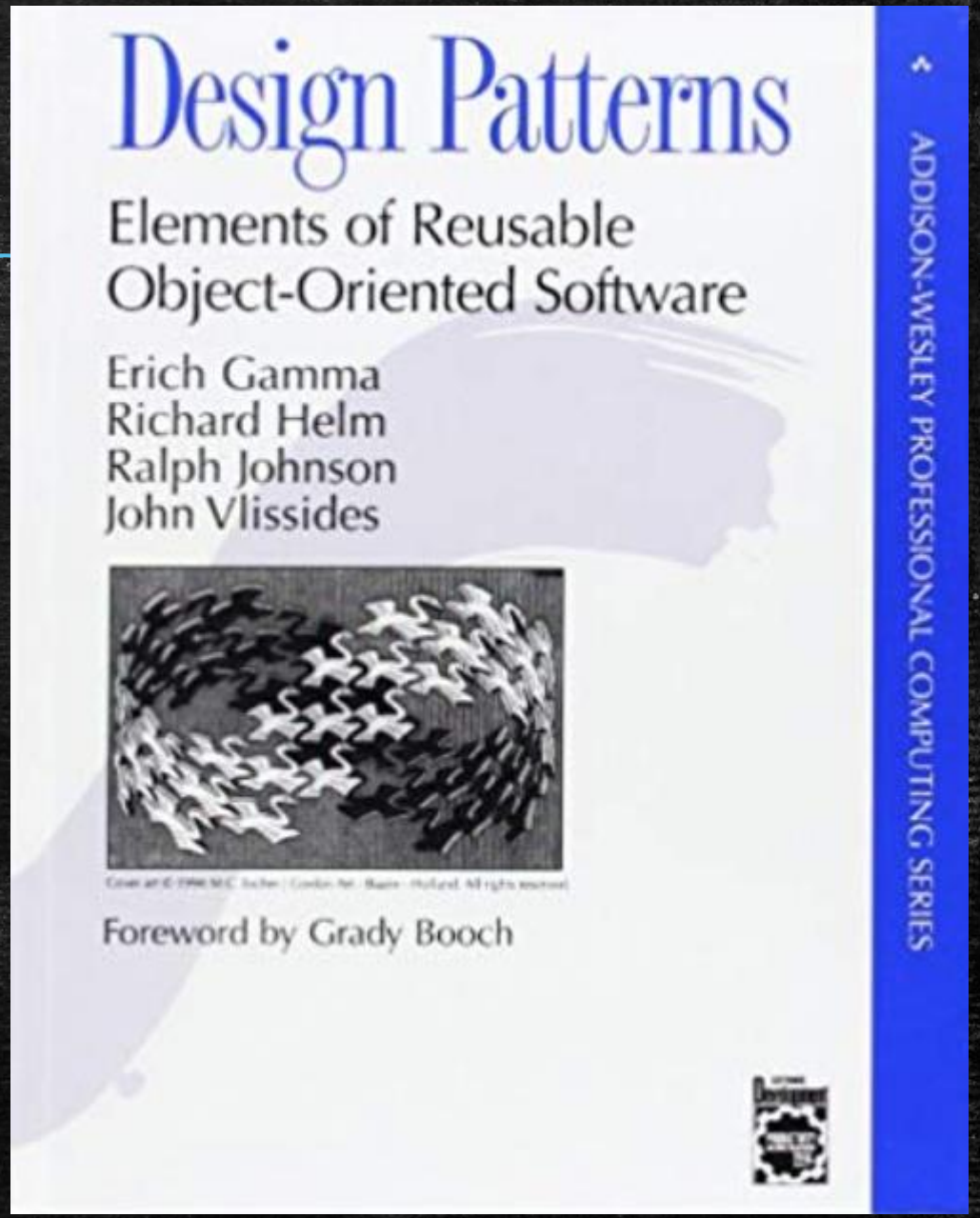
Напредни Пајтон

Принцип Шаблони Дизајна

Увод

- 23 чувена шаблона дизајна
- Осмислила Банда од четири – посљедица *најбоље праксе*
- Пајтон има своје специфичности



- Имамо класу која врши испис. Не смијемо је мијењати и морамо је користити.
- Потребно је исписивати садржај једноставних објеката у облику рјечника или json објекта, користећи искључиво класу Ispis

```
class Ispis():
    def __init__(self, tekst):
        self.tekst = tekst

    def ispis(self):
        print(self.tekst)
```

- Проширити класу Animal тако да садржи метод за перзистенцију, али да се може бирати која је перзистенција у питању: штампање на екран, упис у фајл или упис у базу података

- Шаблон који третира једноставне и комплексне објекте на јединствен начин.
- Направити Зоолошки врт од свих животиња, груписати их у више група, неке појединце одвојити од групе.

