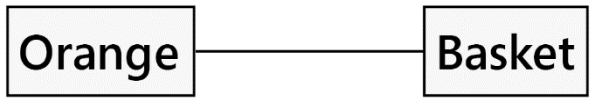
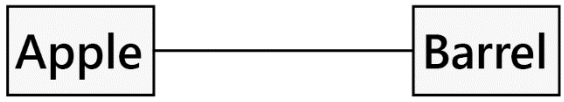
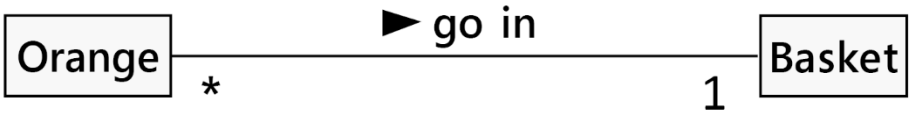
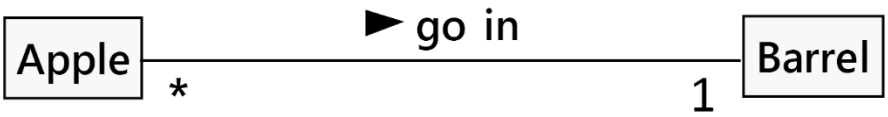
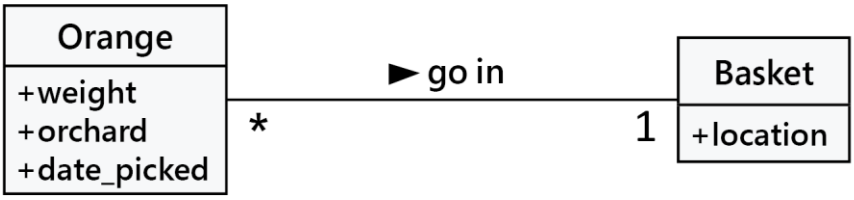
Chapter 1: Object-Oriented Design

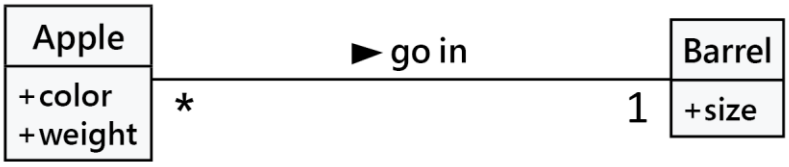


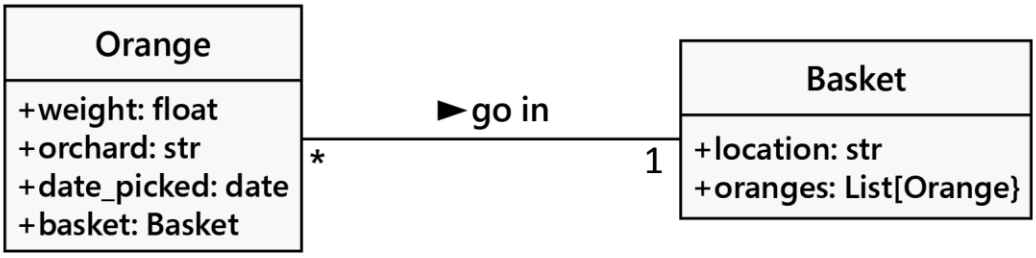




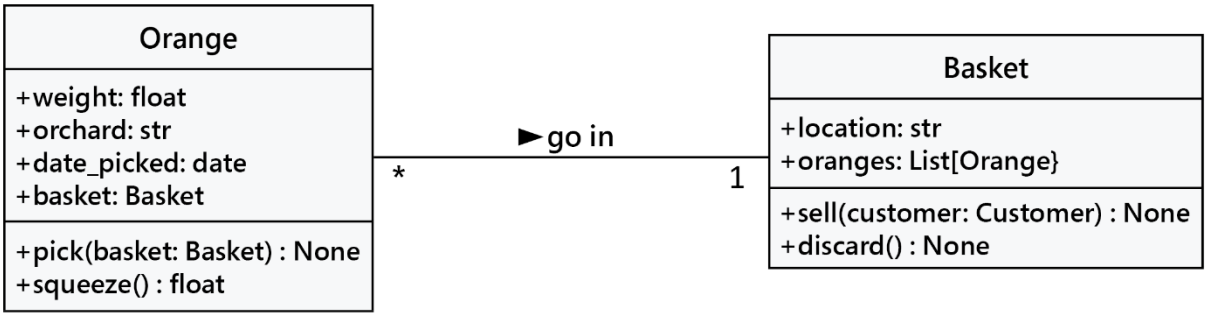


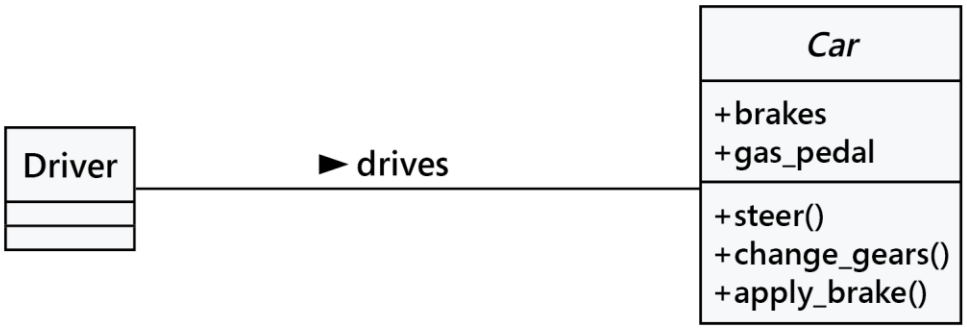

Orange
+weight
+orchard
+date_picked
► go in
*
1
Basket
+location

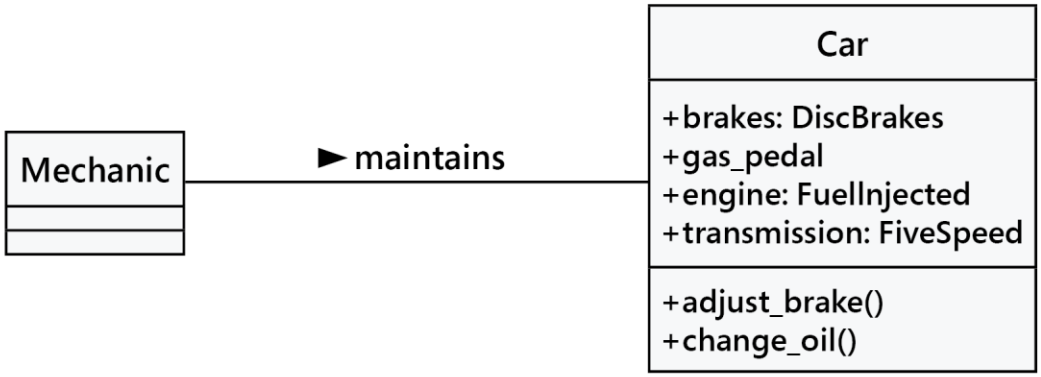

Apple
+color
+weight
► go in
*
1
Barrel
+size

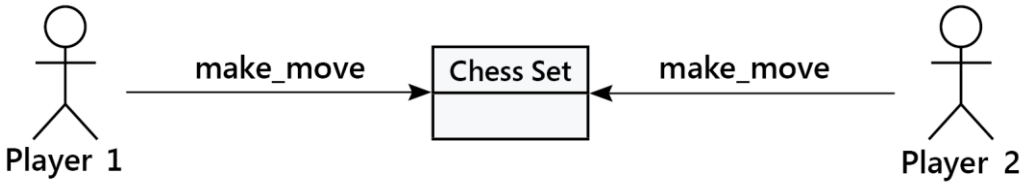

Orange
+weight: float
+orchard: str
+date_picked: date
+basket: Basket
► go in
*
1
Basket
+location: str
+oranges: List[Orange}

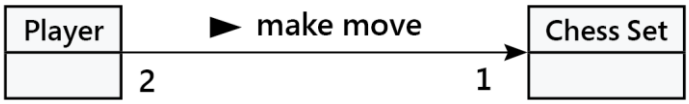

Apple
+color: str
+weight: float
+barrel: Barrel
► go in
*
1
Barrel
+size: int
+apples: List[Apple]

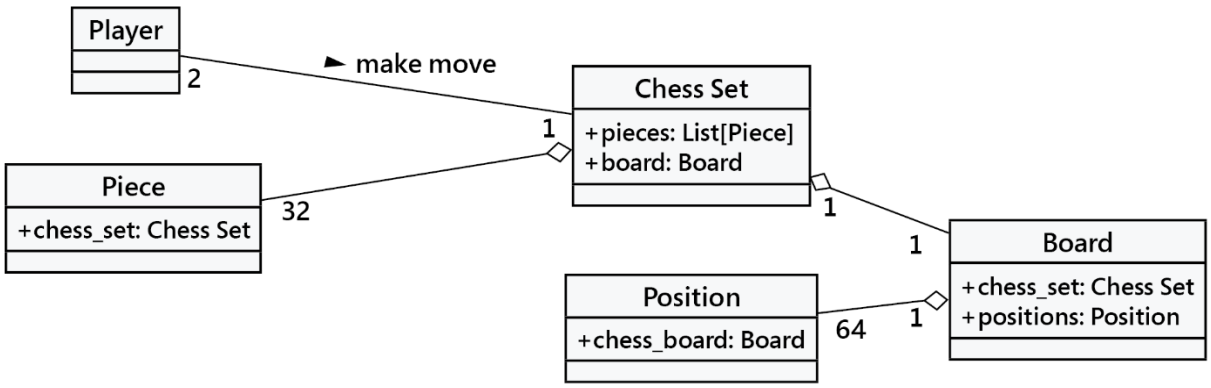

Orange
+weight: float
+orchard: str
+date_picked: date
+basket: Basket
+pick(basket: Basket) : None
+squeeze() : float
► go in
*
1
Basket
+location: str
+oranges: List[Orange}
+sell(customer: Customer) : None
+discard() : None

