

Практика №1. ООП

Данное задание является стартовым для курса и включает несколько целей:

1. Отработка навыка решения типовых объектно-ориентированных задач с использованием языка Java. В ходе выполнения работы необходимо разработать группу классов, которые реализуют прописанные в задании требования.
2. Отработать навык формирования исполняемого приложения на языке Java
3. Отработать навык упаковки Java приложений в исполняемый архив.

Работа оценивается следующим образом:

- Задания 1-4 являются основными и приносят 6 баллов в сумме
- Задание 5 является необязательным т.к. требует разобраться в еще не разобранном материале. Задание приносит 1 балл, который начисляется сверх основных
- После полной сдачи и проверки домашних заданий назначается собеседование по теме. Собеседование приносит вплоть до 6 основных баллов.
 - Собеседование согласуется через MR в репозитории (или другими удобными способами, если оно доступно)
 - Содержание собеседования концентрируется вокруг тем, которые необходимо знать для выполнения данного практического задания
 - Оценка производится преподавателем таким образом, что 0 – у проверяемого нет знаний по теме, 6- уверенное знание и понимание материала, любое число в промежутке от 0 до 6 соответствует примерному проценту знаний.
 - В случае несогласия с оценкой - допустимо подать однократную апелляцию для прохождения повторного собеседования, которое может как увеличить, так и уменьшить оценку за результат выполнения работы.

Итого: 12 (+1) баллов

Базовые требования:

1. Код должен содержать JavaDoc описания класса и методов. Необходимо приложить сгенерированный html файл с JavaDoc. JavaDoc должен описывать назначение класса, параметры методов, возвращаемые значения и возможные исключения являющиеся контрактом метода.
2. Должны быть реализованы в main методе типовые сценарии применения разработанных классов
3. Должен быть приложен скрипт, включающий компиляцию, формирование jar архива, и его запуск на исполнение

Задание 1. Разработка базового класса (1 балл)

Необходимо разработать класс **Студент**. Данный класс должен отвечать следующим требованиям:

1. Имеет имя (строка)
2. Имеет список оценок (числовой)

3. Имя можно изменить или узнать в любой момент времени
4. Оценки можно добавлять, удалять и просматривать. Конкретный алгоритм добавления\удаления остается на усмотрение автора.
5. Два **Студента** равны если у них одинаковые имена и список оценок. Два списка оценок являются одинаковыми, если Списки равны, если содержат одинаковые значения с одинаковой кратностью, порядок не учитывается. Например [1,2,3] и [3,2,1] равны, а [1,2,3] и [3,3,3] не равны
6. Текстовое представление студента имеет вид: “Имя: [оценка1, оценка2,...,оценкаN]”
7. Студент может быть создан двумя способами:
 - a. с указанием имени
 - b. с указанием имени и списка оценок.

Критерии оценивания:

- Выполнены требования по наличию полей\методов\конструкторов

Задание 2. Добавление условий. (1 балл)

Необходимо внести следующие изменения в базовый код **Студента**:

1. Имя не может быть пустой строкой или **null**
2. Тип оценки может быть не только числом, но вообще любым типом данных, например строкой или датой. Тип оценки указывается при создании объекта.
3. При создании объекта **Студента** ему может быть задано указание на то, что является корректной оценкой, а что нет (например четные числа в диапазоне от 1 до миллиона, или слова “зачет” и “незачет”). Если признак корректности не задан, значит все значения данного типа данных являются корректными оценками. Условие корректности не может быть изменено в течении жизни объекта.
4. При попытке добавить **Студенту** некорректную оценку должно выбрасываться исключение.
5. Не должно быть способа добавить некорректную оценку.
6. При добавлении некорректной оценки должно выбрасываться специализированное unchecked-исключение.

Критерии оценивания:

- Изменение типа данных оценки не требует изменения класса **Студента**
- Признак корректности может быть любым мыслимым условием, и его изменение не требует изменения класса **Студента**

Задание 3. Отмена. (2 балла)

Необходимо внести следующие изменения в код класса **Студент** полученный в части 2:

Добавьте Студенту метод отмены последнего успешно выполненного действия. Действиям считаются изменение списка оценок или изменения имени.

Метод отмены можно вызывать до тех пор, пока по очереди не отменяются все действия и Студент не вернется к начальному состоянию. Попытка вызвать откат при отсутствии выполненных действий должна генерировать ошибку. Если какой-либо из методов изменения состояния студента упал с исключением – такое действие не учитывается при откате.

Критерии оценивания:

- Метод отмены не требует параметров
- Появление у класса дополнительных полей не потребует внесения изменений никуда, кроме как в методы получения и изменения значений добавленных полей. При этом у них также должна быть возможность участвовать в “отмене действия”.

Задание 4. Поток данных (2 балла)

Разработайте класс Flow, предназначенный для выполнения операций над коллекцией значений. Объект класса позволяет задавать список преобразований для каждого элемента, результируя либо одним финальным значением, либо их множеством.

Для класса необходимо выполнить следующие требования:

Создается через статические фабричные методы, позволяющие принимать:

- Список значений любого типа
- Варарг значений любого типа
- Начальное значение любого типа, способ получения нового значения на основе предыдущего, условия окончания.

Имеет четыре метода:

- **Функция.** Метод принимает функцию изменения одного значения и применяет её к каждому элементу исходного потока данных. Метод возвращает сам объект Flow.
- **Фильтр.** Метод принимает функцию проверки одного значения и оставляет только такие элементы оригинального потока данных, которые удовлетворяют этому значению. Метод возвращает сам объект Flow.
- **Сокращение.** Метод принимает бинарную операцию и применяет её последовательно ко всем элементам так, чтобы результатом стало одно итоговое значение. Полученное значение возвращается из метода.
- **Коллекционирование.** Метод принимает способ генерации начального состояния множества и способ добавления очередного значения в множество. Полученное множество возвращается из метода.

Гарантируйте, что методы **Функция** и **Фильтр** выполняются за $O(1)$ и не используют промежуточные коллекции. Все операции выполняются только в рамках методов **Сокращение** и **Коллекционирование**

Возможный пример:

```
Flow.of(1,-2,3).filter(x->x>0).reduce((x,y)->x+y)
```

Результатом будет 4, т.к. -2 не прошло условие, а остальные значения были сложены.

Задание 5. Тесты (1 бонусный балл)

Предоставьте в репозитории набор Junit тестов, которые тестируют корректность реализации кода для частей 1-4 задания.

Критерии оценивания:

1. Тесты запускаются Maven или Gradle скриптом
2. Code Coverage 80%
3. К работе необходимо приложить сгенерированный отчет по тестовому покрытию

Примеры вопросов для собеседования

1. Какие механизмы обеспечивают вариативность типа оценки Студента и чем они отличаются от Object
2. Покадите точки обеспечения инкапсуляции класса Студент.
3. Какие требования переопределения equals должны быть выполнены помимо явно прописанного в задании.
4. Какие механизмы обеспечивают возможность менять правило корректности оценки.
5. Почему реализация отмены изменения оценок через копии – плохая идея.
6. Что означают ленивые вычисления в контексте класса Flow