически должны расширяться.
8) Весь код модели системы должен быть покрыт модульными тестами, подтверждающие качество и работоспособность ядра системы. Необходимо проверить все тестовые случаи. Для реализации модульных тестов рекомендуется использовать ***jUnit*** фреймворк.
9) При разработке программ придерживайтесь соглашений по написанию кода на *Java* (***Java Code-Convention***) !!!

Основное задание 

Необходимо произвести рефакторинг программной системы, созданной в предыдущей лабораторной работе, следующим образом:

- классы, описывающие объекты соответствующей предметной области (бизнес-объекты), должны быть сведены в иерархическую структуру (произвести, где это необходимо, классификацию типов); к примеру, в предыдущей лабораторной работе была только сущность автомобиль/автотранспорт, а теперь должна быть иерархия автотранспорта с соответствующими характеристиками: легковой автомобиль (седан, универсал, хэтчбэк, ...), грузовой автомобиль (фура, самосвал, бетономешалка ,...), пассажирский автомобиль (автобус, микроавтобус, минивэн, ...) и т.д., никто никого не ограничивает в фантазиях;
- логика системы должна быть реализована внутри соответствующих функциональных классов;
- логика системы и большинство других компонентов должны зависеть преимущественно только от абстракции, а не от реализаций;
- необходимо дополнительно для безопасности выполнения кода добавить по возможности в методы бизнес логики проверку входящих объектов на соответствие типа, с которым должна взаимодействовать данная логика.

Контрольные вопросы 

1. Что такое повторное использование кода?
2. Какие подходы и средства предлагает методология ООП для повторного использования кода?
3. На что указывает принцип ***DRY*** (***Don't Repeat Yourself***)?
4. Что такое зависимость? Как зависимость реализуется в языке Java?
5. Что такое ассоциация? Как реализована ассоциация в языке Java?
6. Что такое наследование? Как реализовано наследование в языке Java?
7. Какие преимущества и недостатки есть у множественного наследования классов?
8. Разрешено ли в языке Java множественное наследование? Если разрешено множественное наследование в языке Java, то как наследуются дочерним классом одинаковые характеристики из разных базовых классов?
9. Как при наследовании вызываются конструкторы?
10. Зачем нужно в языке Java ключевое слово ***super***?
11. Кто находится во главе всей иерархии в языке Java?
12. Какой именно минимальный функционал получают все экземпляры классов в языке Java?
13. Опишите функционал базового класса Object для всего ООП-мира в языке Java.
14. Что такое агрегация? Как реализована агрегация в языке Java?
15. Что такое композиция? Как реализована композиция в языке Java?
16. Что такое делегирование? Как реализовано делегирование в языке Java?
17. Основное отличие наследования от агрегации/композиции?
18. Преимущества и недостатки композиции и агрегации? Преимущества и недостатки наследования?
19. Как с помощью языка UML отобразить все зависимости между классами на UML-диаграмме классов?
20. Какой должен базовый класс у классов-исключений? Как создавать пользовательские исключения?