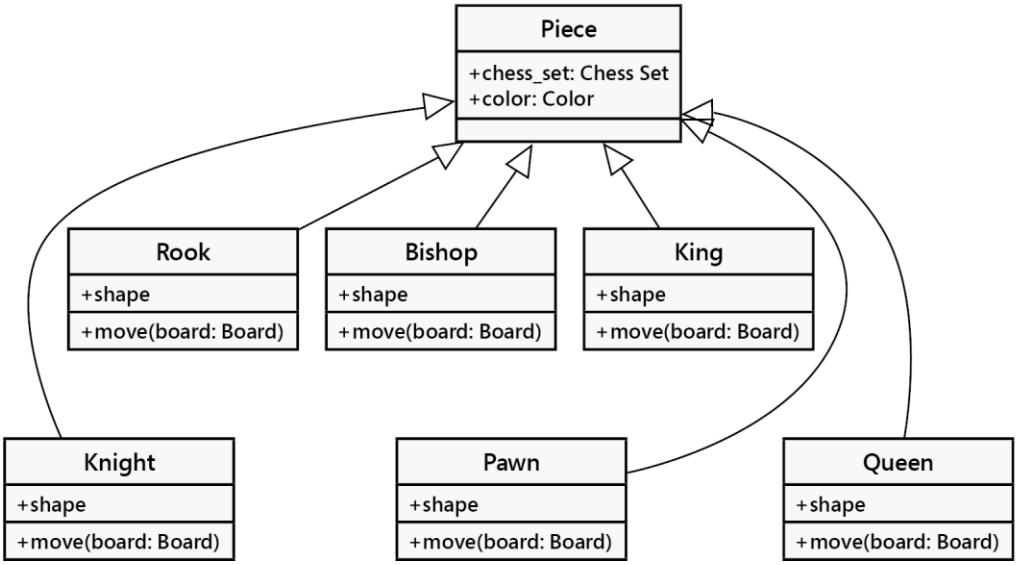

Car
+brakes
+gas_pedal
+steer()
+change_gears()
+apply_brake()
Driver
drives

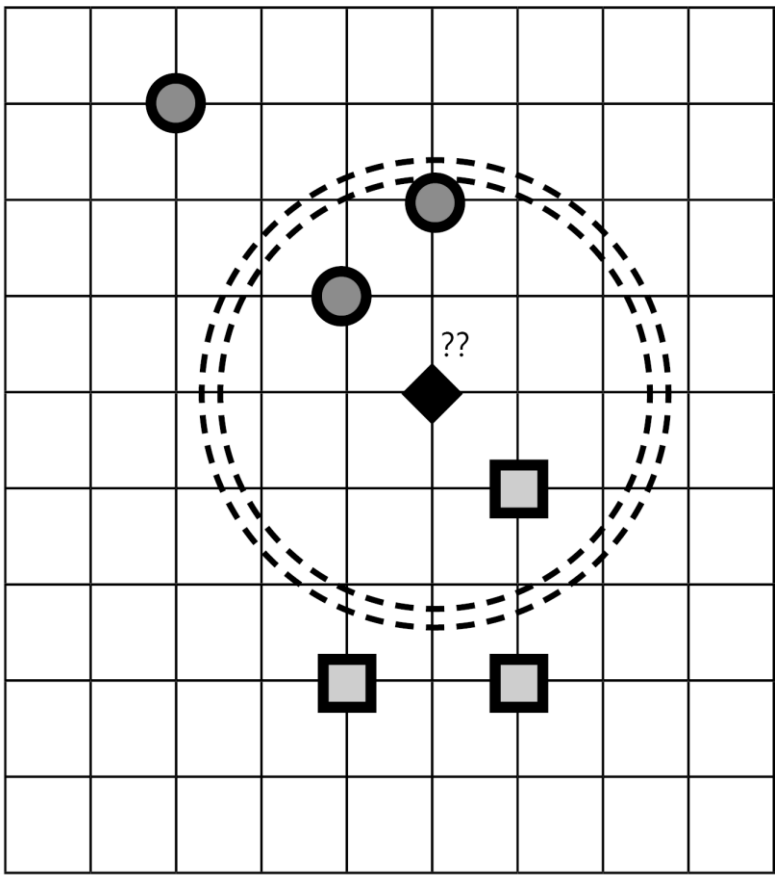

Car
+brakes: DiscBrakes
+gas_pedal
+engine: FuelInjected
+transmission: FiveSpeed
+adjust_brake()
+change_oil()
Mechanic
maintains

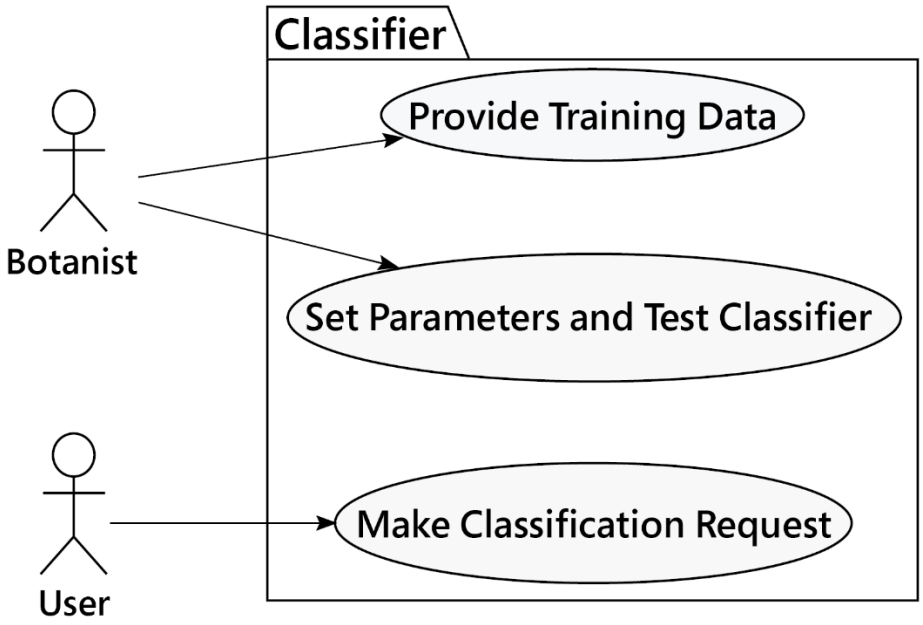

Player 1
make_move
Chess Set
make_move
Player 2

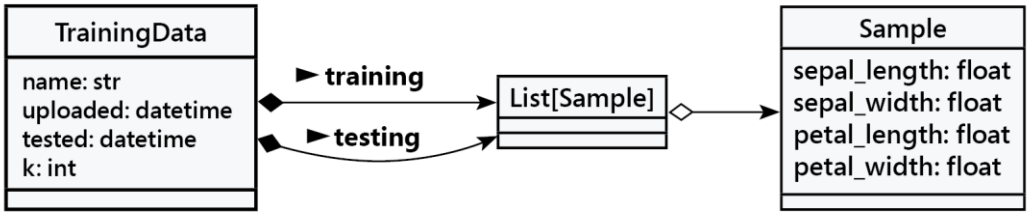

Player
make move
Chess Set
2
1

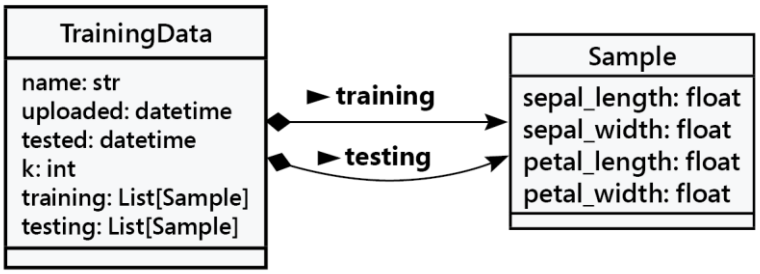

Player
2
make move
Chess Set
+pieces: List[Piece]
+board: Board
1
Piece
+chess_set: Chess Set
32
1
1
Board
+chess_set: Chess Set
+positions: Position
Position
+chess_board: Board
64
1

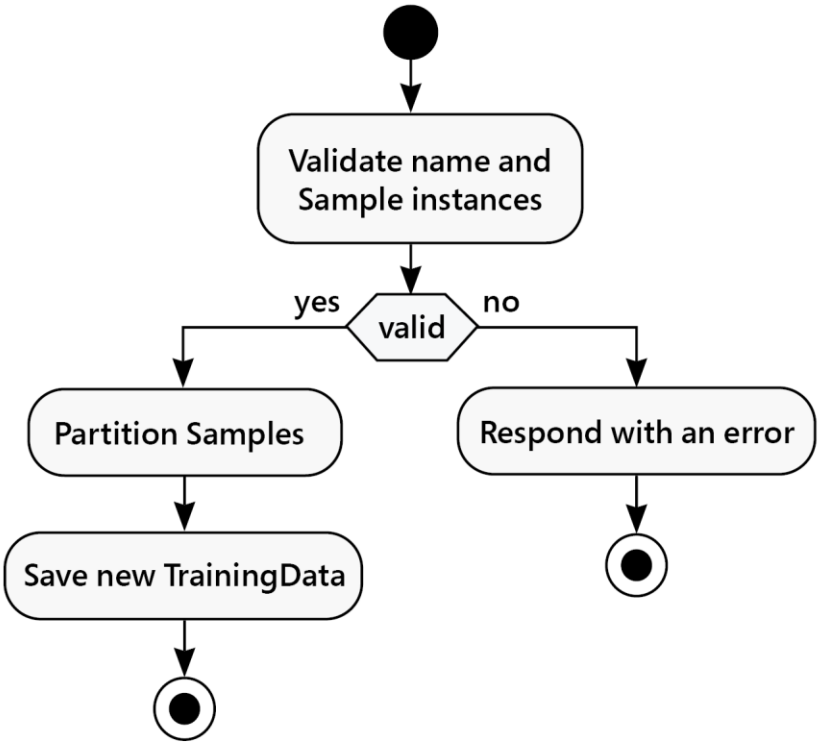

Piece
+chess_set: Chess Set
+color: Color
Rook
+shape
+move(board: Board)
Bishop
+shape
+move(board: Board)
King
+shape
+move(board: Board)
Knight
+shape
+move(board: Board)
Pawn
+shape
+move(board: Board)
Queen
+shape
+move(board: Board)

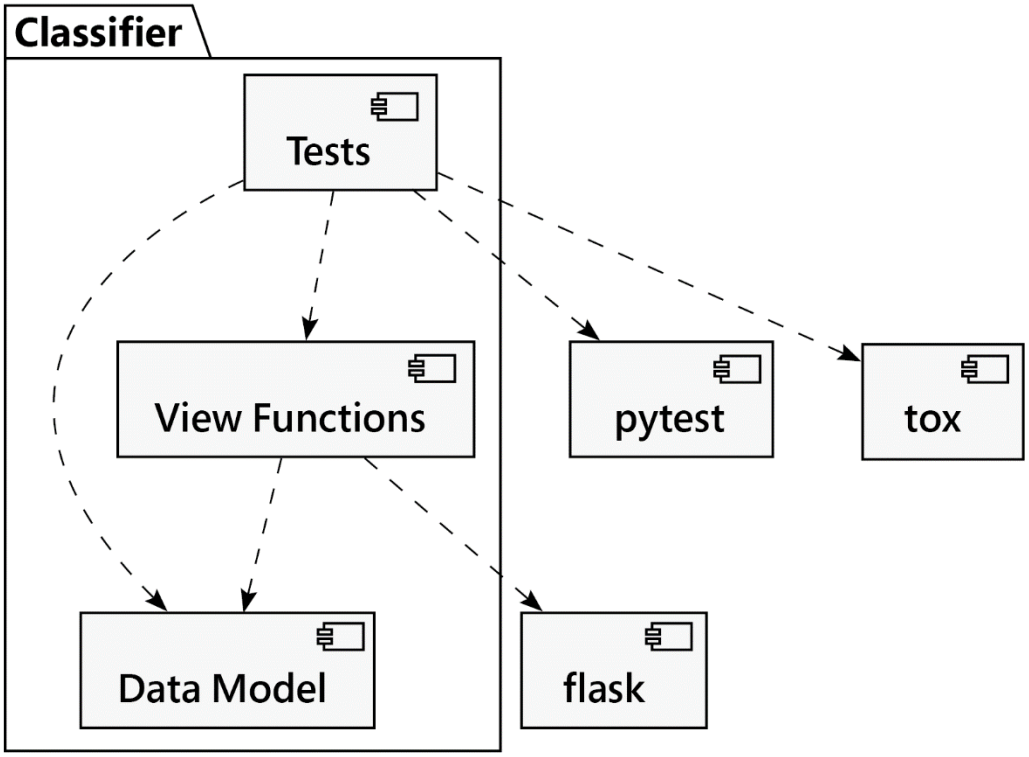

??

