# Основы разработки приложений на Java

## Уральский производственный центр АйТи – УГАТУ

### Лабораторная работа №1

Цели: 1. Освоение синтаксиса и основ ООП в Java.

2. Изучение возможностей стандартных структур данных (коллекций)

3. Освоение работы с библиотекой тестирования JUnit

Задание.

Делать можно в любой среде NetBeans (рекоменд.), Eclipse, IDEA.

Обязательные требования:

* отсутствие дублирование кода >2-3 строк.
* осмысленные и соответствующие соглашению имена классов, методов и полей
* инкапсуляция: минимум public-методов, все поля private.

**`**

**1.** Разработать демонстрационную систему классов для моделирования систем хранения объектов.

* Основной класс – предмет (Item или аналогичное название)

имеет **название,** **вес и доп. свойства**

вес – не обязательно поле, у составного объекта он вычисляется

доп. свойства – это Set<String> (=new HashSet<String>()) - сюда можно помещать параметры типа цвета или «плоский», «тяжелый»)

(абстрактный класс? интерфейс?)

Пример - кирпич, 2 кг, {плоский=да}

Свойства предметов неизменяемы.

метод getInfo – подробная информация о предмете.

* Есть предметы-контейнеры. Они тоже являются предметами, но фактически содержат несколько предметов внутри. У всех их должен быть базовый класс (или интерфейс), позволяющий перечислить все предметы внутри (можно реализовать Iterable<Item>)
* Пример – **мешок**. методы – добавить предмет, вытащить предмет (достаётся случайный), найти предмет по имени. У мешка есть предельный вес *внутри*, больше которого добавлять нельзя. Собственный вес мешка задаётся в конструкторе.

Cам мешок тоже является предметом.

* Другой пример – **стопка.** Методы те же, но добавляется всегда наверх, забирается также сверху. Вместо макс. веса – максимальное кол-во предметов.( можно сделать функцию «можно ли положить ещё?»).

Вставить проверку, что в стопку можно сложить только “плоские” предметы (размерами пренебрагаем).

* **Коробка –**  как мешок, только плоская (можно складывать в стопки)

Набор методов зависит от вашего опыта и фантазии, минимум перечислен.

(можно ещё открыть/закрыть коробку, ….)

У всех классов должен быть переопределён toString(кратко – для отладки).

**2.** При превышении веса или высоты стопки выбрасывается **собственное исключение (ItemStoreException?)** (для обучения сделайте его проверяемым, т.е. классом-наследником Exception).

Организуйте защиту от помещения одного и того же предмета в разные коробки/стопки одновременно (флажок у предмета или ссылка на контейнер или доп. свойство «внутри чего-то» или что-то ещё). (**ItemAlreadyPlacedException?**)

**3.** Проведите тестирование полученной системы (создать наборы тестов JUnit)   
Обязательно протестировать: создание объектов, вычисление веса, упаковка/распаковка выбрасывание исключений [**@Test(expected=SomeException.class ],** лучше тестировать классы сначала по отдельности, а потом вместе.

Пример сложного теста на словах:

(сначала проверить операции по отдельности)

A = кирпич1, 2кг, плоский

B = кирпич2, 2кг, плоский

C = кирпич3, 2кг, плоский

D= новый мешок (мешок1, макс. 5 кг, сам весит 0.1кг)

E= новая коробка(коробка1, макс. 10 кг, сама весит 0.5кг)

D.добавить (A)

D.добавить (B)

**assertEquals** (D.вес() , 4.1 кг)

E->добавить(C)

**assertEquals** (E.вес() , 2.5 кг)

E->добавить(D) - должно быть исключение если коробка не вмещает 2+4.1 = 6.1 кг.

**assertEquals** (E.вес() , 6.5 кг)

F = новая стопка

F.добавить(E) // ок – коробка плоская

F.добавить(С) // ок

**assertTrue** (F.вытащить() == С )

//F.добавить(D) - должно быть исключение