**Вопросы по теме Инициализация данных:**

1. Разница между ранним (статическим) и поздним (динамическим) связыванием?
2. Для чего используются статические блоки инициализации?
3. Порядок вызова конструктора и блоков инициализации?
4. Если в блоке инициализации ошибка, что будет?
5. Где разрешена инициализация статических / нестатических полей?

**Ранее (статическое) и позднее (динамическое) связывание.**

**Связывание** – установка связи между переменной / методом и кодом, относящимся к этой переменной / методу. Например, у нас есть метод toString() класса Object. Мы вызываем этот метод в нашем коде myObject.toString(). Тут идет связывание между вызванным нами методом и кодом, который определяет этот метод в классе Object. Если переопределить метод toString(), то связь уже будет между вызванным методом и кодом переопределенного метода.

С помощью связывания определяется какой код будет запускаться при вызове переменной или метода.

**Ранее (статическое) связывание** – происходит во время компиляции программы. Применяется для связывания перегруженных методов, а также private, static, final методов.

**Позднее (динамическое) связываение** – происходит во время выполнения программы. Применяется для связывания переопределенных методов и виртуальных методов (Виртуальный метод - метод, тело которого может быть переопределено в момент выполнения программы. По умолчанию в Java методы, которые не static, не final и не private, являются виртуальными.)

*List<?> myList = new ArrayList<>();*

При раннем связывании для того, чтобы понять какой метод / переменную вызвать, используется тип ссылочной переменной (у переменной myList будет тип List).

При позднем связывании для того, чтобы понять какой метод / переменную вызвать, используется объект (у переменной myList будет тип ArrayList ).

**Инициализация данных.**

**Инициализация** – это когда впервые присваивается значение переменной.

**Блок инициализации** – блок кода, выполняемого во время загрузки классов и создании объектов. Существует два вида блоков инициализации: статический и динамический.

**Статический блок инициализации** – выполняется при первой загрузке класса, один раз. Используется для инициализации статических переменных или / и для выполнения необходимой нам логики (например, добавить в список некоторые значения). Если в статическом блоке произойдет ошибка, то будет вызвана ошибка ExceptionInInitializerError.

**Динамический блок инициализации** – выполняется перед конструктором при создании объекта. Используется для инициализации переменных (статич. и нестатич.) или / и для выполнения необходимой нам логики (например, увеличить счетчик, который подсчитывает количество созданных объектов). Если в динамическом блоке произойдет ошибка, то будет проброшено исключение-источник.

**Порядок вызова блоков инициализации и конструктора:**

1. Статический блоки класса-родителя
2. Статический блок класса-каследника
3. Динамический блок класса-родителя
4. Конструктор класса-родителя
5. Динамический блок класса-наследника
6. Конструктор класса-наследника