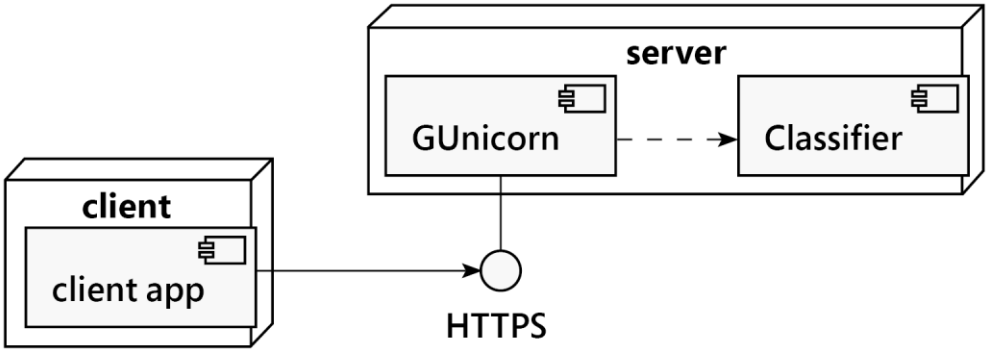

Classifier
Provide Training Data
Set Parameters and Test Classifier
Make Classification Request
Botanist
User


TrainingData
name: str
uploaded: datetime
tested: datetime
k: int
training
testing
List[Sample]
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float

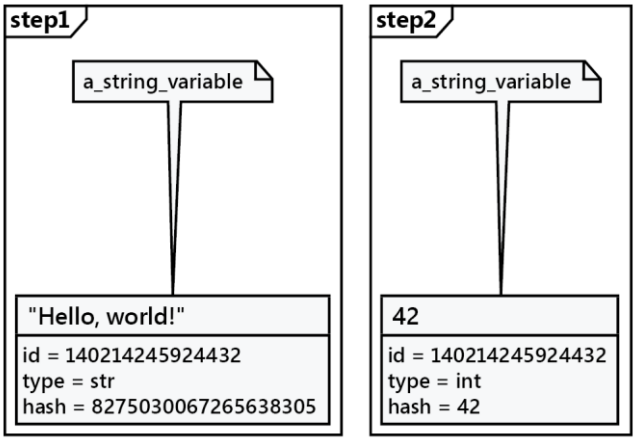

TrainingData
name: str
uploaded: datetime
tested: datetime
k: int
training: List[Sample]
testing: List[Sample]
training
testing
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float

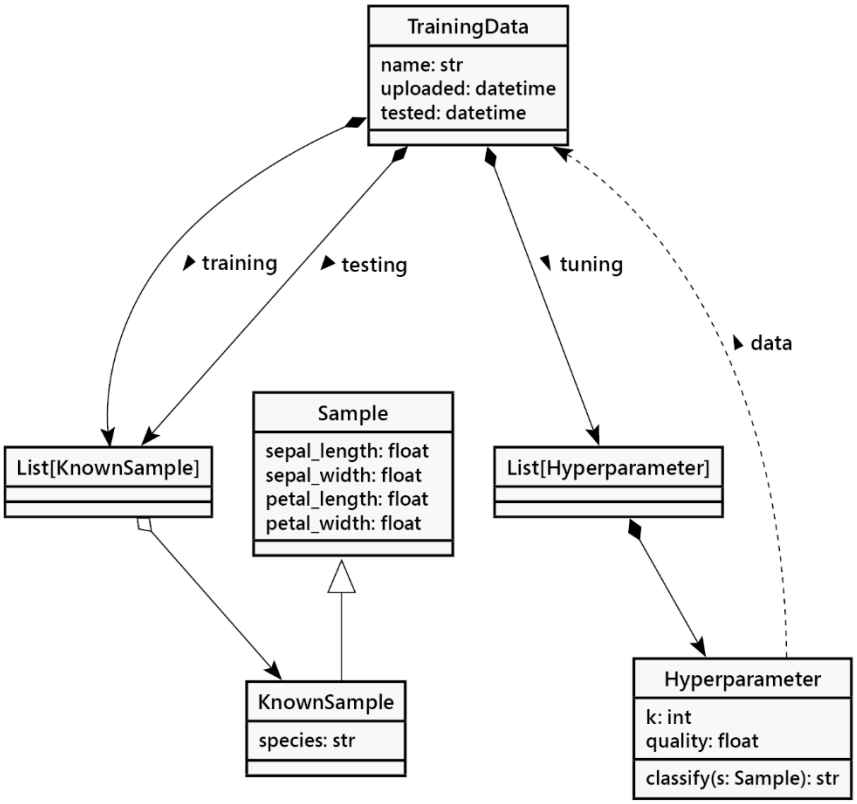

Validate name and Sample instances
valid
yes
no
Partition Samples
Respond with an error
Save new TrainingData

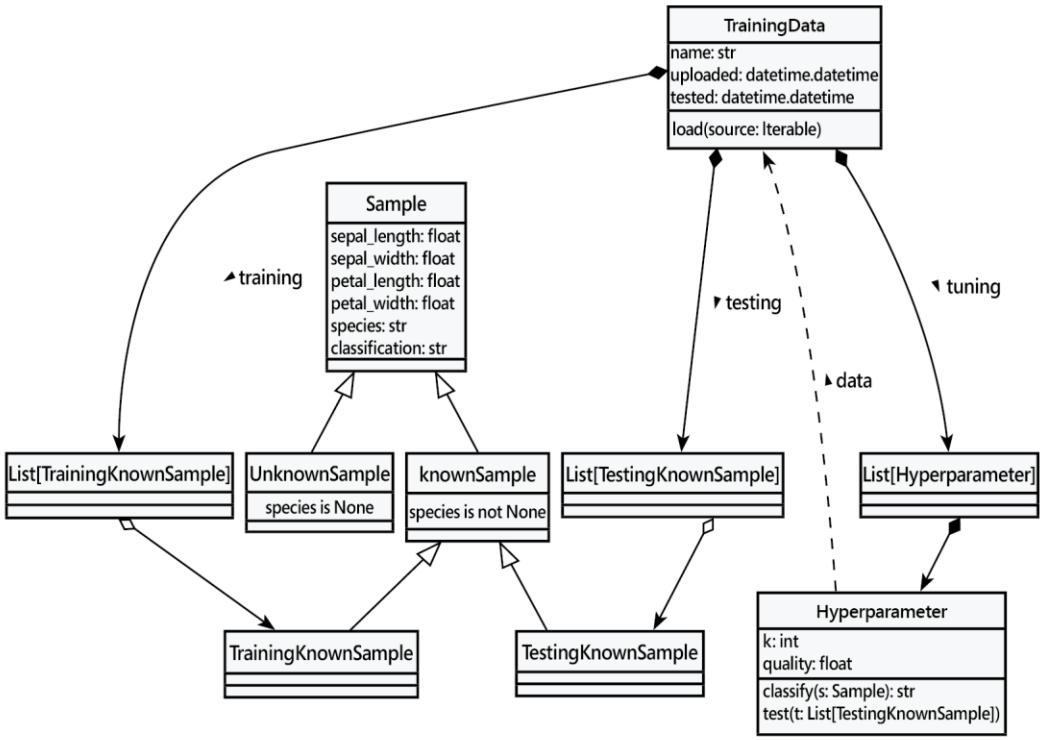

Classifier
Tests
View Functions
Data Model
flask
pytest
tox


server
GUnicorn
Classifier
client
client app
HTTPS

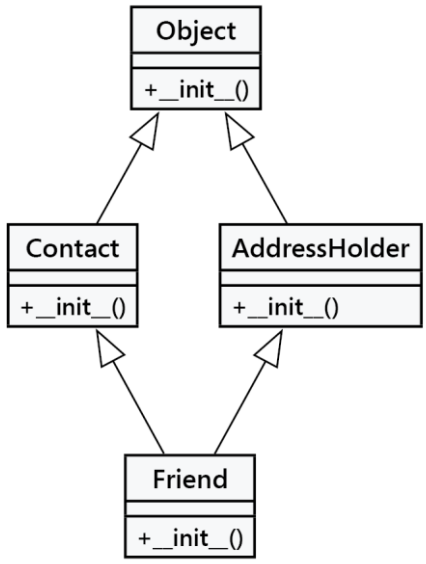
Chapter 2: Objects in Python

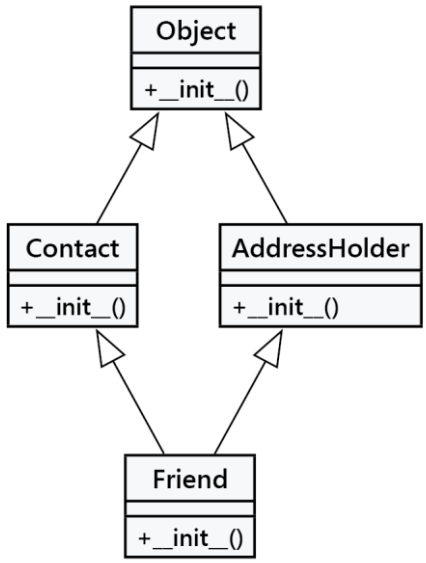


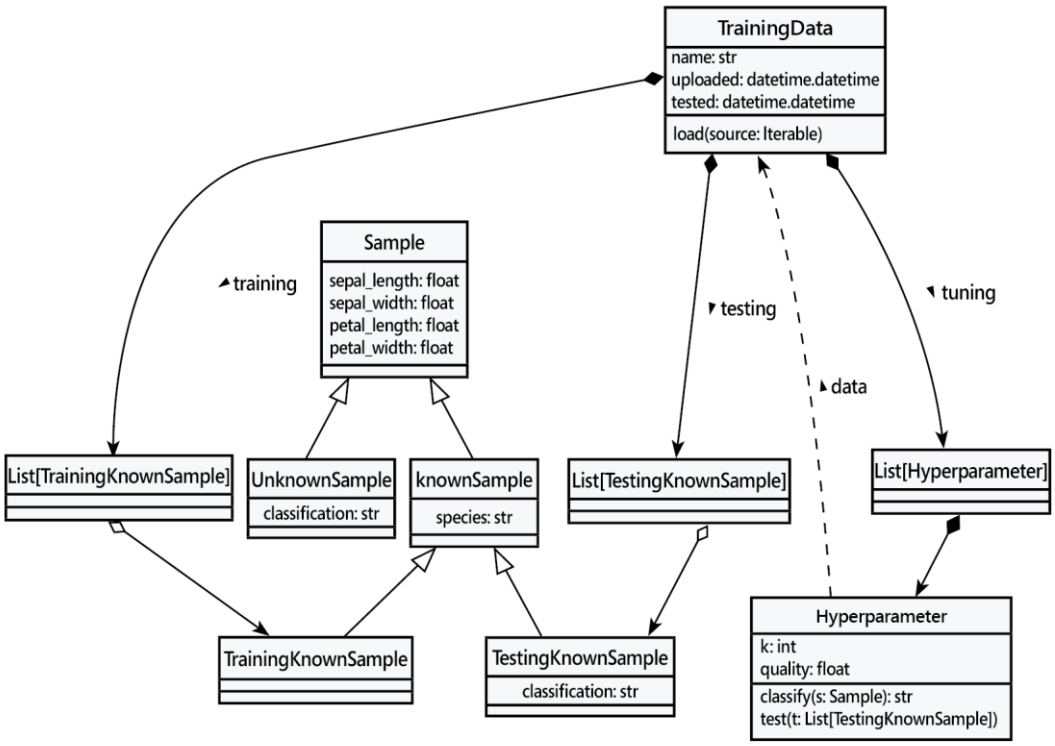


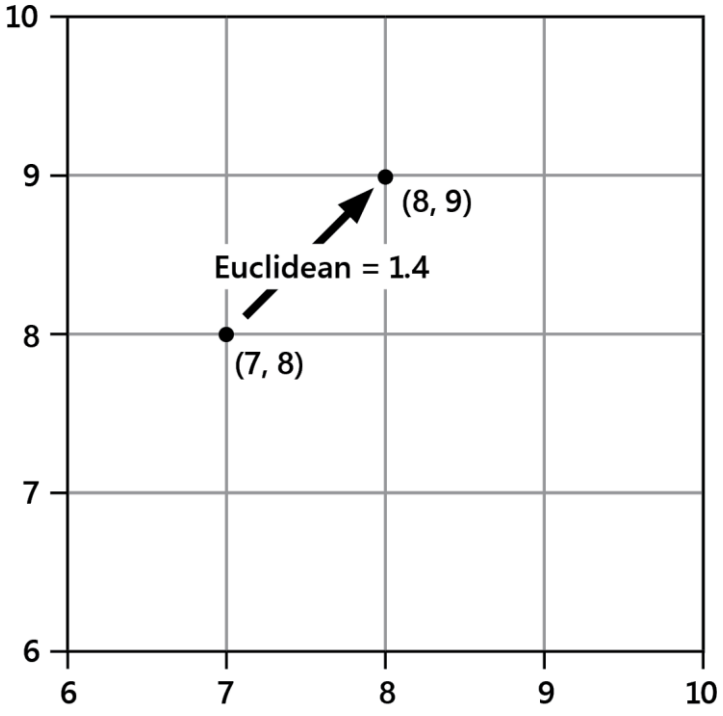

TrainingData
name: str
uploaded: datetime.datetime
tested: datetime.datetime
load(source: Iterable)
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
species: str
classification: str
training
testing
data
tuning
List[TrainingKnownSample]
UnknownSample
species is None
knownSample
species is not None
List[TestingKnownSample]
List[Hyperparameter]
TrainingKnownSample
TestingKnownSample
Hyperparameter
k: int
quality: float
classify(s: Sample): str
test(t: List[TestingKnownSample])

