*В лесу родилась ёлочка,*

*В лесу она росла.*

*Зимой и Летом – Стройная, Зелёная*

*Была.*

**БРАЖИНСКИЙ**

1. Проектирование
   1. IFirTree
2. «Ель – род деревьев...» (с) Wiki – у нас род начинается и заканчивается на ели;
3. Hight – это высота про которую Высоцкий пел. У дерева (рост) – Height;
4. IFirTree.GrowFirTree() – Grow() – достаточно;
5. string Form/Color – напрашиваются enums;
6. { get; } – ель-интраверт.
   1. IForest
7. AddFirTree(IFirTree) – предположим лес сосновый, но он тогда должен быть IPineForest;
8. WinterIncrease, SummerIncrease – некорректное поведение. Это переводится как “Лес торкал ёлочку в рост зимой и летом». О сезонах тоже лесу не обязательно знать. А точнее он за них не отвечает. Лес может находится в ‘Состоянии' Сезона – сущность которого не определена.
9. ShowHightFirTree() – магический метод.
10. Тестирование
    1. FirTreeTest

a)TestGrow - не проверяется на отрицательный параметр;

b)TestColor, TestForm – не проверяется состояние в зависимости от сезона. Hardcoding.

2.2 ForestTest

a) TestAddNewFirTreeInForest() – не даёт никакой гарантии факта рождения данной конкретной ёлки в данном конкретном лесу. AddFirTree() может быть захардкоджен на возвращение true. Нужно проверять экземпляр ели в коллекции леса.

b) TestIncrease – до реализации ещё не добрались, стало понятнее что-это, но 1.2.b – сомнительное поведение, за рамками предметной области, с нарушением логики поведения.

с) TestShowHightFirTree() – ГОСТ для Новогодней ёлки ☺

1. Реализация.
   1. FirTree
2. Form/Color – см. выше – ёль-интраверт, hardcoding. Можно было обойтись константами.
3. GrowFirFilter() – реализована возможность постепенного уменьшения в размерах. Там дальше по предметной области максимум «под корешок».

3.2 Forest

a) AddFirTree() – <http://insidecpp.ru/antipatterns/premature_optimization/>, изначально неоправданно ограниченный функционал.

В лесу родилась ёлочка, а вдруг еще одна ? ☺

b) WinterIncrease, SummerIncrease – magic numbers

c) ShowHightFirTree – black box.

**ЕРМОЛОВ**

1. Проектирование
   1. Tree
2. Почему не интерфейс ? Собственно, в целом – где интерфейсы ?
3. Length – деревья не длинные, деревья высокие;
4. Color – enum, bits, RGB, встроенные типы...
5. Shapely – следуя логике данного свойства, вместо Color должно быть [bool Green] - неоправданное ограничение характеристик сущности.
6. Type – вот здесь мы имеем смешение фабричного и шаблонного метода, или вообще внедрение сильной типизации. В случае типизации класса самим собой, у нас нет необходимости в фабричном методе и в иерархии деревьев. Можно обойтись одним деревом, выставляя его состояние. Опять же – абстрактный класс (будем считать его интерфейсом), сразу знает о своих возможных реинкарнациях –ошибка проектирования. Назначение свойства Type на данном этапе является неясным, тем более в С# достаточно средств для динамической типизации объектов.
7. Реализация

Всвязи с отсутствием интерфейсов, переходим сразу к реализации, минуя TDD (судя по тайм-стемпам в заголовках файлов).

* 1. TreeCreator

1. фабричный метод, да – правильно. Но здесь в идеале и должен находится словарь деревьев;
2. Oak, Birch – superfluous cases, all goes to default:
   1. FirTree
3. 1.1.e – Tree firTree => new Tree(TypeofTree.Fir,”Green”);
4. utilize auto-properties
   1. SomeAbstractTree
5. дерево-мутант;
   1. Forest
6. Forest.Forest;
7. Season.None – ядерная зима ? => new Exception()
8. Лес не может менять времена года. В данном случае Season является глобальным состоянием, а Лес – подписчиком на изменения данного состояния: Singelton, Events, Subscribers, Observer
9. Лес – коллекция деревьев. Ок.
10. GetSeason2(): if (month == 12 && month < 3) ???
11. Тестирование
    1. «Недружелюбный» деплоймент. Вы помоему через адаптер и расширение к вижуалу должны деплоить тесты.
    2. ForestCreateTest
12. Stability. Летом тесты пофейлятся... Тесты не должны зависить от внешнего енвайрмента, они тестят модель – соответственно и состояния должны быть модельные или моковые (2.4.с)

3.3 AddFirTest

* 1. Статическая типизация... Это ёлочка, а наша ли ?
  2. FirColorTest – да, но опять же 2.4.с
  3. Вообщем ясно, тестятся свойства сосновых, а не конкретно нашей ёлки. Имеет право быть.
  4. Лес тоже можно было протестить, вырастить-вырубить ёлку.

**СЫСОЕНКО**

1. Проектирование
   1. ITree
2. enums Color, Shape: более культурно выносить отдельно: в Forest.Constants, Forest.Primitives, Forest.Enums и тд и тп.;
3. CountOfDays() – звучит удручающе ☺ как обратный отсчёт... Кукушка-кукушка, сколько мне дней осталось ☺
4. TreeShape, DisplayTree – немного тафтологии;
5. DisplayTree() – посмотрим, что оно делать будет; может ёлка с монитором... ггг
   1. IForest

Что ? Где ? Когда ?

