Изображение выглядит как текст

Автоматически созданное описание

Нарушен 4 принцип SOLID – принцип **Принцип разделения интерфейса** (interface segregation principle / ISP) в формулировке Роберта Мартина: «клиенты не должны зависеть от методов, которые они не используют». Принцип разделения интерфейсов говорит о том, что слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы клиенты маленьких интерфейсов знали только о методах, которые необходимы им в работе. В итоге, при изменении метода интерфейса не должны меняться клиенты, которые этот метод не используют.

Изображение выглядит как текст

Автоматически созданное описание

Здесь нарушено 2 принципа SOLID:

1. Принцип инверсии зависимостей (dependency inversion principle / DIP) — модули верхних уровней не должны зависеть от модулей нижних уровней, а оба типа модулей должны зависеть от абстракций; сами абстракции не должны зависеть от деталей, а вот детали должны зависеть от абстракций.
2. Принцип открытости / закрытости (open-closed principle / OCP) декларирует, что программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения. Это означает, что эти сущности могут менять свое поведение без изменения их исходного кода.

Изображение выглядит как текст

Автоматически созданное описание

Здесь нарушен 3 принцип SOLID - Принцип подстановки Барбары Лисков (Liskov substitution principle / LSP) в формулировке Роберта Мартина: «функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа не зная об этом».

Изображение выглядит как текст

Автоматически созданное описание

Переопределены hashCode и toString. Потому что MyClass наследуется от Object, а в нем нет метода save. Аннотации в данном случае для программиста.

Изображение выглядит как текст

Автоматически созданное описание

Просто пишем:

waiting();

Потому что класс Head имеет доступ ко всем полям класса Animal

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

2 ошибки:

1. Исключение и обрабатывается, и пробрасывается
2. В блоке catch Throwable e что неправильно. Правильно было бы сузить ошибку (например написать FailThisTest e)

Изображение выглядит как текст

Автоматически созданное описание

Это тип Throwable

Изображение выглядит как текст

Автоматически созданное описание

Вроде давали варианты ответов.

1. И можно кидать
2. Они наследуются от RuntimeException
3. Его нужно либо прокинуть либо обернуть в try-catch-finally

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, монитор, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

В Main написан ответ. Остальное – для понимания как вообще рефлексия работает.

Из полезного:

**Принципы SOLID:**

Принцип единственной обязанности / ответственности (single responsibility principle / SRP) обозначает, что каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.

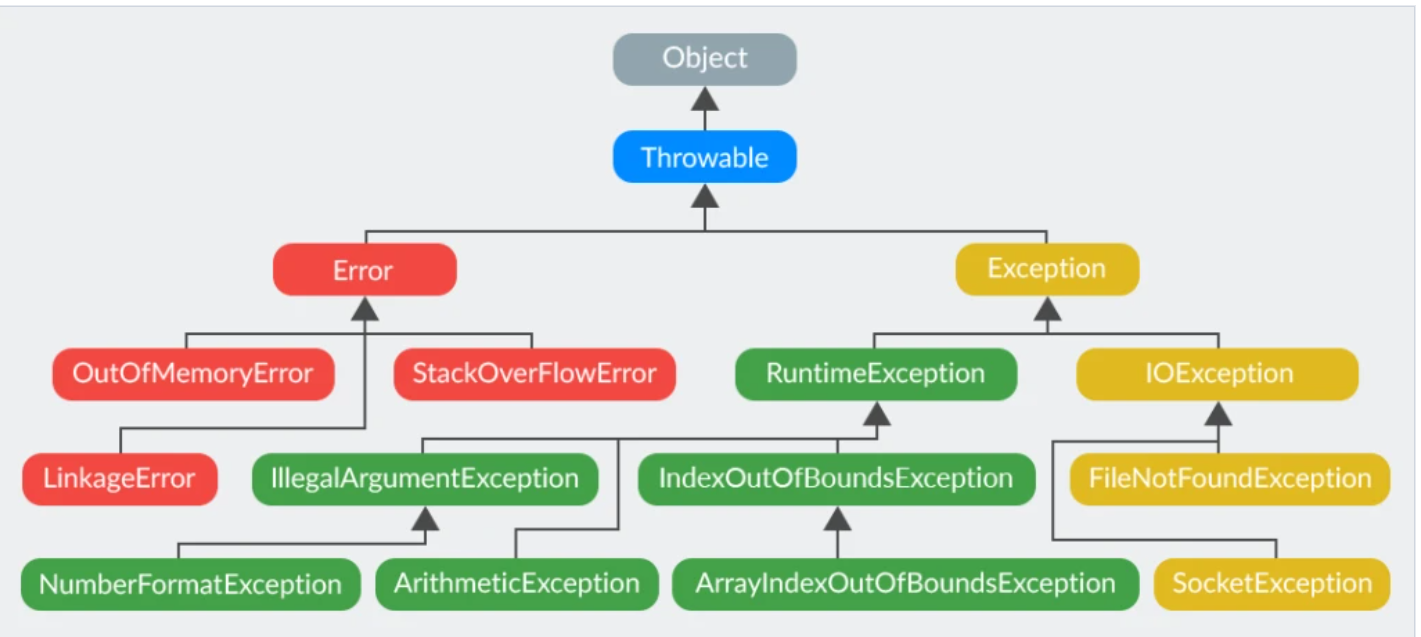
Принцип открытости / закрытости (open-closed principle / OCP) декларирует, что программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения. Это означает, что эти сущности могут менять свое поведение без изменения их исходного кода.

Принцип подстановки Барбары Лисков (Liskov substitution principle / LSP) в формулировке Роберта Мартина: «функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа не зная об этом».

Принцип разделения интерфейса (interface segregation principle / ISP) в формулировке Роберта Мартина: «клиенты не должны зависеть от методов, которые они не используют». Принцип разделения интерфейсов говорит о том, что слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы клиенты маленьких интерфейсов знали только о методах, которые необходимы им в работе. В итоге, при изменении метода интерфейса не должны меняться клиенты, которые этот метод не используют.

Принцип инверсии зависимостей (dependency inversion principle / DIP) — модули верхних уровней не должны зависеть от модулей нижних уровней, а оба типа модулей должны зависеть от абстракций; сами абстракции не должны зависеть от деталей, а вот детали должны зависеть от абстракций.

**Структура исключений в Java:**

****