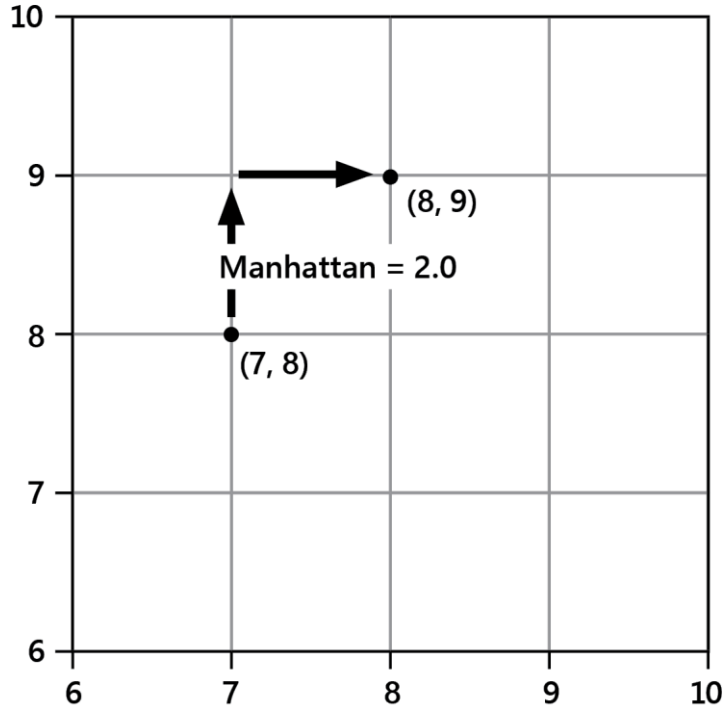
Chapter 3: When Objects Are Alike

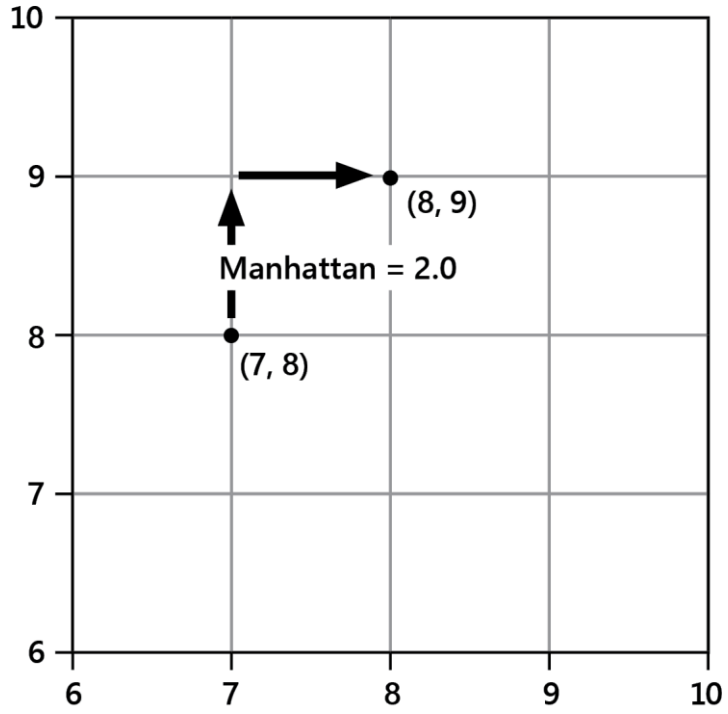


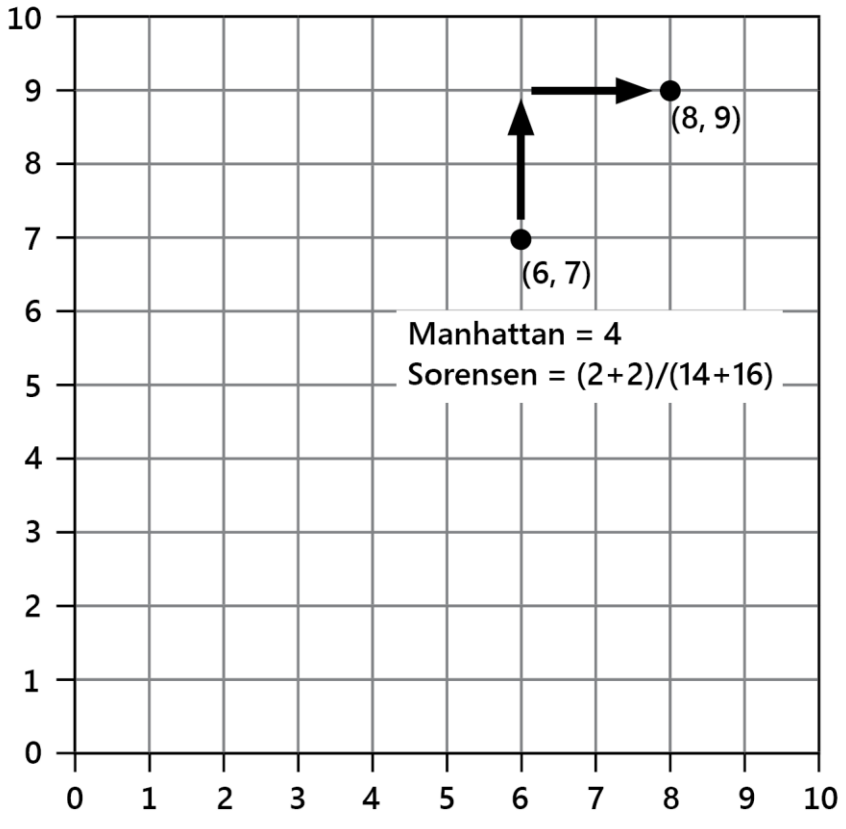



TrainingData
name: str
uploaded: datetime.datetime
tested: datetime.datetime
load(source: Iterable)
training
testing
data
tuning
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
List[TrainingKnownSample]
UnknownSample
classification: str
knownSample
species: str
List[TestingKnownSample]
List[Hyperparameter]
TrainingKnownSample
TestingKnownSample
classification: str
Hyperparameter
k: int
quality: float
classify(s: Sample): str
test(t: List[TestingKnownSample])

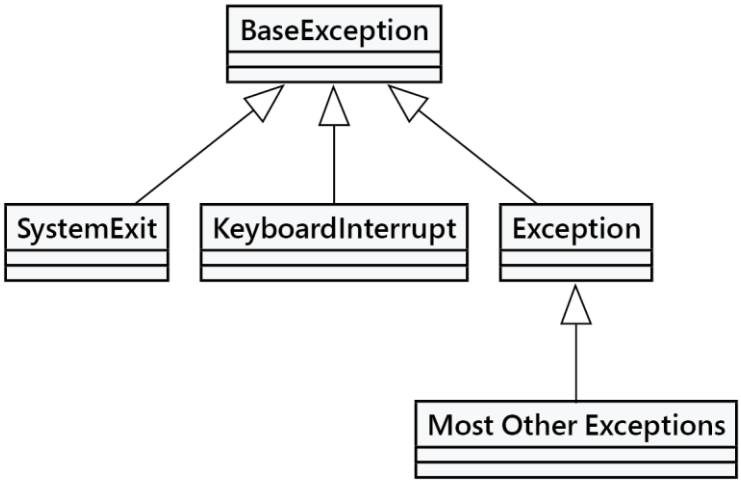

10
9
8
7
6
6
7
8
9
10
(8, 9)
Euclidean = 1.4
(7, 8)

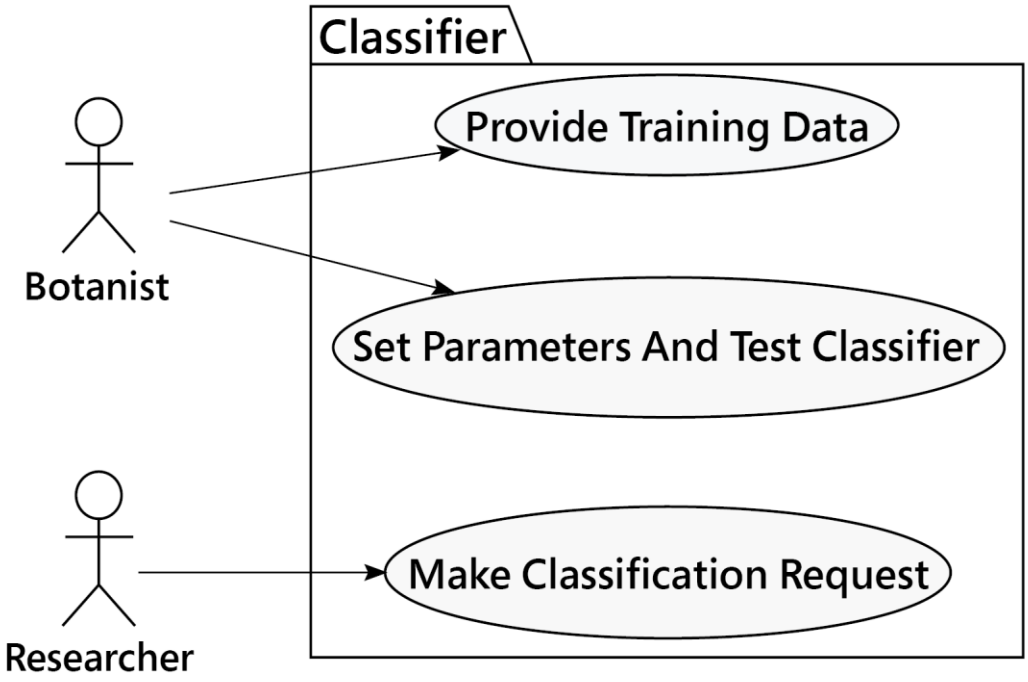

10
9
8
7
6
6
7
8
9
10
(8, 9)
Manhattan = 2.0
(7, 8)

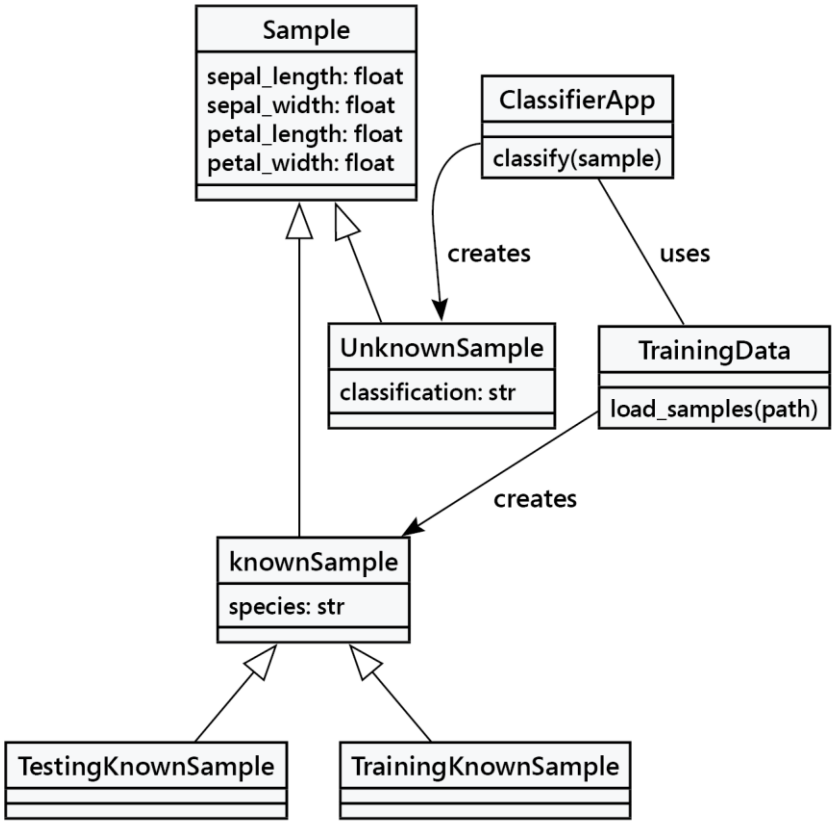

10
9
8
7
6
6
7
8
9
10
(8, 9)
Manhattan = 2.0
(7, 8)


10
9
8
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
8
9
10
(8, 9)
(6, 7)
Manhattan = 4
Sorensen = (2+2)/(14+16)

