Вопросы для подготовки:

- Что такое Рефлексия?

Рефлексия - это механизм исследования данных о программе во время её выполнения. Рефлексия позволяет исследовать информацию о полях, методах и конструкторах классов.

- Минусы Рефлексии

Худшая производительность в сравнении с классической работой с классами, методами и переменными;

Ограничения безопасности. Если мы захотим использовать рефлексию на классе, который защищен с помощью специального класса SecurityManager, то ничего у не выйдет т.к. этот класс будет выбрасывать исключения каждый раз, как мы попытаемся получить доступ к закрытым членам класса. Такая защита может применяться, например, в Апплетах (Applets);

Получение доступа к внутренностям класса, что нарушает принцип инкапсуляции.

- Что такое Класс класса, у кого он есть?

В Java есть специальный класс по имени Class, да-да, именно Class. Поэтому его и называют классом класса. С помощью него осуществляется работа с рефлексией, он и является входной точкой в мир рефлексии.

Class есть у:

классов, интерфейсов, перечислений;

примитивов и обёрток над ними;

массивов;

void. Да, ключевое слово void также имеет Class.

В общем, Class есть у всех объектов в Java.

- Как получить Класс класса?

1. Class.forName(“имя.пакета.класса”)

2. getClass()

3. имякласса.class

- Как получить информацию о переменных класса с помощью Рефлексии?

1) getDeclaredFields() - Метод возвращает все объявленные переменные в классе

2) getDeclaredField() - Метод возвращает переменную по её имени. Если переменной с таким именем нет, то метод выбросит checked NoSuchFieldException.

3) getFields() - В отличии от метода getDeclaredFields(), метод getFields() возвращает только public переменные

4) getField() - По аналогии с методом getFields(), метод getField() возвращает только public переменные. Даже если поле с таким именем есть, но оно не публичное, метод getField() бросит NoSuchFieldException

- Как получить информацию о методах в классе с помощью Рефлексии?

1) getDeclaredMethods()

Метод возвращает все объявленнные методы в классе

2) getDeclaredMethods()

Метод возвращает все объявленнные методы в классе

3) getMethods()

Метод возвращает все public методы класса и public методы его родительского класса/интерфейсов

4) getMethod()

Как и getMethods(), метод getMethod() возвращает только публичные методы. Если такого метода нет или он не публичный, мы получим NoSuchMethodException.

5) getEnclosingMethod()

Если класс является локальным или анонимным, метод getEnclosingMethod() возвращает тот метод в котором этот класс был создан, иначе метод возвращает null.

- Что вы знаете о классе Класс Field?

Класс Field предоставляет возможность:

получить значение поля, его тип, имя а так же модификаторы поля

получить список аннотаций, класс, в котором объявлено поле и другую информацию

установить новое значение в поле, даже если оно объявлено как private

- Что такое Аннотации в Java?

Аннотации в Java являются своего рода метками в коде, описывающими метаданные для функции/класса/пакета

- Какие функции выполняет Аннотации?

Аннотации используются для анализа кода, компиляции или выполнения. Аннотируемы пакеты, классы, методы, переменные и параметры.

- Какие встроенные аннотации в Java вы знаете?

@Override — аннотация-маркер, которая может применяться только к методам. Метод, аннотированный как @Override, должен переопределять метод супер класса. @Override — проверяет, переопределён ли метод. Вызывает ошибку компиляции / интерпретации, если метод не найден в родительском классе или интерфейсе;

@Deprecated — отмечает, что метод устарел и не рекомендуется к использованию. Предполагается, что по каким-то причинам этот метод пока оставлен, но будет удалён в будущих версиях. Вызывает предупреждение компиляции, если метод используется;

@SuppressWarnings — указывает компилятору подавить предупреждения компиляции, определённые в параметрах аннотации;

@SafeVarargs — указывает, что никакие небезопасные действия, связанные с параметром переменного количества аргументов, недопустимы. Применяется только к методам и конструкторам с переменным количеством аргументов, которые объявлены как static или final.

Аннотации, применяемые к другим аннотациям:

@Retention — определяет, как отмеченная аннотация может храниться — в коде, в скомпилированном классе или во время работы кода. Аннотация @Retention позволяет указать жизненный цикл аннотации: будет она присутствовать только в исходном коде, в скомпилированном файле, или она будет также видна и в процессе выполнения. Выбор нужного типа зависит от того, как вы хотите использовать аннотацию, например, генерировать что-то побочное из исходных кодов, или в процессе выполнения стучаться к классу через reflection.

@Documented — отмечает аннотацию для включения в документацию. Аннотация @Documented указывает, что помеченная таким образом аннотация должна быть добавлена в javadoc поля/метода и так далее.

@Target — отмечает аннотацию как ограничивающую, какие элементы аннотации могут быть к ней применены. Аннотация @Target указывает, что именно мы можем пометить этой аннотацией, это может быть поле, метод, тип и т. д.

@Inherited — отмечает, что аннотация может быть расширена подклассами аннотируемого класса. Аннотация @Inherited помечает аннотацию, которая будет унаследована потомком класса, отмеченного такой аннотацией.

- Что делают аннотации @Retention, @Documented, @Target и @Inherited?

- Что делают аннотации @Override, @Deprecated, @SafeVarargs и @SuppressWarnings?

- Какой жизненный цикл аннотации можно указать с помощью @Retention?

Существует 3 возможные варианты чтобы указать где аннотация будет жить. Они инкапсулированы в перечисление java.lang.annotation.RetentionPolicy. Это SOURCE, CLASS, RUNTIME.

SOURCE - содержаться только в исходном файле и отбрасываются при компиляции.

CLASS - сохраняются в файле, однако они недоступны JVM во время выполнения.

RUNTIME - сохраняются в файле во время компиляции и остаются доступными JVM во время выполнения.

- К каким элементам можно применять аннотацию, как это указать?

@Target

ElementType.PACKAGE – только для пакетов

TYPE – для классов

CONSTRUCTOR

METHOD

FIELD – для атрибутов (переменных) класса

PARAMETER

LOCAL\_VARIABLE

- Как создать свою Аннотацию?

public @interface About{

String info() default "";

}

- Атрибуты каких типов допустимы в аннотациях?

Примитивы, String, Class, enum, annotation, массивы любого из вышеперечисленных