В тот самый момент, когда вы, известнейший специалист по паттернам проектирования, заходите в зал заседаний корпорации «МегаГигаКо», генеральный директор и члены совета директоров празднуют свой контракт на проектирование новой серии автомобилей в самом спокойном виде, каком только можно было бы себе это представить – высокими ударами ладоней, беготнёй по залу и криками на всю катушку.

«Этот контракт сулит огромный доход для нашей компании!», выкрикивает директор, слегка расплескав шампанское на стол из-за избытка эмоций. «Всё что нам нужно сделать, это убедиться, что мы понимаем процесс проектирования ВЕРНО.» Он поворачивается к проектору, и когда на стене появляются несколько больших диаграмм, директор продолжает: «Итак, вот моя идея…»

«Неверно понимаете», громко говорите вы.

Директор «МегаГигаКо» выглядит удивлённым и пытается продолжить: «Но если мы…»

«Нет», говорите вы, качая головой.

«Что…»

«Извините», киваете вы директору и его прекрасным диаграммам: «это очевидно, что вы рискуете всем своим контрактом, потому что делаете всё неправильно. Я вижу у вас дюжину проблем просто посмотрев на вот эту диаграмму.»

Совет директоров недовольно шумит, и директор переходит в наступление: «Что насчёт вашей нескромной персоны?»

«Я и есть профессионал паттернов проектирования, который решит все ваши большие проблемы проектирования», сделав небольшую, но эффектную паузу, вы продолжаете: «за большую плату, разумеется».

Директор поспешно записывает предварительную цифру оплаты ваших трудов, которая, несмотря на свою величину, всё же не кажется вам достаточно большой.

«И опять неверно понимаете», произносите вы.

Директор смотрит на вас, приподняв брови.

«Паттерны проектирования», терпеливо объясняете вы, «представляют собой решения известных проблем программирования. Но не только это, они также представляют собой хорошую программистскую практику, которая позволяет гораздо легче поддерживать и дополнять ваш код. Так что, как вы можете заметить, привлечь такого эксперта как я имеет большой смысл – ибо когда я вижу проблему, которая уже решена с помощью паттерна проектирования, я могу рассказать о ней всё.

«Ну», осторожно говорят программисты компании «МегаГигаКо», «идея паттернов выглядит хорошо, но мы уже пользуемся методами объектно-ориентированного программирования в наших разработках. Разве это не покрывает наши проблемы?»

«Нет», говорите вы в очередной раз. На самом деле, это один из моментов, лежащих в основе паттернов проектирования – они расширяют объектно-ориентированное программирование.

# Композиция(агрегация) против наследования: первая попытка разработки новых машин.

Пресловутые программисты корпорации «МегаГигаКо» знают о наследовании, и они начали разработку новых машин, наплевав на ваши предупреждения подождать встречи с вами. Предполагалось, что они разработают серию машин, так что они стартовали с создания базового класса Vehicle и метода с названием go, который отображает текст «Сейчас я еду.».

public abstract class Vehicle

{

public Vehicle()

{

}

public void go()

{

System.out.println(“Сейчас я еду.”);

}

}

Затем они создали ещё классы, такие как StreetRacer, используя класс Vehicle как базовый класс и применив наследование:

public class StreetRacer extends Vehicle

{

public StreetRacer()

{

}

}

Всё идёт по плану. Если создать класс StreetRacer и запустить его вот так:

public static void main(String[] args)

{

StreetRacer streetRacer = new StreetRacer();

streetRacer.go();

.

.

.

}

То мы увидим на экране:

Сейчас я еду.

Неплохо выглядит. Так здорово, что корпорация «МегаГигаКо» решила следовать этим курсом и сконструировала болид «Формулы-1», который также наследует класс Vehicle как вы можете увидеть ниже:

public class FormulaOne extends Vehicle

{

public FormulaOne()

{

}

}

И вы можете запустить и Стритрейсера и болид Формулы -1 одновременно вот так:

public static void main(String[] args)

{

StreetRacer streetRacer = new StreetRacer();

FormulaOne formulaOne = new FormulaOne();

streetRacer.go();

formulaOne.go();

.

.

.

}

И мы получаем:

Сейчас я еду.

Сейчас я еду.

«Неплохо», говорит директор совместно с советом директоров. «Кому нужны паттерны проектирования?», спрашивают они, бросая на вас невежливые взгляды. Но затем они получают контракт на производство вертолётов. Вертолёты, замечают они, это просто ещё одна разновидность машин. Так что они создают вертолёты, используя класс Helicopter, расширяя класс Vehicle:

public class Helicopter extends Vehicle

{

public Helicopter()

{

}

}

Но тут появляется проблема. Если запустить вертолёт наряду со стритрейсером и болидом формулы-1 вот так:

public static void main(String[] args)

{

StreetRacer streetRacer = new StreetRacer();

FormulaOne formulaOne = new FormulaOne();

Helicopter helicopter = new Helicopter();

streetRacer.go();

formulaOne.go();

helicopter.go();

.

.

.

}

То мы получим по сообщению от каждого аппарата.

Сейчас я еду.

Сейчас я еду.

Сейчас я еду.

Это выглядит как-то неправильно, с сомнением в голосе произносит директор. Почему вертолёт должен «ехать»? Разве он не должен «лететь»? И проблема становится ещё больше, когда «МегаГигаКо» получает контракт на производство самолётов, которые опять расширяют базовый класс Vehicle.

public class Jet extends Vehicle

{

public Jet()

{

}

}

Соответственно, при запуске всех аппаратов, мы получаем:

Сейчас я еду.

Сейчас я еду.

Сейчас я еду.

Сейчас я еду.

«Вот это точно полная ерунда», произносит директор, «самолёты не едут, они в воздухе. Они летают – и причём весьма быстро».

«Нет проблем», говорят программисты корпорации. «Мы можем переопределить метод go в классах Helicopter и Jet чтобы дать им соответствующее поведение. И они создают что-то вроде вот такого, что заставит вертолёт летать, а не ехать:

public class Helicopter extends Vehicle

{

public Helicopter()

{

}

public void go()

{

System.out.println(“Сейчас я лечу.”);

}

}

«Okay», говорит директор, «но совет директоров уже проголосовал за изменение сообщения на следующую неделю с «Сейчас я лечу.» на «Сейчас я лечу со скоростью 200 миль в час.». И, насколько я знаю совет, изменений ещё будет много.»