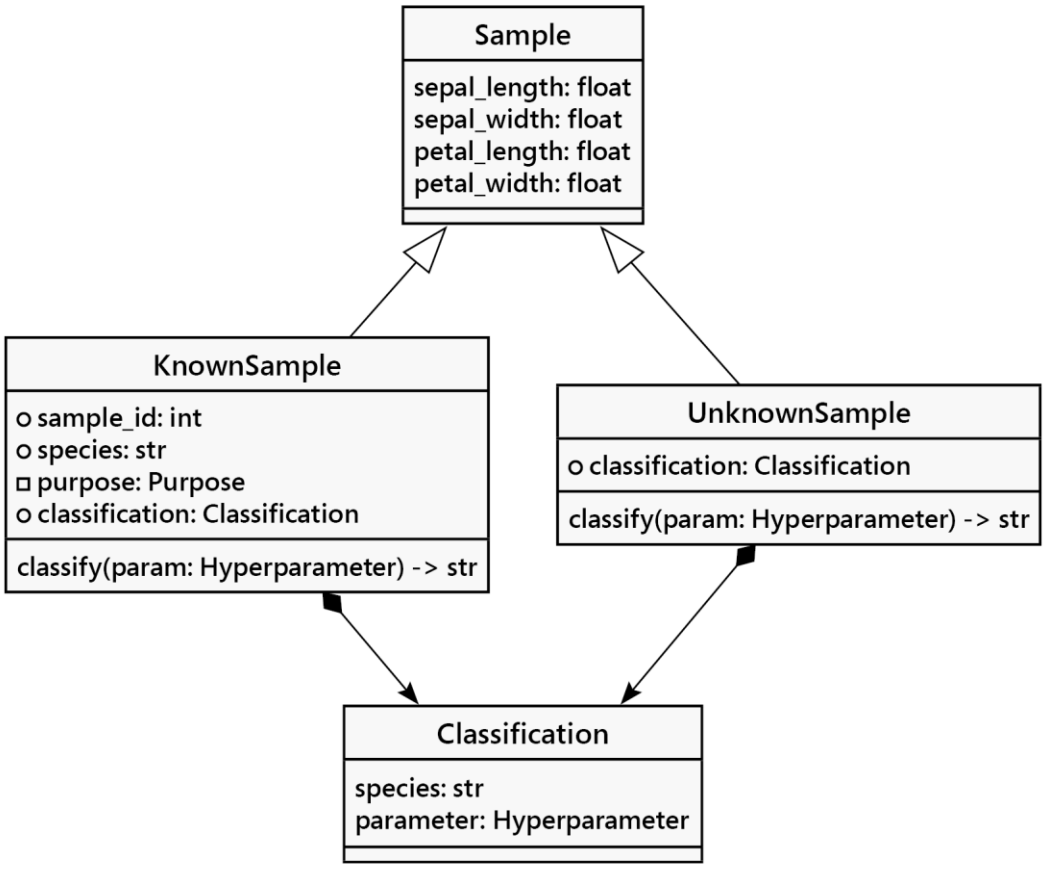
Chapter 4: Expecting the Unexpected



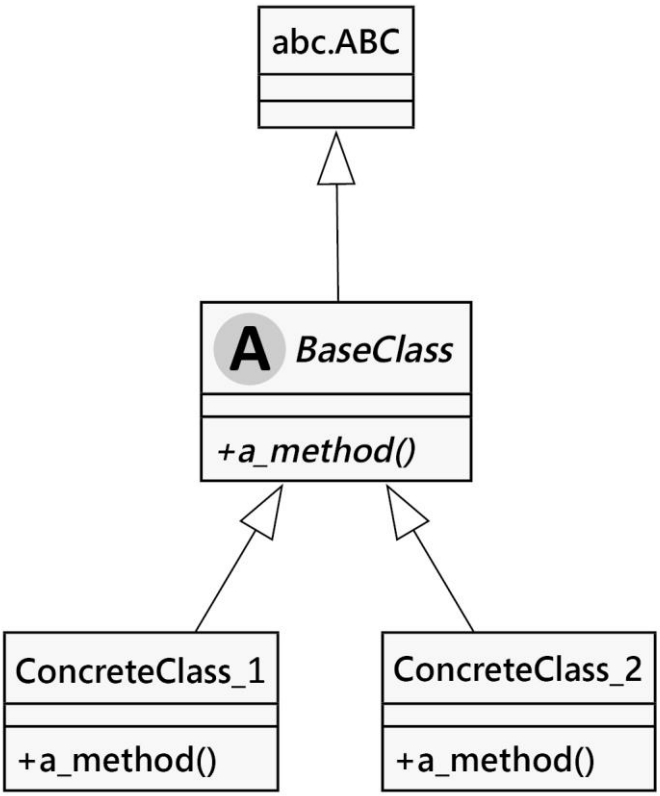


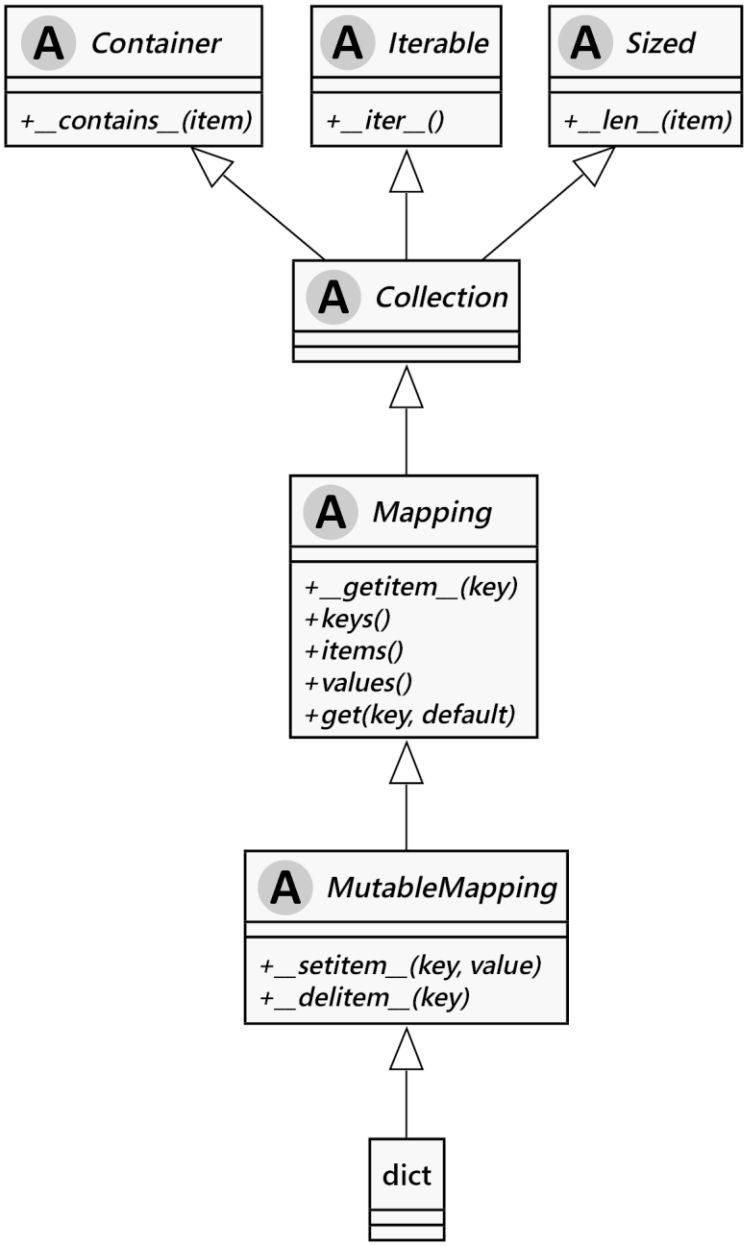

Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
ClassifierApp
classify(sample)
creates
uses
UnknownSample
classification: str
TrainingData
load_samples(path)
creates
knownSample
species: str
TestingKnownSample
TrainingKnownSample

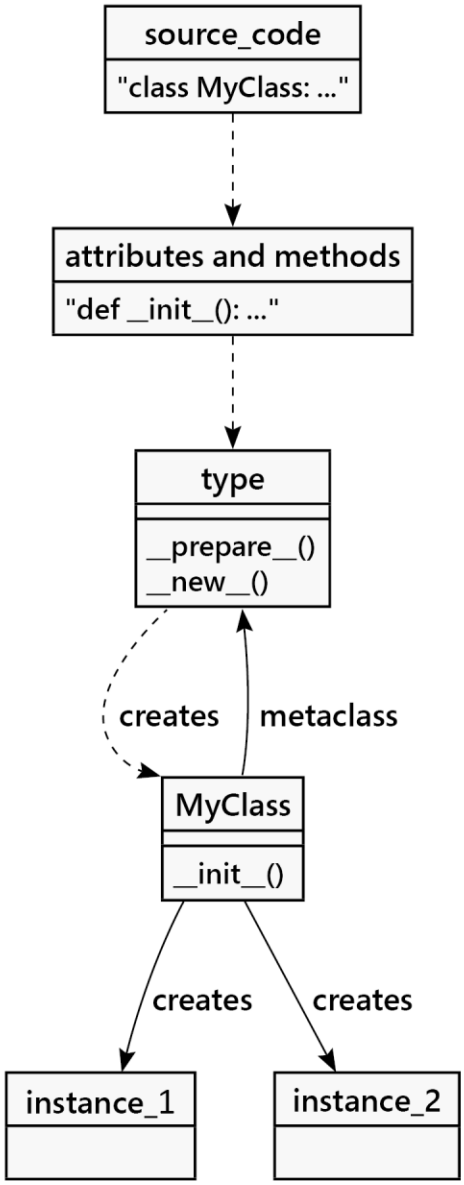
Chapter 5: When to Use Object-Oriented Programming

