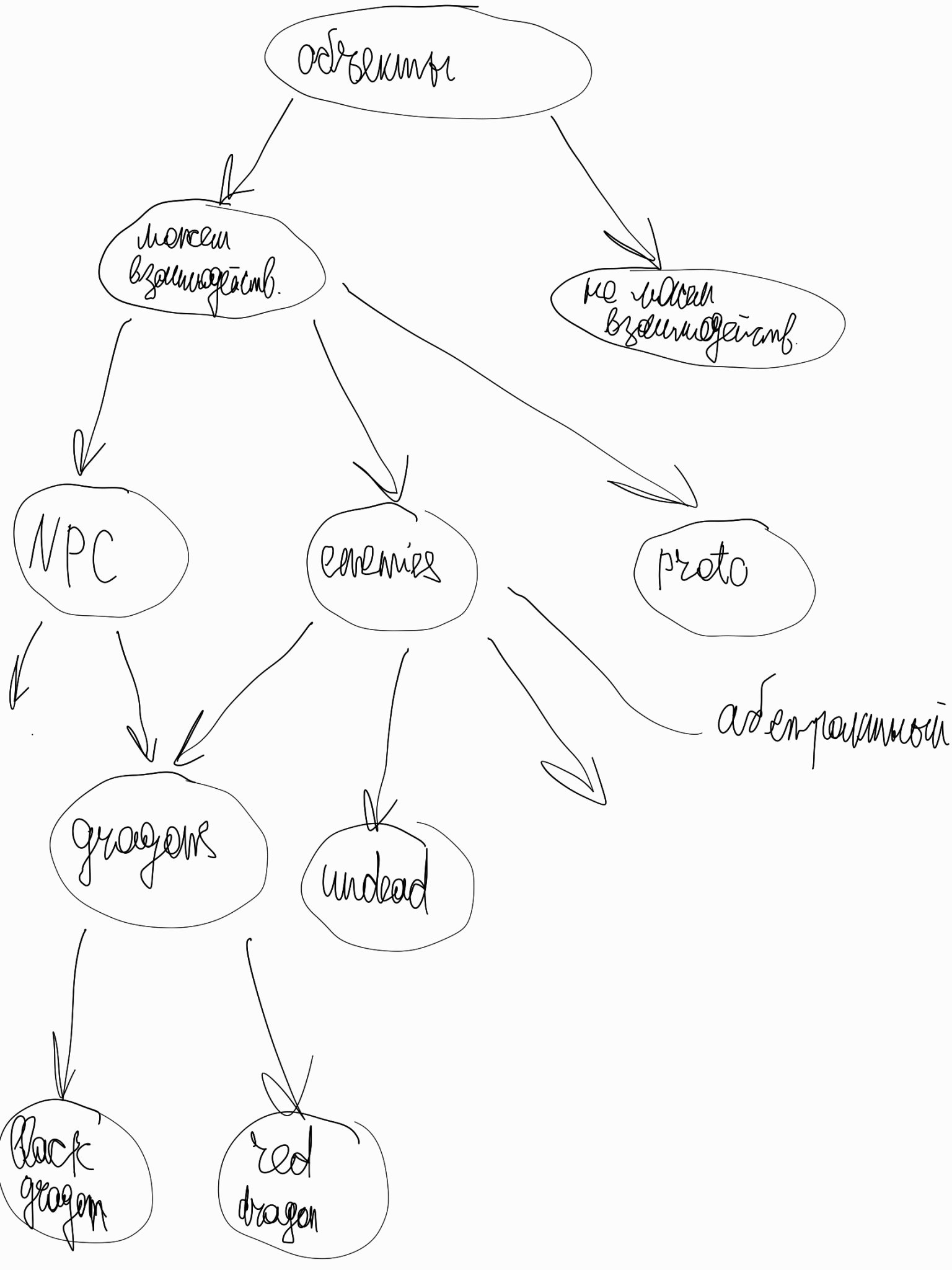
Наследование, обработка исключений

Наследование задает отношение родитель-потомок

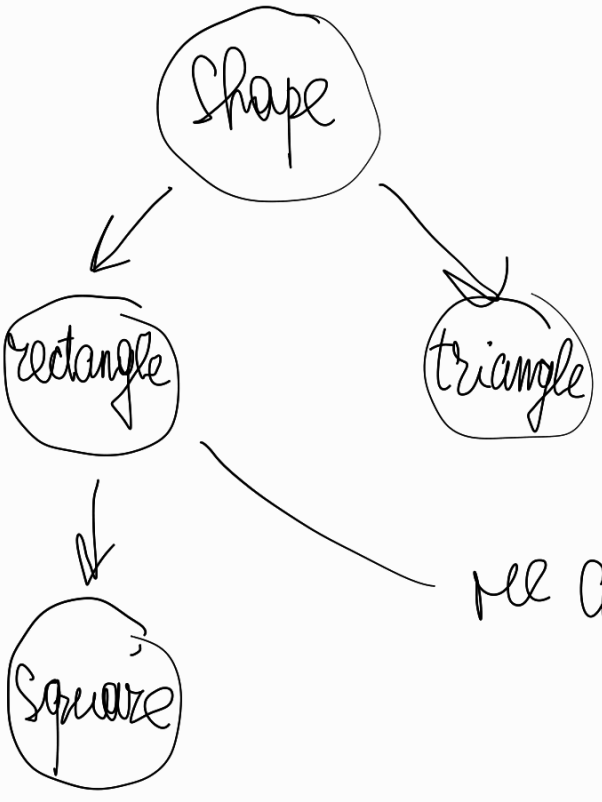
потомок полностью наследует и интерфейс и реализацию, но он может их дополнять и перераспределять.

