Ниже перечислены критерии, которым должен удовлетворять код программы, чтобы задача была принята, и приведена аргументация, почему так надо делать.

# Общие рекомендации

1. Код должен быть выровнен.  
    **Аргументация:** так гораздо лучше читается, четко видна вложенность кода.  
   Можете выравнивать самостоятельно (рекомендую), либо использовать Ctrl+Alt+L
2. Нигде не должно быть 2 и более пустых строк подряд  
    **Аргументация:** код выглядит неаккуратно, это ухудшает читаемость
3. Для читаемости можно вставлять пустые строки в середине функции, чтобы отделять логические части программы.

**Пример без пустых строк – читается плохо:**Scanner scanner = new Scanner(System.in);  
System.out.print(“Введите первое число: ”);  
int a = scanner.nextInt();  
System.out.print(“Введите второе число: ”);  
int b = scanner.nextInt();  
int result = a \* b;  
System.out.println(result); **Пример с пустыми строками – отделили создание сканнера, чтение данных и получение результата:**Scanner scanner = new Scanner(System.in);  
  
System.out.print(“Введите первое число: ”);  
int a = scanner.nextInt();

System.out.print(“Введите второе число: ”);  
int b = scanner.nextInt();  
  
int result = a \* b;  
System.out.println(result);

1. Программа должна быть без warning’ов (предупреждений компилятора).

**Аргументация:** warning’и говорят о том, что либо в коде есть потенциальная ошибка, либо есть неиспользуемый код, либо код написан неоптимально. В любом случае это нужно исправлять.

IDEA выделяет warning’и желтым цветом фона, иногда бледным цветом, если это неиспользуемый код. На полосе прокрутки текстового редактора warning’и отмечены желтыми полосками.

1. Не должно быть пустых строк сразу после { и перед }.

**Аргументация:** это самое начало или конец блока кода, поэтому там нет смысла ставить пустые строки.  
Код выглядит неаккуратно.

**Плохой пример:**

if (a > 3) {  
  
 b = 3;  
  
}

**Хороший пример:**

if (a > 3) {  
 b = 3;  
}

# Имена переменных, функций и классов

1. Имена переменных и функций в camel case с маленькой буквы. Имена классов в camel case с заглавной буквы.  
    **Аргументация**: такой стиль сложился исторически для Java (и некоторых других языков), и все его придерживаются. По-другому код никто не пишет.  
     
   Примеры:  
   int countOfPeople = 5;  
     
   public double getSum(double x, double y) {  
    return x + y;  
   }  
     
   public class InsertionSort {  
   }
2. Имена переменных, классов и функций должны быть полными и понятными. Не допускаются сокращения и транслит. Короткие имена допускаются только для счетчиков циклов i, j и т.д.  
    **Аргументация:** это сильно повышает читаемость кода. Печатать длинные названия – это быстро, потому что есть автодополнение по первым введенным символам. Для счетчиков – так исторически сложилось, и сам счетчик обычно не несет смысловой нагрузки.
3. Имена переменных и классов – это существительные. Имена функций должны начинаться с глаголов  
     
   **Аргументация:** переменные и классы – это некоторые сущности (существительные), а функции – это действия.
4. Имена функций, которые выдают boolean, должны начинаться со слова is (если больше подходит, то has, need и др.).  
   Аналогично для переменных типа boolean.  
     
   **Аргументация**: так подходит по смыслу, и так принято  
     
    **Пример:**  
   public static boolean isPositive(int x) {  
    return x > 0;  
   }  
     
   boolean isPrime = true;
5. Имена void функций не должны начинаться с глагола **get**, т.к. void функции ничего не выдают. Имя должно начинаться с глагола, который соответствует тому, что делает функция.

Например, если функция печатает, то должен быть глагол **print**, если сортирует, то **sort**, если что-то преобразовывает, то **convert** и т.д.  
  
**Аргументация**: так правильно по смыслу и с точки зрения английского  
  
**Плохой пример:**  
public static void getSort(int[] array) {  
 // тут идет код сортировки  
}

**Хороший пример:**public static void sort(int[] array) {  
 // тут идет код сортировки  
}

# Область видимости переменных

1. Переменные нужно объявлять в месте первого присваивания (если, конечно, у неё не должна быть большая видимость).  
    **Аргументация:** так программу легче читать и анализировать.   
   До первого присваивания всё равно переменная не используется. Тогда зачем объявлять её раньше?   
   И тогда весь код, относящийся к одной переменной, будет находиться ближе друг к другу.  
     
   **Плохой пример 1:**double x;  
   x = scanner.nextDouble();  
     
   **Хороший пример 1:**  
   double x = scanner.nextDouble();

**Плохой пример 2:**double x;  
// куча кода, который не работает с x  
x = 3;

**Хороший пример 2:**// куча кода, который не работает с x  
double x = 3;

# Ветвления, циклы

1. Всегда нужно ставить фигурные скобки в if/else и циклах, даже если внутри одна команда.  
     
   **Аргументация:** так лучше читается и нет возможности случайно ошибиться вроде такого.  
   int a = 3;  
   int b = 2;   
   if (a > 0 && b > 0)  
    System.out.println(“a > 0”);  
    System.out.println(“b > 0”);  
   Здесь строка “b > 0” будет печататься всегда, даже если это не так.
2. Если проверки взаимоисключающие, то последний else if нужно заменять на else.

**Аргументация:** тогда в коде станет на одну проверку меньше для худшего случая. В некоторых случаях без этого возможна ошибка компиляции (если внутри каждой ветки делается return из функции).

**Плохой пример:**

if (a > 0) {  
 System.out.println(“a > 0”);  
} else if (a < 0) {  
 System.out.println(“a < 0”);  
} else if (a == 0) {  
 System.out.println(“a == 0”);  
}

**Хороший пример:**if (a > 0) {  
 System.out.println(“a > 0”);  
} else if (a < 0) {  
 System.out.println(“a < 0”);  
} else {  
 System.out.println(“a == 0”);  
}

1. После веток с break/continue/return/throw не нужно писать else.

**Аргументация:** код станет проще читаться при том же самом поведении. Во-первых, кода станет меньше. Во-вторых, в цепочке if’ов или в if-else последняя ветка будет идти без фигурных скобок. Это читается намного лучше.

**Плохой пример:**

if (a < 0) {  
 return 0;  
} else {  
 /\* здесь идет большое количество строк кода \*/  
}

**Хороший пример:**if (a < 0) {  
 return 0;  
}  
/\* здесь идет большое количество строк кода \*/

Этот код ведет себя так же, как с else, но зато сам else писать не нужно.

И сам код идет без фигурных скобок.

Код, который сдвинут на табуляцию, читается хуже.

А здесь получается на 1 табуляцию меньше.