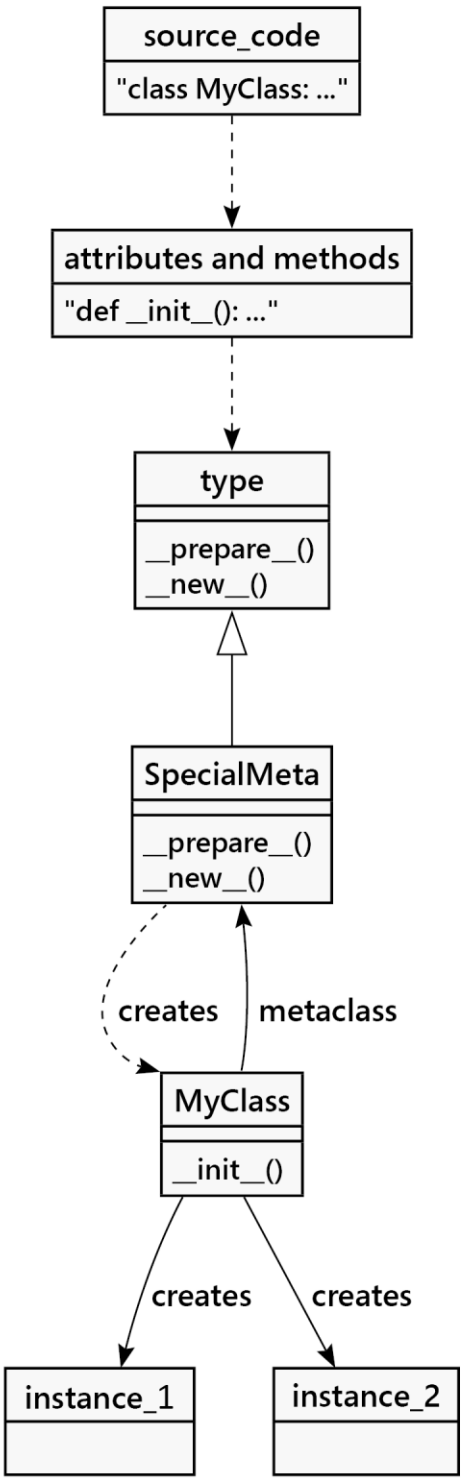


Chapter 6: Abstract Base Classes and Operator Overloading

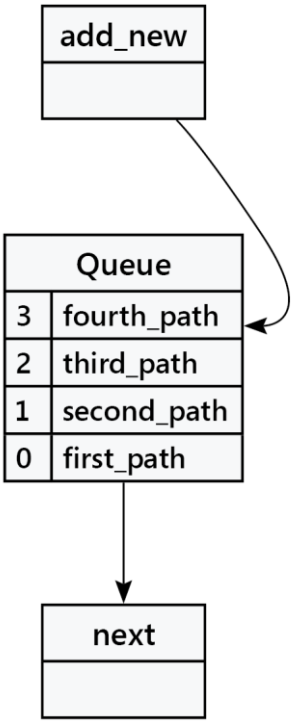


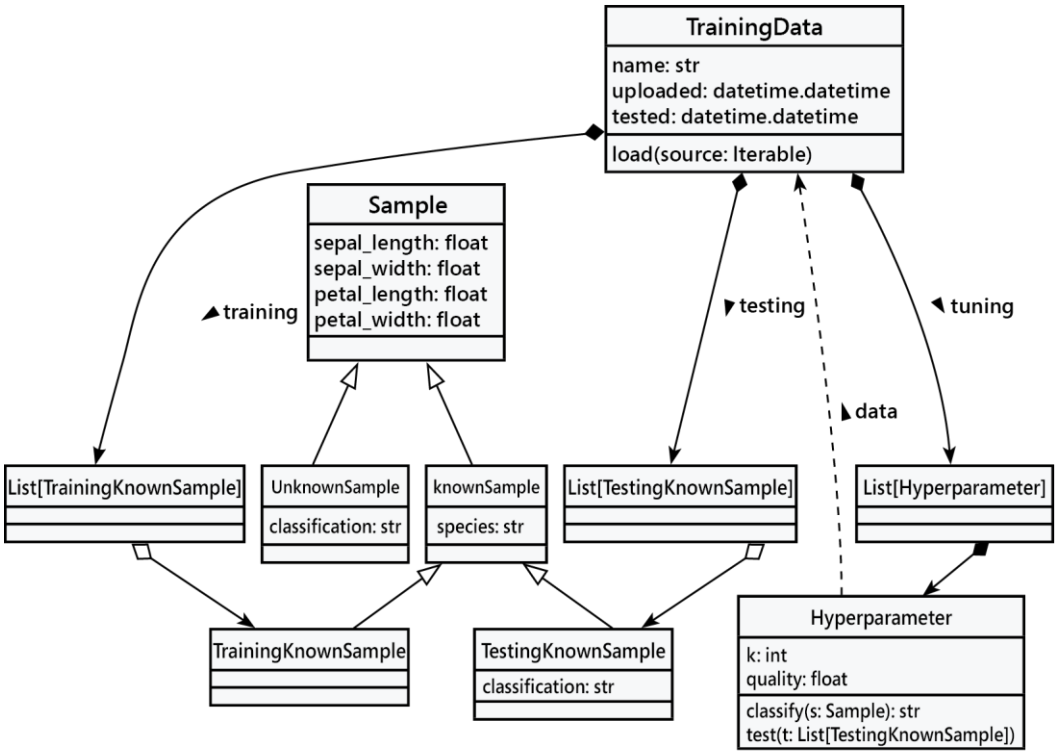

Container
+__contains__(item)
Iterable
+__iter__()
Sized
+__len__(item)
Collection
Mapping
+__getitem__(key)
+keys()
+items()
+values()
+get(key, default)
MutableMapping
+__setitem__(key, value)
+__delitem__(key)
dict

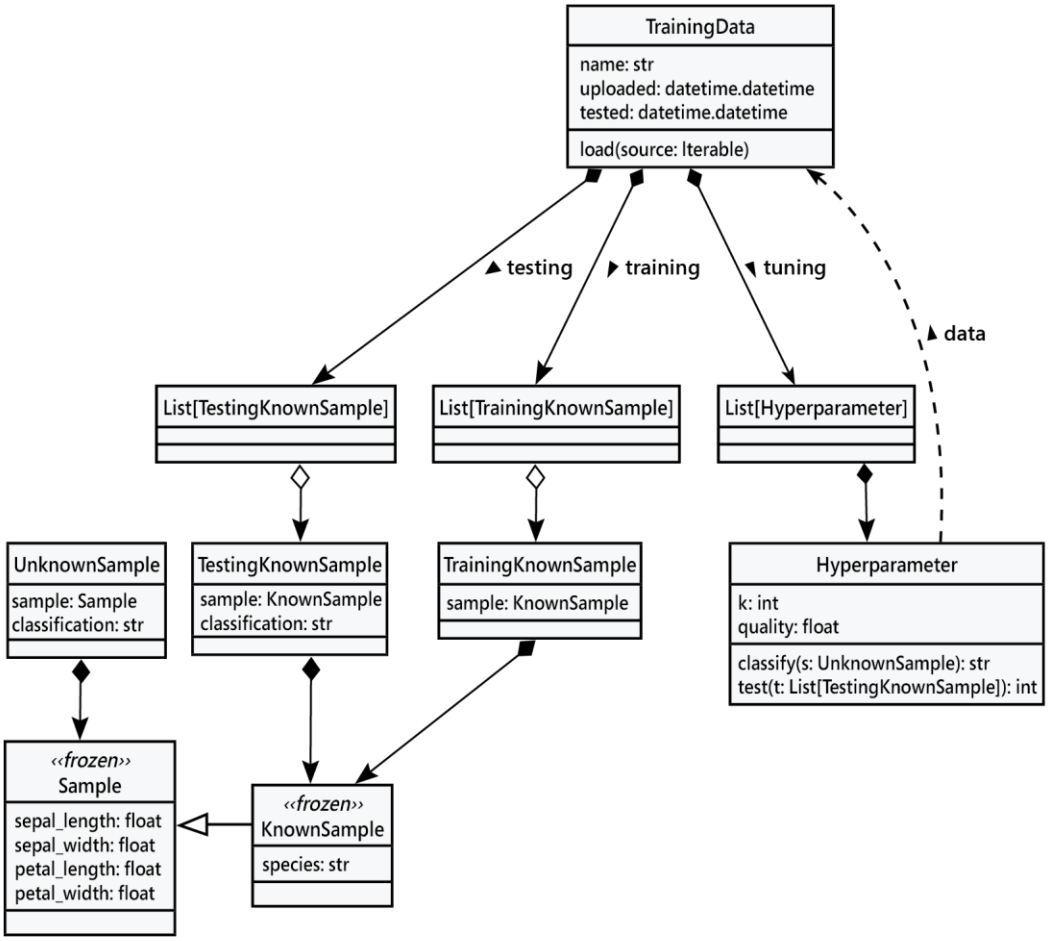

source_code
"class MyClass: ..."
attributes and methods
"def __init__(): ..."
type
__prepare__()
__new__()
creates
metaclass
MyClass
__init__()
creates
creates
instance_1
instance_2

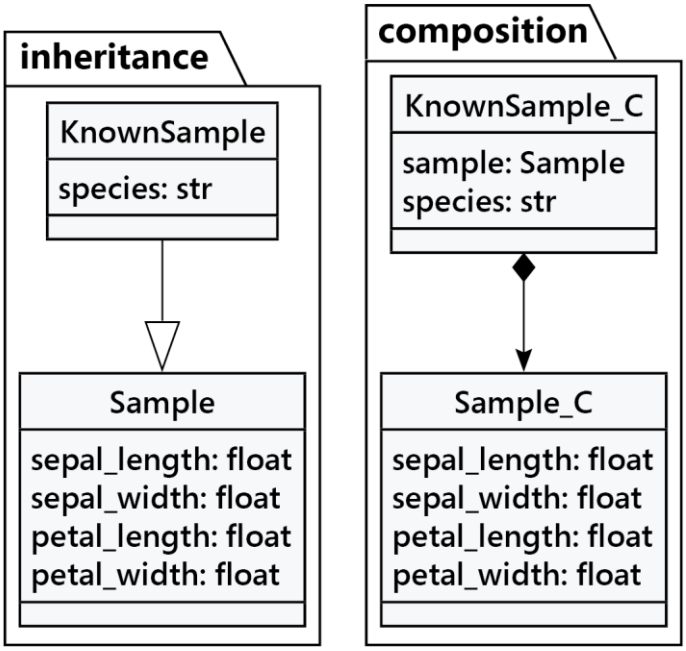

source_code
"class MyClass: ..."
attributes and methods
"def __init__(): ..."
type
__prepare__()
__new__()
SpecialMeta
__prepare__()
__new__()
creates
metaclass
MyClass
__init__()
creates
creates
instance_1
instance_2

Chapter 7: Python Data Structures

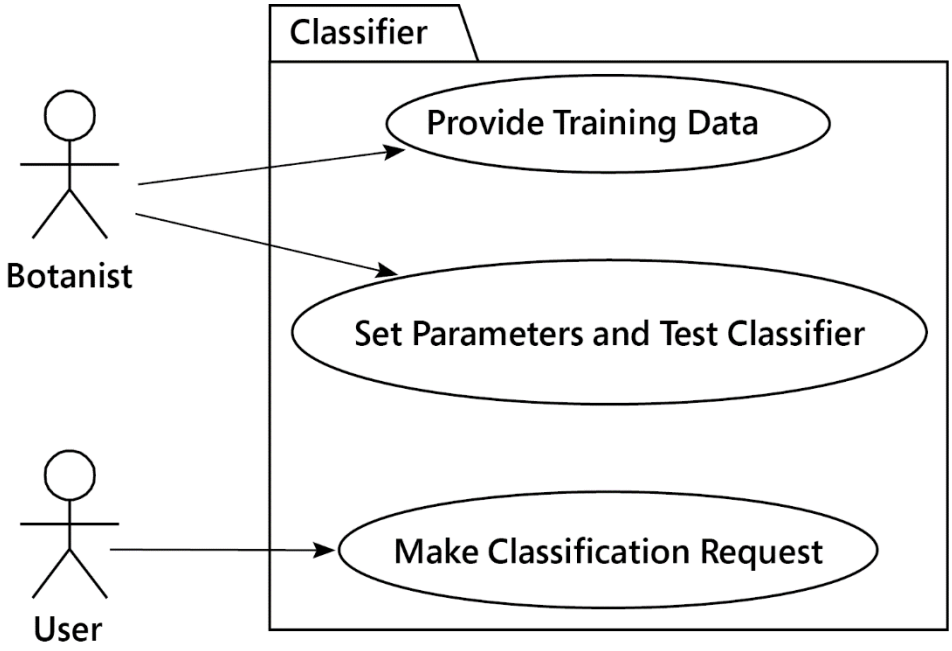


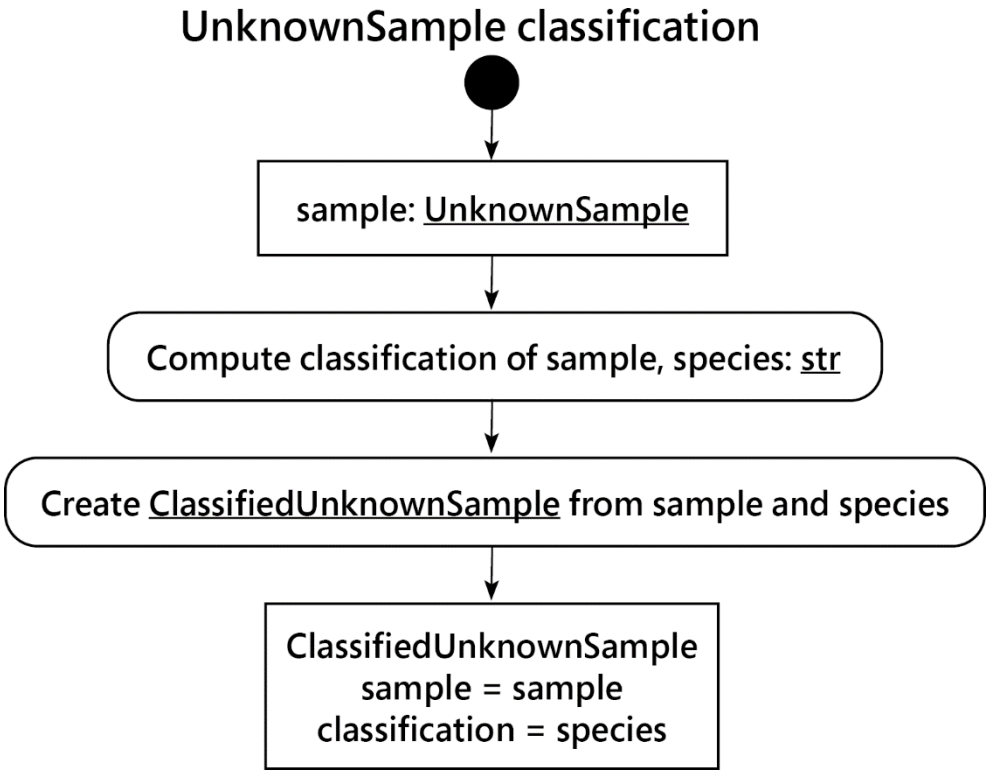

TrainingData
name: str
uploaded: datetime.datetime
tested: datetime.datetime
load(source: Iterable)
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
training
testing
tuning
data
List[TrainingKnownSample]
UnknownSample
classification: str
knownSample
species: str
List[TestingKnownSample]
List[Hyperparameter]
TrainingKnownSample
TestingKnownSample
classification: str
Hyperparameter
k: int
quality: float
classify(s: Sample): str
test(t: List[TestingKnownSample])

