**Ответы на вопросы**

1. ООП - принцип взаимодействия с программой, где программа рассматривается как набор взаимодействующих между собой объектов. ООП позволяет упрощать код путем разбиения больших задач на несколько маленьких что делает возможным работу над кодом доступной сразу для нескольких программистов.

2. ООП был придуман как способ правильного написания кода и упрощения работы с ним. Создание классов хорошо помогает при выявлении ошибки в коде и и ориентации в нем, т.к. в противном случае код будет состоять из множества последовательно идущих операций не разбитых на части. ООП также хорош для работы в команде, т.к. разбиение больших задач на несколько маленьких делает возможным работу над кодом сразу для нескольких программистов.

3. Класс - тип данных, который можно рассматривать как набор данных и функций для работы с ними.

4. Объект - представитель класса имеющий конкретное состояние и поведение, которое определяется классом.

5. Свойства - любые данные, которыми можно охарактеризовать объект класса.

6. Метод класса - блок кода имеющий свое имя, который может быть вызван из разных частей программы. При вызове метода он выполняет свой код, а затем возвращает управление тому коду, который его вызвал. Методы также могут возвращать некоторое значение.

7. Модификаторы доступа – ключевые слова, которые позволяют задать область доступности элементов класса в коде, модификаторы определяют контекст в котором можно употреблять данную переменную или метод.

8. Интерфейс - совокупность методов и правил взаимодействия элементов системы, он определяет как элементы будут взаимодействовать между собой.

9. **Абстрактные классы в объектно-ориентированном программировании** — это базовые классы, которые можно наследовать, но нельзя реализовывать. То есть на их основе нельзя создать объект. В программировании абстрактные классы могут понадобиться, чтобы объединить реализацию нескольких схожих классов.

10 С**татический метод** класса – метод, не имеющий доступа к состоянию объекта. Слово статический используется в том смысле, что статические методы не относятся к динамике объекта, не используют и не меняют его состояния. Также из статического метода нельзя вызвать нестатический метод того же класса для вызванного объекта.

11. Инкапсуляция - свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали  
реализации от пользователя.Инкапсуляция неразрывно связана с понятием интерфейса класса.

Полиморфизм - свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Наследование - свойство системы, позволяющее описать новый класс на основе существующего с частично или полностью заимствованным функционалом. Класс, от которого производится наследование, называется базовым или родительским, новый класс – потомком.

**Сравнение ООП**

1. Имя исходного Java-файла должно соответствовать имени хранящегося в нем класса, поэтому мы обязаны назвать файл именем класса.

В Python вы можете объявить класс где угодно и когда угодно.

1. Во всех объектно-ориентированных языках данные об объекте где-то хранятся. И в Python, и в Java эти данные хранятся в атрибутах, которые являются переменными, связанными с конкретными объектами.

Одним из наиболее значительных отличий между Python и Java является то, как они определяют атрибуты класса и объекта и как эти языки управляют ими. Некоторые из этих различий вызваны ограничениями, налагаемыми языками, в то время как другие связаны с более эффективной практикой.

1. В Java мы объявляем атрибуты (с указанием их типа) внутри класса, но за пределами всех методов. Перед тем, как использовать атрибуты класса, мы должны их определить.

В Python же мы объявляем и определяем атрибуты внутри метода класса **init**(), который является аналогом конструктора в Java.

1. Вместо приватных переменных в Python имеется понятие непубличных (**non-public**) переменных экземпляра класса. Все переменные, названия которых начинаются с одинарного подчеркивания, считаются непубличными. Это соглашение об именах затрудняет доступ к переменной, но это всего лишь соглашение об именах, и мы все равно можем обратиться к переменной напрямую.

Java управляет доступом к методам и атрибутам, различая **публичные** и **приватные** данные. В Java ожидается, что атрибуты будут объявлены как приватные (или защищенные — protected, если нужно обеспечить к ним доступ потомкам класса). Таким образом мы ограничиваем доступ к ним извне. в Java не приветствуется объявление атрибутов публичными. Рекомендуется объявлять их приватными, а затем использовать публичные методы.

1. В Java мы получаем доступ к приватным атрибутам при помощи **сеттеров** (setters) и **геттеров** (getters).

В Python **свойства** обеспечивают управляемый доступ к атрибутам класса при помощи декораторов (decorators). Используя свойства, мы объявляем функции в питоновских классах подобно геттерам и сеттерам в Java (бонусом идет удаление атрибутов).

1. В Java класс ссылается сам на себя, используя ключевое слово this.

В Python ключевое слово self служит аналогичной цели: обращение к членам-атрибутам, но в отличие от Java, оно **обязательно.**

1. Python есть функции, а в Java их нет.
2. Python поддерживает множественное наследование, то есть создание класса более чем от одного родителя.

Язык Java же, в свою очередь, поддерживает только одиночное наследование, что означает, что классы в Java могут наследовать данные и поведение только от одного родительского класса. Зато в Java возможно наследование от множества интерфейсов.

1. Все классы в Java имеют своим предком класс Object, который содержит определенный набор методов и передает их своим потомкам. Потомки могут эти методы либо переопределять, либо использовать по умолчанию. Класс Object включает в себя следующие методы.

В Python для строкового представления объекта имеется два метода: **repr**() и **str**(). Однозначное представление объекта возвращается методом **repr**(), в то время как **str**() возвращает его в удобочитаемом виде.

1. Перегрузка операторов в Python означает возможность в классах переопределять различные операторы языка. Магические методы Python позволяют реализовать перегрузку операторов, чего Java не предлагает вообще.
2. И Java, и Python предоставляют способы исследования атрибутов и методов в классе.
3. В обоих рассматриваемых языках имеются способы проверить тип объекта. В Python мы используем type()для отображения типа переменной и isinstance () для определения, является ли данная переменная экземпляром или потомком определенного класса.

В Java мы вызываем метод getClass() для определения типа и используем instanceof для проверки на принадлежность классу.

1. В Python при помощи dir() мы видим все атрибуты и функции, содержащиеся в объекте (включая магические методы). Чтобы получить конкретные сведения о данном атрибуте или функции, используем getattr().

Java рассматривает атрибуты и методы как отдельные сущности, поэтому публичные методы извлекаются при помощи getDeclaredMethods().

1. И в Java, и в Python имеются механизмы для вызова методов через рефлексию, однако методы Python проще в управлении и вызове, чем в Java.

Основными отличиями ООП языка Python от стандартных реализаций является удобство в его использовании: множественное наследование, перегрузка, упрощенный вызов методов через рефлексию. В остальном же данные языки достаточно схожи в своей структуре, наибольшее отличие заключается лишь в краткости и простоте понимания кода языка Python.