Позволяет структурировать логику используемых сущностей, уменьшает дублирование кода.

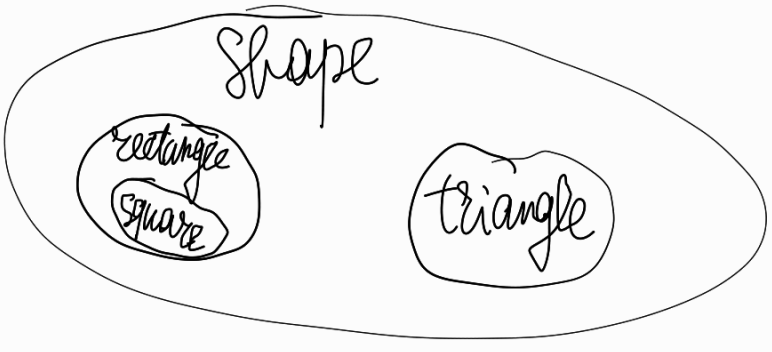
У одного родителя может быть несколько потомков, у одного потомка несколько родителей, но порядок родителей имеет значение.


объекты
может взаимодейств.
не может взаимодейств.
NPC
enemies
proto
абстрактный
dragons
undead
black dragon
red dragon

Класс, создаваемый только для логики называется абстрактным

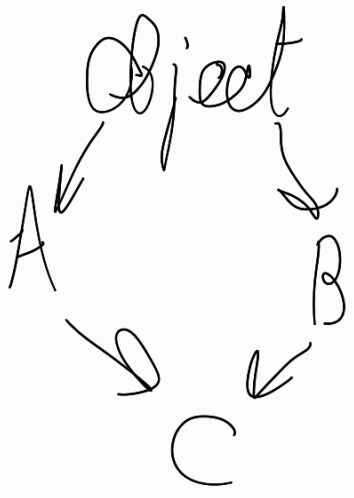


С точки зрения программирования, потомок расширяет предка, но с точки зрения логики потомок должен уточнять предка, являться его представителем.



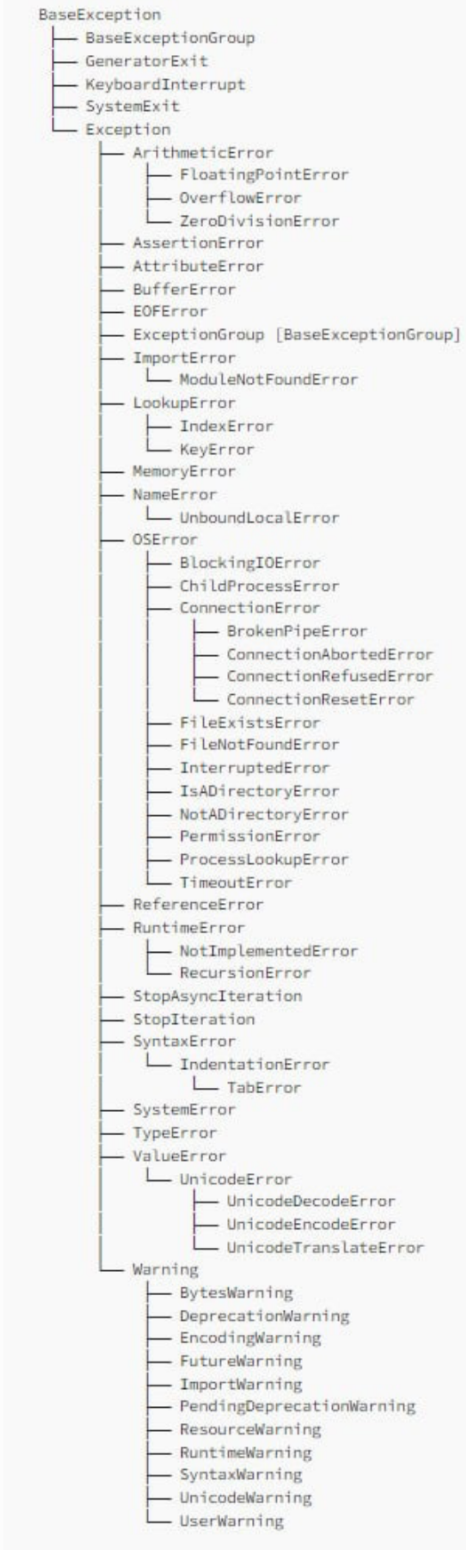
```
class Base:
    pass
class Derived(Base):
    .
    .
    .
```

```
class A:
    .
    .
class B:
    .
    .
class C(A,B):
    .
    .
```



super() - позволяет обращаться к методам предков.

Ре любое исключение повод завершить работу программы.



raise Exception

try:

: ← код который может выбросить исключение

except ZeroDivisionError: ← будем ловить ее и ее потомков

except:

else: ← не было исключения

:

finally: ← в любом случае

:

```
class MyException(Exception):
    pass

raise MyException("Exception text")

except MeException as Err:
    print(Err)
```