Ответы на контрольные вопросы:

1 вопрос: На чём основано объектно-ориентированное программирование?

Принципы ООП:

0. Абстракция - выделение значимой информации и исключение из рассмотрения незначимой.
1. Инкапсуляция - свойство системы, позволяющее объединить данные и методы, работающие с ними, в одном классе.
2. Наследование - свойство системы, позволяющее описать один класс на основе уже существующего класса с частично или полностью заимствующейся функциональностью.
3. Полиморфизм - свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

2 вопрос: Какие аспекты стоит учитывать при проектировании классов?
→ Структура класса должна быть простой для понимания и модификации.
→ Классы должны проектироваться на основе принципов ООП (вопрос 1), например, хорошо бы использовать инкапсуляцию.
→ Классы не должны быть public в большинстве случаев (просто чтобы мы не дотянулись до переменных класса в других местах и случайно ничего не сломали), это тоже в какой-то степени инкапсуляция.

3 вопрос: Почему удобно разделять классы на интерфейс и реализацию?
(В языке C++ для разделения объявлений классов используются заголовочные файлы. В них определяется полная структура класса, включая приватные поля).

→ Это красиво и удобно: в одном классе у нас не будет понапихано всего сразу: если интерфейс класса, его переменные и функции, очень обширны, то проще подключить заголовочный файл для их объявления, а не тащить всё в одну кучу.
→ Мы можем наследовать интерфейс, но не обязаны это делать. В главном классе (классе реализации) у нас будет выбор использования того или иного интерфейса. Если их не разделять, то пришлось бы писать больше громоздких классов, каждый со своим интерфейсом.
→ Если попытаться развить вторую мысль, то можно прийти к выводу, что один интерфейс может применяться к нескольким реализациям. Например, елси это стандартные для нашей программы, но не стандартные в общем (нет в стандартных библиотеках) методы, то почему бы не описать их в интерфейсе, чтобы в нужный момент всегда можно было легко ими воспользоваться.

4 вопрос: Чем внутреннее связывание отличается от внешнего связывания?
Внешнее связывание означает, что переменные и функции после компоновки текущего файла будут доступны во всех файлах программы. Внутренне связывание говорит о том, что объекты будут видны только внутри текущего файла.
Можно провести аналогию с private и public классами (если позабыть про друзей), где только внутренние члены класса могут обращаться к private членам, а члены класса public доступны всем.

5 вопрос: Какими особенностями обладают именованные пространства имён?
Конфликт имён возникает, когда два одинаковых идентификатора находятся в одной области видимости, и компилятор не может понять, какой из этих двух следует использовать в конкретной ситуации. Компилятор или линкер выдаст ошибку, так как у них недостаточно информации, чтобы решить эту неоднозначность. Для решения таких проблем и придумали концепцию пространства имён.
Пространство имён определяет область кода, в которой гарантируется уникальность всех идентификаторов. Конфликт имён всё еще не решится, если оба идентификатора состоят в одном глобальном пространстве имён. Поэтому необходимо объявлять собственные пространства имён через ключевое

слово *namespace*. Всё, что объявлено внутри пользовательского пространства имен, - принадлежит только этому пространству имён.
→ У пространства имён должно быть своё имя (иначе это глобальное пространство).
→ Имя пространства имён обычно совпадает с названием библиотеки, которую мы пишем.
→ Пространство имён - дополнительное имя к тому, что уже есть.