Краткая теоретическая справка

Сигнатура метода в Java — это имя метода плюс параметры (причем порядок параметров имеет значение). В сигнатуру метода не входит возвращаемое значение, бросаемые им исключения, а также модификаторы. Ключевые слова public, protected, private, abstract, static, final, synchronized, native, strictfp в т.ч. аннотации для метода — это модификаторы и не являются частью сигнатуры.

**Перегрузка**

Создание метода с тем же именем, но с другим набором параметров называется перегрузкой (overload). Перегрузка обычно используется, чтобы не придумывать каждый раз новое имя, когда методы отличаются только входными параметрами.

**Пример 1.**

**public** **int** sum (**int** x1, **int** x2) {

**return** x1+x2;}

**public** **int** sum (**int** x1, **int** x2, **int** x3) {

**return** x1+x2+x3;}

Все работает нормально – метод sum перегружен.

**Пример 2.**

**public** **int** sum (**int** x1, **int** x2) {

**return** x1+x2;}

**public** **int** sum (**int** y1, **int** y2) {

**return** y1+y2;}

Не работает т.к. сами параметры не изменились, а имя параметра не имеет значения.

**Пример 3.**

**public** **int** sum (**int** x1, **int** x2) {

**return** x1+x2;}

**private** **int** sum (**int** x1, **int** x2) {

**return** x1+x2;}

**public** **static** **int** sum (**int** x1, **int** x2) {

**return** x1+x2;}

Не работает т.к. сами параметры не изменились, а изменение модификатора не перегружает метод.

**Пример 4.**

**public** **int** sum (**int** x1, **int** x2) {

**return** x1+x2;}

**public** **long** sum (**int** x1, **int** x2) {

**return** x1+x2;}

Не работает т.к. сами параметры не изменились, а изменение типа возвращаемого значения не перегружает метод.

**Пример 5.**

Различные комбинации sum (**int** x1, **int** x2) и sum (**int** x1, **int** x2, **int** x3) с различными модификаторами и типом возвращаемых значений работают, так как различны входные параметры.

**Вывод. При перегрузке**

**-** не должно быть двух методов с одним и тем же списком параметров **–** методы должны иметь новый список параметров, при этом имя параметра не имеет значения;

**-** могут иметь новый тип возвращаемого значения;

**-** могут иметь новый уровень доступа без ограничений;

**-** вроде могут бросать новые exception'ы, но мной не проверялось :)

**Переопределение (override)**

Метод в классе-наследнике, совпадающей по сигнатуре с методом из родительского класса называется переопределенным методом. Переопределение используется для уже существующего метода. Переопределение метода выполняется только в том случае, если имена и сигнатуры типов двух методов идентичны **-** в противном случае два метода являются просто перегруженными

Когда переопределённый метод вызывается из своего подкласса, он всегда будет ссылаться на версию этого метода, определённую подклассом. А версия метода из суперкласса будет скрыта. Если нужно получить доступ к версии переопределённого метода, определённого в суперклассе, то используйте ключевое слово ***super***.

Переопределяемые методы необходимо предварять аннотацией @Override, и компилятор получает возможность проверить, что метод переопределен и выдаст сообщение об ошибке, если вместо переопределения будет случайно выполнена перегрузка.

**Пример** из домашнего задания HomeTask4

//переопределение метода toString

@Override

**public** String toString() {

**final** StringBuilder sb = **new** StringBuilder("Bus{");

sb.append("Ф.И.О. водителя: ").append(driverName);

sb.append(", № автобуса ").append(busNumber);

sb.append(", № маршрута").append(routeNumber);

sb.append(", Марка: ").append(brend);

sb.append(", Год начала эксплуатации ").append(year);

sb.append(", Пробег ").append(mileage).append(" км.}");

**return** sb.toString();

}

Здесь переопределен метод toString из базового класса Object.

**Вывод. При переопределении**

**-** методы должны иметь неизменный список аргументов;

**-** методы должны иметь неизменный тип возвращаемого значения (но можно сузить возвращаемое значение, если они совместимы);

**-** нельзя сузить модификатор доступа к методу (например с *protected* до *private*), но можно расширить (например с *protected* до *public*),

**-** методы не должны бросать новых exception'ов;

**-** переопределить статический метод нельзя;

**-** методы конструкторы не могут быть переопределены— только перегружены.