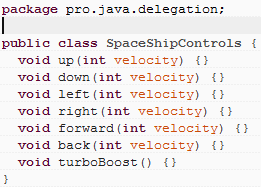
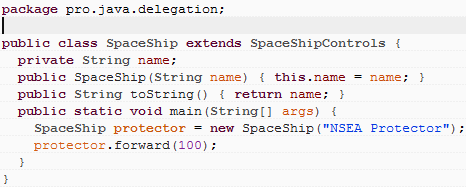
**Делегирование.**

Еще один вид отношений, не поддерживаемый в Java напрямую, называется делегированием (delegation). Он занимает промежуточное положение между [наследованием](http://pr0java.blogspot.ru/2015/07/blog-post.html) и [композицией](http://pr0java.blogspot.ru/2015/07/blog-post_1.html): экземпляр существующего класса включается в создаваемый класс (как при композиции), но в то же время все или некоторые методы встроенного объекта становятся доступными в новом классе (как при наследовании). Очень часто такие виды отношений используются при построении графического интерфейса, например для реализации модели MVC библиотека Swing использует делегирование.

А сейчас рассмотрим на простом примере что же это за зверь такой – делегирование.

Например, класс **SpaceShipControls** имитирует модуль управления космическим кораблем. А Для построения космического корабля можно воспользоваться наследованием в классе **SpaceShip**.

[](https://lh3.googleusercontent.com/-Dzobb8zrDiU/VuwDsjn8byI/AAAAAAAAPWY/KYAYth-wyPA/s1600-h/SS001%255B2%255D.png)[](https://lh3.googleusercontent.com/-4z8ugVSn6rk/VuwDtrp0XmI/AAAAAAAAPWg/ujKW1iYkosU/s1600-h/SS002%255B2%255D.png)

Однако космический корабль не может рассматриваться как частный случай своего управляющего модуля — несмотря на то, что ему, к примеру, можно приказать двигаться вперед (forward()). Точнее сказать, что **SpaceShip**содержит **SpaceShipControls**, и в то же время все методы последнего предоставляются классом **SpaceShip**. Проблема решается при помощи делегирования:

[](https://lh3.googleusercontent.com/-w89trfvJrUc/VuwE5Ih_B1I/AAAAAAAAPWw/grN1fTizcvQ/s1600-h/SS003%255B2%255D.png)

Как видите, вызовы методов переадресуются встроенному объекту **controls**, а интерфейс остается таким же, как и при наследовании. С другой стороны, делегирование позволяет лучше управлять происходящим, потому что вы можете ограничиться небольшим подмножеством методов встроенного объекта.