- Додатно понашање које се може додат постојећем понашању без промјене постојећег кода
- Логовање времена извршења функције, без промјене функције

- Обезбиједити једноставан и разумљив опис и начин кориштења иза којег се налази веома комплексан програмски код
- Сакривање комплексности и једноставан АПИ
- ХИНТ, кад ти клијент каже: *Дај ти мени то поједностави, да ја само кликнем…*

- Проширити дио гдје су дефинисане све животиње тако да постоје додатне информације о свакој врсти НА НИВОУ ОБЈЕКТА (не класе)
- Уштедити на меморији док додајемо ове информације

- Не смијем дирати постојећу класу, али јој морам додати додатна ограничења или додатно понашање
- Техничка особа је класа која има особину-сениоритет и којој се додијели *Issue.* Issue је такође класа која има тежински фактор као особину.
- Имплементирати, без промјене ових класа: Особа са сениоритетом ЈУНИОР не може да добије *Issue* са највишим тежинским фактором.

```python
class Person:

    def __init__(self, name, seniority):
        self.name = name
        self.seniority = seniority
        self.issue = None

    def addIssue(self, issue):
        self.issue = issue
```

```python
class Issue:
    def __init__(self, number, dif):
        self.number = number
        self.difficulty = dif

    def __str__(self):
        return f'Issue {self.number}, difficulty {self.difficulty}'
```

- Обезбиједити само једну иницијализацију података.
- Класа не може имати више од једне инстанце, без обзира колико пута се инстанцира
- YML фајл је извор података за *Issue* – једини и само једном се учитава.
- Класа која учитава ове податке је само једна

```
retval = []
with open("issues.yml", 'r') as f:
    issues = list(yaml.load_all(f, Loader=yaml.SafeLoader))
    for issue_ in issues[0]["issues"]:
        retval.append(Issue(issue_['number'], issue_['difficulty']))
```

```
issues:
- number: 122
  difficulty: MEDIUM
- number: 124
  difficulty: EASY
- number: 123
  difficulty: HARD
```

- У случају компликоване конструкције објекта:
  - Не може се креирати у једном једноставном конструктору
  - Постоји више од 6 аргумената конструктора (некоме смета и мање)
  - Постоји цијела церемонија око креирања објекта, јер се сви подаци не познају одмах

```
class Issue:
    def __init__(self, project):
        self.project = project
        self.description = None
        self.person = None
        self.number = None
        self.status = "new"
        # other: numbered, assigned,
                    open, in work, closed
        self.reviews = []
```

- Имамо само један конструктор, а више различитих конструкција објекта
- Обезбиједити креирање животиње Animal и на основу json објекта и на основу dict и на основу стринга вриједности одвојених зарезом (name, legs, species)

- Када су објекти толико комплексни да је потребно да их клонирамо и само дјелимично измјенимо.
- Имамо класе канцеларија, адреса, запослени. Канцеларија садржи спрат и назив/број канцеларије, адреса садржи канцеларију адресу и град, а запослени садржи адресу и име и презиме.
- Међутим, постоје само двије адресе: Вука Караџића и Јована Дучића. Креирати запослене тако да дјелимично буду креирани, а онда само додатно измјенимо

- Направити систем за ривју *issue*-а на пројекту. Свака техничка особа може да напише ривју за одређени *issue.*
- Да би особа писала *issue* мора му бити додијељен. Или особа мора бити додата на issue.
- *Issue* служи да свакога ко ради на њему обавијести о датом ривјуу. Саме техничке особе нису директно свјесне једна друге.

- Jenkins има сљедеће кораке извршавања, од којих сваки може да прође или да падне. У случају пада посао се зауставља
  - Статичка анализа кода
  - Билд
  - Јединични тестови
  - Паковање
  - Диплој
- Направити систем који ово подржава

- Класа представља инструкцију која треба да изврши одређену активност. Објекат садржи све информације потребне да се акција изврши.
- Постоји листа ишјуа којима је могуће мијењати статусе, а приликом промјене упише се опис, особа која је промијенила, нови статус и тренутно вријеме. Ипак могуће је и да се уради UNDO!

- Направити програм који омогућава да стринг са исправно написаном математичком операцијом која садржи цијеле бројеве, множење, дијељење, као и заграде, односно груписање бинарне операције – евалуира у израчунату вриједност

- У бинарном стаблу прођи кроз све елементе у in-order (или preoder, postorder)

- Омогућити функционалност враћања у неко стање објекта.
- Послије неколико промјена на банковном рачуну, вратити се у неко претходно стање, или омогућити враћање у било које стање на банковном рачуну.

- Када се промијени нека особина објекта, сам објекат уради нешто што изазове да се покрене неки екстерни догађај
- Неко слуша догађаје, па када се догђај деси, низ активности се изврши.
- Омогућити зоолошком врту да свака животиња може да се разболи и да се аутоматски позове доктор и обавијести директор зоолошког врта

- Уколико имамо потребу да динамички мијењамо понашање објекта у зависности од његовог стања.
- Issue има три статуса: new, open, closed – омогућити да се у сваком од та три стауса понаша другачије. Могуће је из било којег статуса прећи у било који други статус. Понављање статуса ништа не ради, можда само да јави да је већ у том статусу.

- Друга страна медаље у односу на шаблон креирања објекта на различите начине из различитих извора.(factory) На неки начин.
- Омогућити креирање Ишјуа који може имати различите начине BURN-DOWN евалуације. Ишју ће бити 100% завршен након естимираног броја дана, али ће се проценти мијењати другачије кроз дане. Начини ове евалуације могу бити линеарни или неки необични (нпр првих 10% дана, ништа, а онда ...) Избор која ће функционалност бити извршена не зависи ни од којег података из објекта Ишјуа, него се просљеђује при креирању.

- Koji je шаблон дизајна имплементација метода _fly_ и fly у класама Bird и Penguin, Crow?

- Направити могућност да особа може да ради на ишјуу или да ривјујује ишју, без промјене класе особа.
- Како би се имплементирало штампање, без промјене класе Animal?
- У овом случају имамо исту класу чије ће различите инстанце бити обучене у различите улоге.