1. Тестирование
   1. TreeTests.Options...
2. в лесу родилось только Дерево судя по всему;
3. реализация, в которой в качестве параметра конструктора передается Сезон не соответствует реалиям проектной модели; Сезон глобальное явление ареала обитания дерева. В жизненном цикле объекта tree сезон может меняться – данный механизм не реализован.
   1. TreeTests.DieEvent\_Work\_Called
4. CountOfDays = 0; мрачно, точно последний отсчёт ☺
   1. MyForestTests

а) 2.1.b – (я держался) похоже на Мафию ☺ В лесу настает Лето...

b) роль леса в качестве наблюдателя в принципе ясна, но в данном случае актуальным становится фразеологизм «не видеть леса за деревьями». Роль леса ошибочна. Так можно и в поле вырасти.

1. Реализация
   1. Tree
2. сразу, был относительно чистый ITree интерфейс, отражающий сущность предметной области, и мгновенно реализация интерфейса превращается в какую-то оргию ☹
3. что делает в дереве Random ? У меня елка на НГ стояла, я не видел на ней генераторов случайных чисел, календарь тоже на ней не висел.
4. heigh, width – не используется;
5. interval, growDays – fields with magic numbers
6. CountOfDays – use auto-property
7. Всё что со словом DateTime и Random, всё неправда и лишнее
8. bool CheckSeason() – зачем его проверять ?
9. «Зимой и Летом» это фразеологизм – означает всегда.
10. Options() – методы это отражение действия, именуются соответственно. Options – существительное, отражающее состояние.
11. Grow() vs countOfDays – очень мрачно... Хотя в продолжении она не своей смертью умерла – ее маньяк зарубил, еще мрачнее.
12. DisplayTree() – допустимо когда объект себя дампит, но как бы ёлка ☺
13. Left Days: -34 ☺ И я не видел тестов для этого алгоритма...

*Time of Born: 2015-02-14 00:00:00*

*Abies was born in Forest!*

*Abies died in Forest!*

*|Season: Summer | Shape: Slender | Color: Green|*

*Left Days: -34*

* 1. Forest

1. «Семён Семёныч» (с) – оказываются в лесу таки растут елки, и лес является коллекцией – только тссс, это тайна ☺ Та и тайна такая себе – коллекция public. Сажай, вырубывай – не хочу.
2. Createtree()

В лесу родилась ёлочка, а кто ее родил, - 4 пьяных ёжика и Гена крокодил (с)

Фабричный метод – хорошо, но не лес ёлочку рожает. Глагол «родилась», в собственной форме.Значит, кто-то.

* 1. Messanger

1. -> Messenger
2. Нашёл объект в дереве, точно Кукушка ☺

**СЫРНИКОВ**

1. Проектирование
   1. ITree
2. enums => TreeSettings.cs
3. Height – бинго ! ☺
4. Type – см. [Eрмолов] 1.1.e
   1. IForest
5. ChangeSeason, CurrentSeason - лес не контролирует погоду.
6. Тестинг
   1. InWinterGreenAndShapelly – black-box w/ seasons change events, почему ёлка зимой родилась?
   2. InSummerGreenAndShapely – кто сказал, что после двух смен сезонов – наступило лето ? где-то выше, про то что Сезоны года – отдельная глобальная сущность... могующая быть реализована сингелтоном и событийным программированием
   3. ForestTests – см.1.2.а
   4. Тесты на интерфейсах проходить должны, коль скоро они определены.
   5. UseCase - OK
7. Реализация
   1. AbstractTree
8. GrowingSpeed – auto-property; Google => GrowthSpeed;
9. GrowUp() { GrowUp(growingUp); }
10. default: throw Exception
    1. FirTree
11. Base interface ITree is redudant ©
12. ChangeColor() => may be empty, but may be not.
    1. Forest
13. GetTree() – можно было и над LINQ помедитировать
14. 1.1.с про Type – в принципе ясно.
15. ChangeSeason() – [Зима/Лето] - это в сериале такое было помню, «Не родись красивой» ☺

**ПАВЛЮЧЕНКО**

1. Проектирование
   1. ITree
2. Нужно бросить писать комменты, и забыть про русский язык в коде.
3. Colour – British english
4. Condition ☺ На состояние авто похоже ☺
   1. IForest

Скучный интерфейс, опять за времена года лес отвечает, и фсё.

1. Тестирование
   1. Комменты => self-documented code
   2. Смотрю лес почему-то (сутенёром, тьфу) контроллером ёлок выступает – идём в реализацию.
2. Реализация
   1. FirTree
3. Почему свойства не в интерфейсах, чтобы в тестах с интерфейсом работать ?
4. Зачем дереву ссылка на лес ?
   1. Forest
5. Лес состоит из одной ёлочки... Это не лес ☺
6. IfExistGrows – методы не начинаются с If, и вообще в названиях методов не гламурно If смотрится. Метод с глагола начинается. Акцентировать в названии метода его безопасность тоже не обязательно... в крайнем случае SafeGrowth.
7. Идею с лес-контроллер-адаптер к ёлке я не понял. Достаточно вернуть её экземпляр.

**ОБЩЕЕ**

1. Не используется механизм исключений.

2. Лес почему-то управляет временами года.

3. Соответственно не реализована сущность Сезона, с возможностью использования событийного прогарммирования.

4. Русский язык, семантика, наименования методов.

5. Статическая типизация. Присутствие идентификатора типа в реализации сомнительна. Сосну в лесу и без этого найти можно. Особенно если в лесу одно дерево растёт ☺ Или лес сосновый.