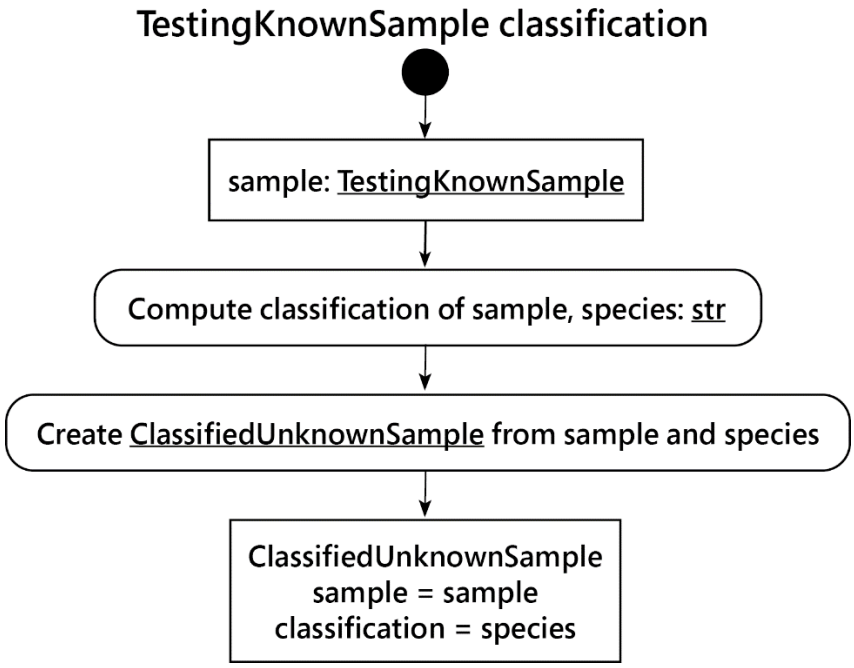

TrainingData
name: str
uploaded: datetime.datetime
tested: datetime.datetime
load(source: Iterable)
testing
training
tuning
data
List[TestingKnownSample]
List[TrainingKnownSample]
List[Hyperparameter]
UnknownSample
sample: Sample
classification: str
TestingKnownSample
sample: KnownSample
classification: str
TrainingKnownSample
sample: KnownSample
Hyperparameter
k: int
quality: float
classify(s: UnknownSample): str
test(t: List[TestingKnownSample]): int
‹‹frozen››
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
‹‹frozen››
KnownSample
species: str

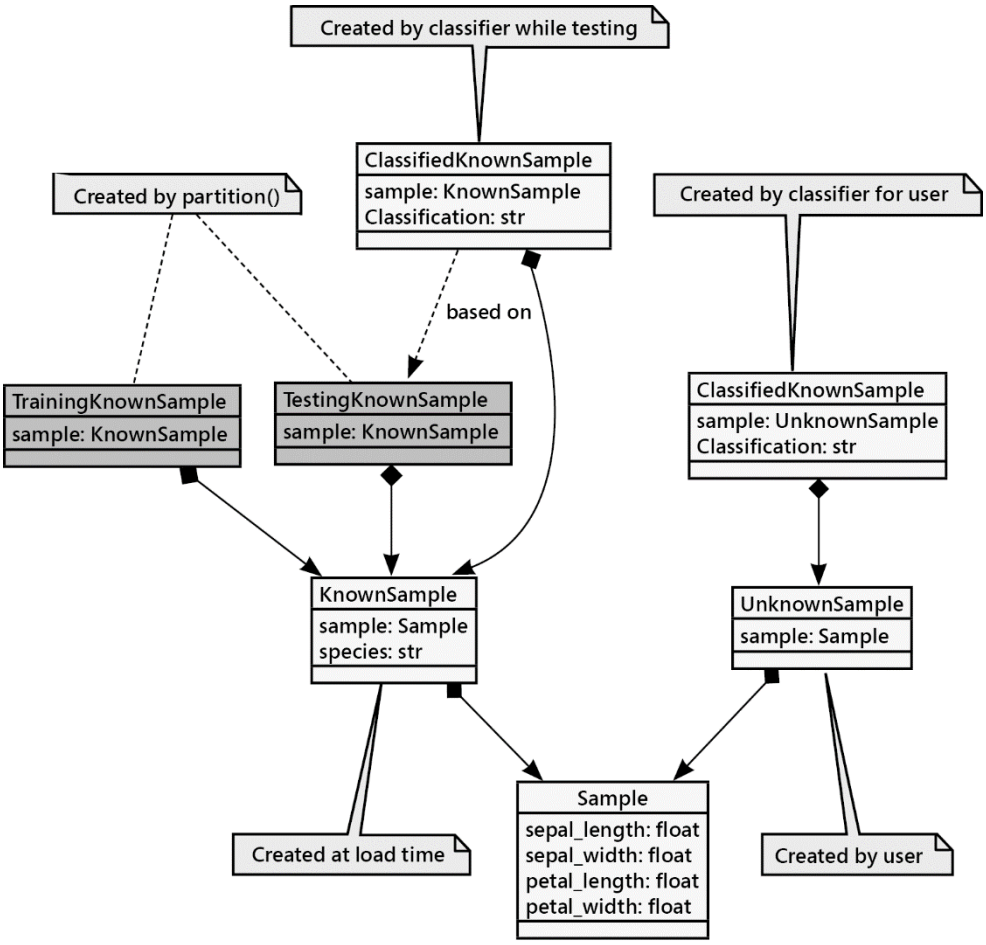

inheritance
KnownSample
species: str
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
composition
KnownSample_C
sample: Sample
species: str
Sample_C
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float

Chapter 8: The Intersection of Object-Oriented and Functional Programming

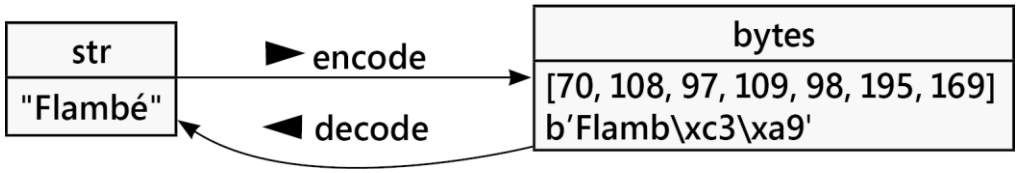




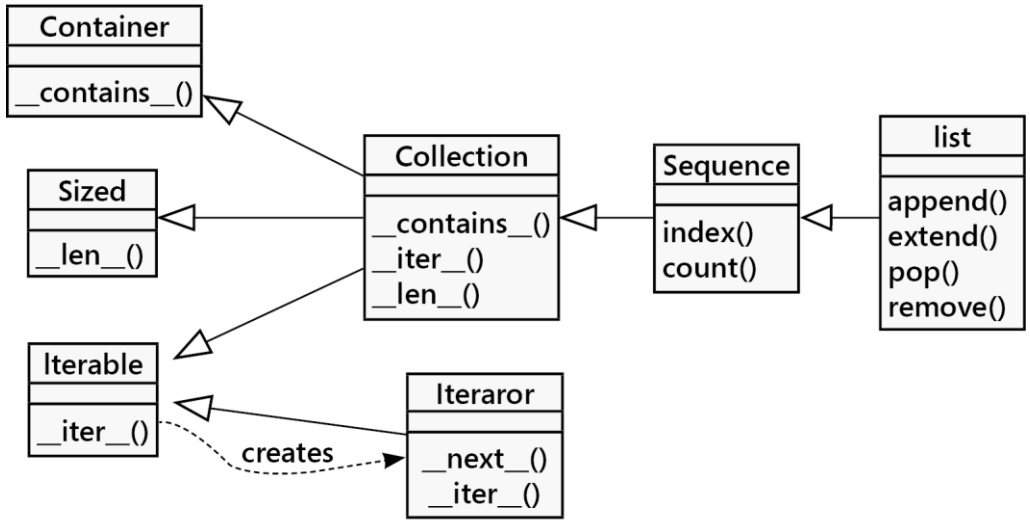

TestingKnownSample classification
sample: TestingKnownSample
Compute classification of sample, species: str
Create ClassifiedUnknownSample from sample and species
ClassifiedUnknownSample
sample = sample
classification = species

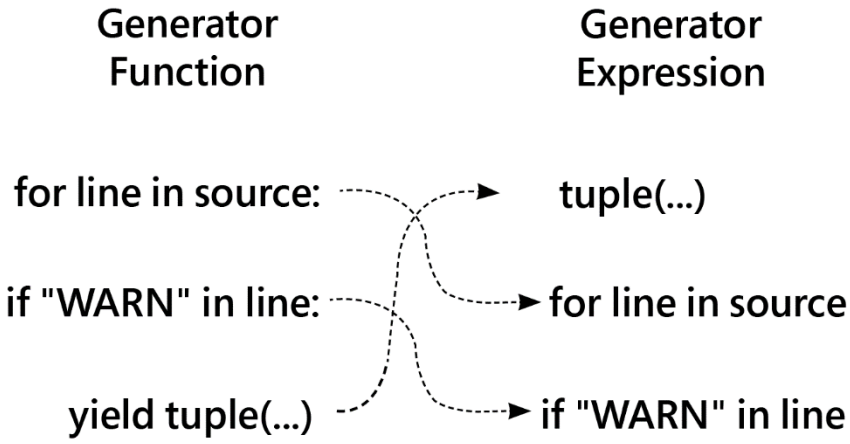

Created by classifier while testing
ClassifiedKnownSample
sample: KnownSample
Classification: str
Created by partition()
Created by classifier for user
based on
TrainingKnownSample
sample: KnownSample
TestingKnownSample
sample: KnownSample
ClassifiedKnownSample
sample: UnknownSample
Classification: str
KnownSample
sample: Sample
species: str
UnknownSample
sample: Sample
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
Created at load time
Created by user

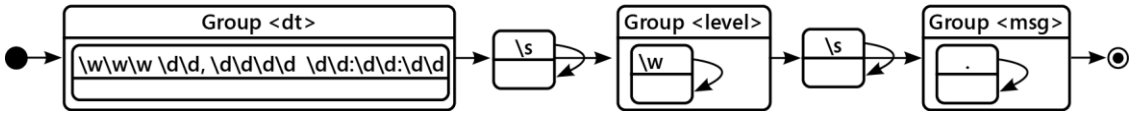
Chapter 9: Strings, Serialization, and File Paths

