**Билет 9**

**Размер файлов**: типичная длина файлов должна составлять 200 строк, с верхним пределом в 500 строк . Хотя это не должно считаться раз и навсегда установленным правилом, такие показатели весьма желательны . Маленькие файлы обычно более понятны, чем большие .

**Вертикальное разделение концепций:** Объявление импорта и все функции разделяются пустыми строками . Это чрезвычайно простое правило оказывает глубокое воздействие на визуальную структуру кода . Каждая пустая строка становится зрительной подсказкой, указывающей на начало новой самостоятельной концепции .

**Вертикальное сжатие**: если вертикальные пропуски разделяют концепции, то вертикальное сжатие подчеркивает тесные связи . Таким образом, строки кода, между которыми существует тесная связь, должны быть «сжаты» по вертикали .

**Вертикальные расстояния:** Концепции, тесно связанные друг с другом, должны находиться поблизости друг от друга по вертикали . Разумеется, это правило не работает для концепций, находящихся в разных файлах . Но тесно связанные концепции и не должны находиться в разных файлах, если только это не объясняется очень вескими доводами . Кстати, это одна из причин, по которой следует избегать защищенных переменных .*Объявления переменных*. Переменные следует объявлять как можно ближе к месту использования .*Переменные экземпляров*, напротив, должны объявляться в начале класса . Это не увеличивает вертикальное расстояние между переменными, потому что в хорошо спроектированном классе они используются многими, если не всеми, методами класса. *Зависимые функции*. Если одна функция вызывает другую, то эти функции должны располагаться вблизи друг от друга по вертикали, а вызывающая функция должна находиться над вызываемой (если это возможно). *Концептуальное родство*. Некоторые фрагменты кода требуют, чтобы их разместили вблизи от других фрагментов . Такие фрагменты обладают определенным концептуальным родством . Чем сильнее родство, тем меньше должно быть вертикальное расстояние между ними .Как мы уже видели, родство может быть основано на прямой зависимости (когда одна функция вызывает другую) или на использовании переменных в функциях . Однако существуют и другие разновидности родства . Например, родство возникает в том случае, если группа функций выполняет аналогичные операции.

**Вертикальное упорядочение:** Как правило, взаимозависимые функции должны размещаться в нисходящем порядке . Иначе говоря, вызываемая функция должна располагаться ниже вызывающей функции . Так формируется логичная структура модуля исходного кода – от высокого уровня к более низкому . Как и в газетных статьях, читатель ожидает, что самые важные концепции будут изложены сначала, причем с минимальным количеством второстепенных деталей .