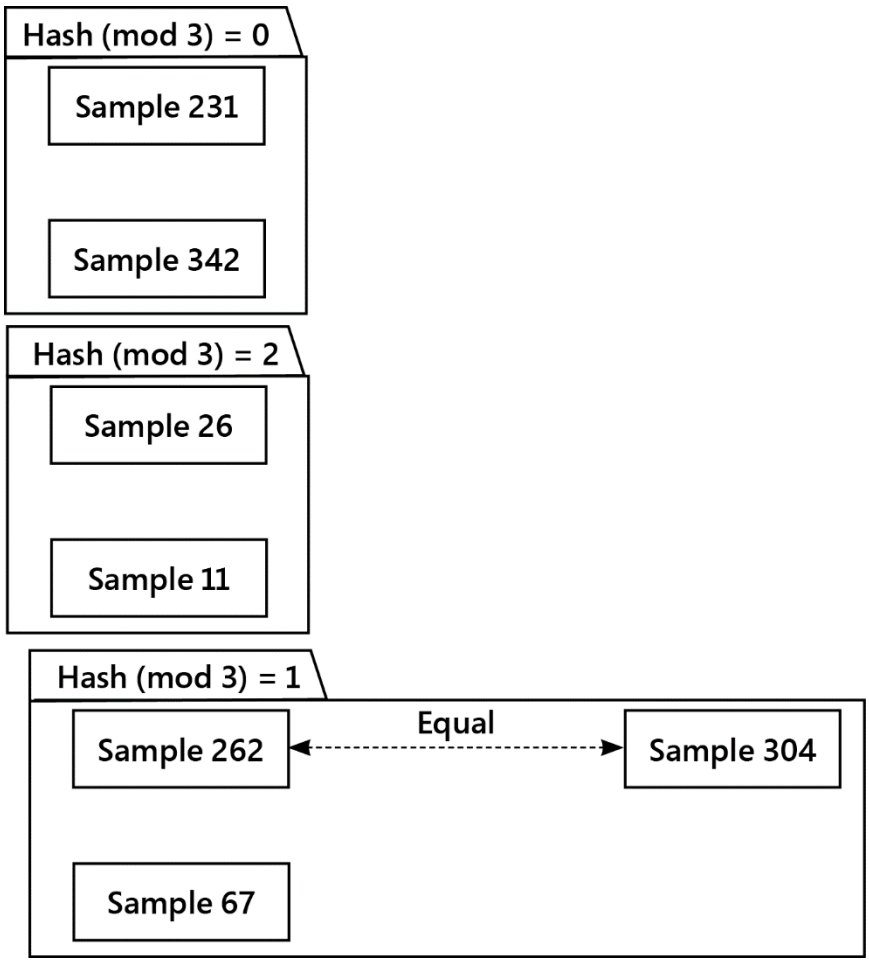


Chapter 10: The Iterator Pattern

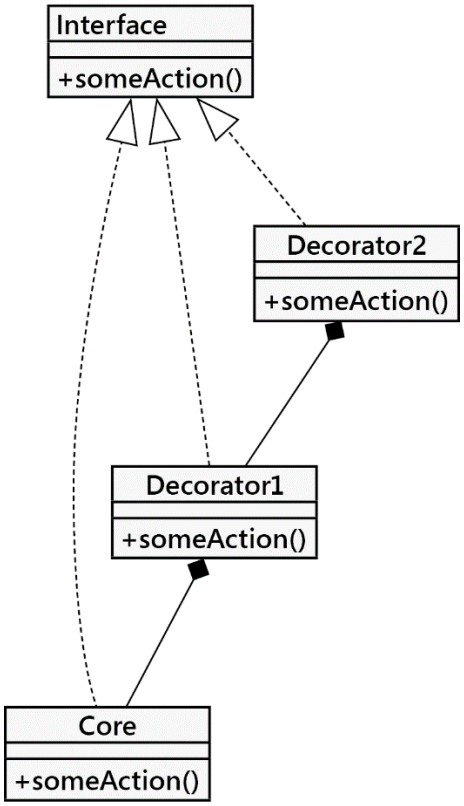


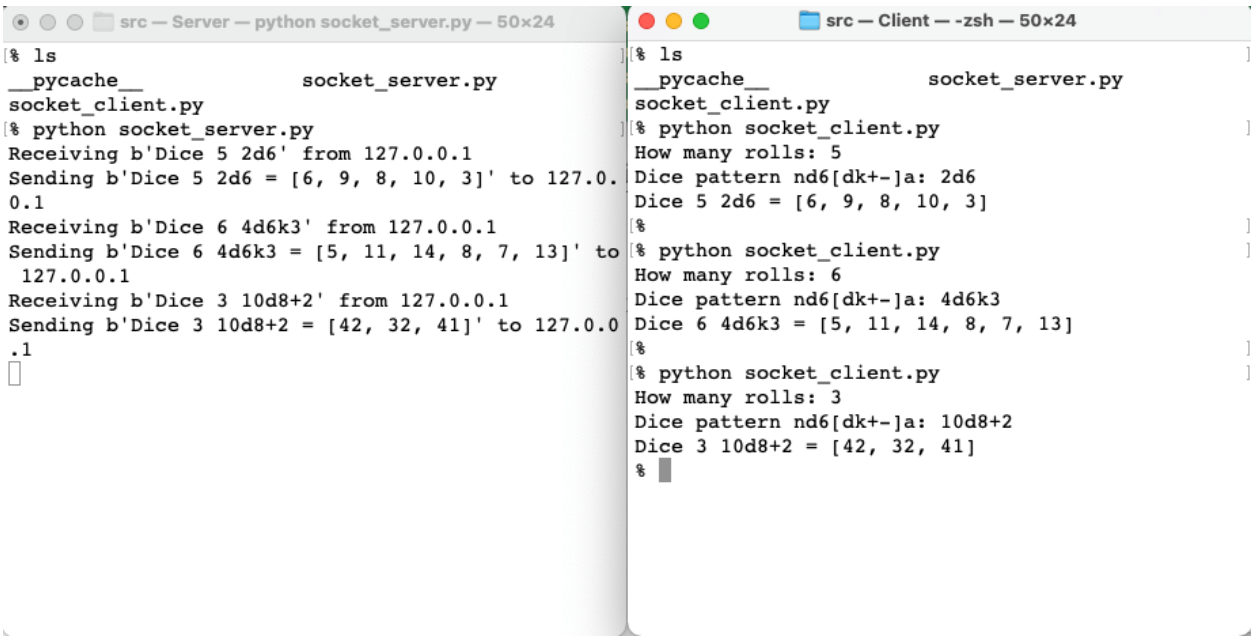


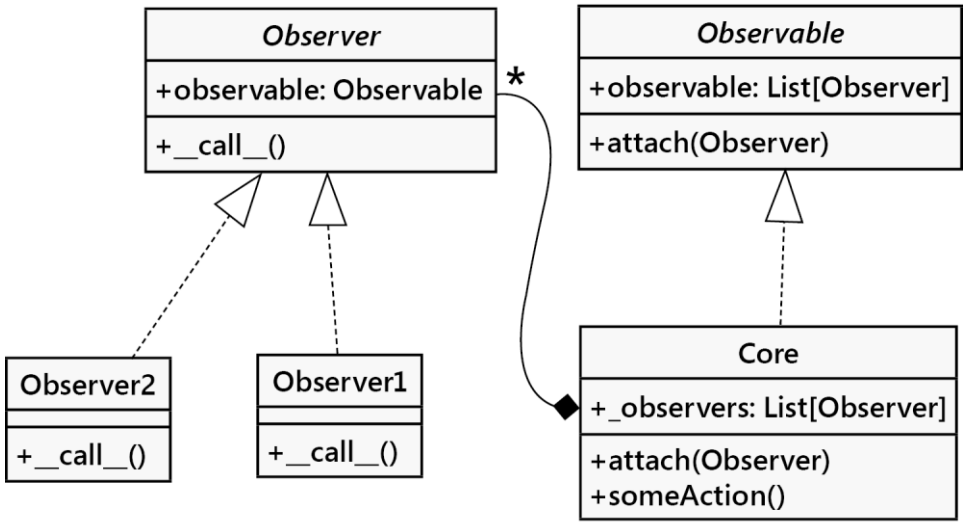


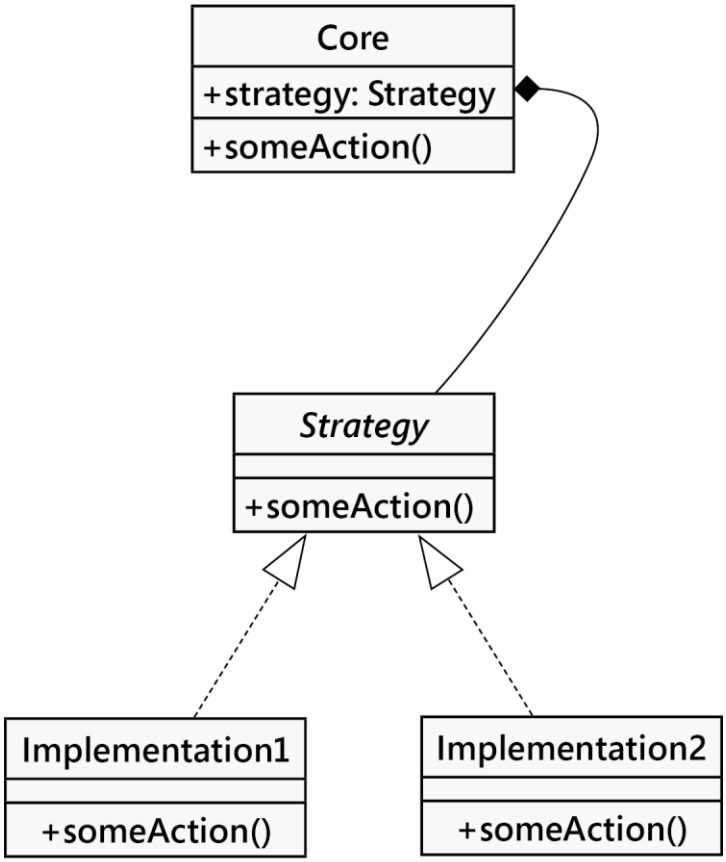

Hash (mod 3) = 0
Sample 231
Sample 342
Hash (mod 3) = 2
Sample 26
Sample 11
Hash (mod 3) = 1
Sample 262
Equal
Sample 304
Sample 67

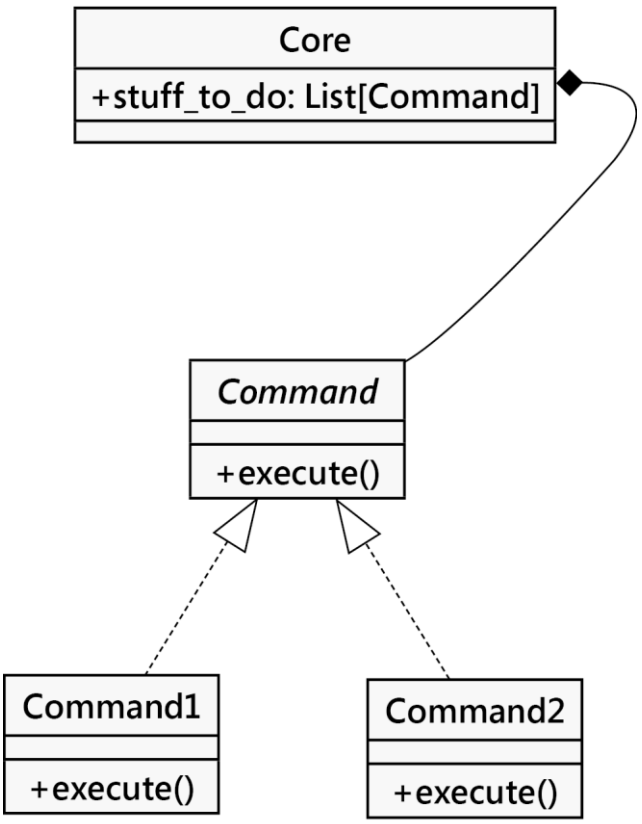
Chapter 11: Common Design Patterns

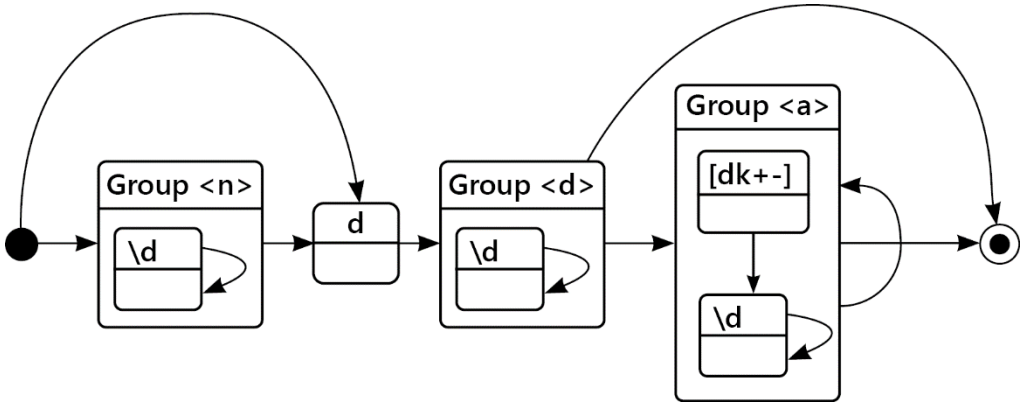


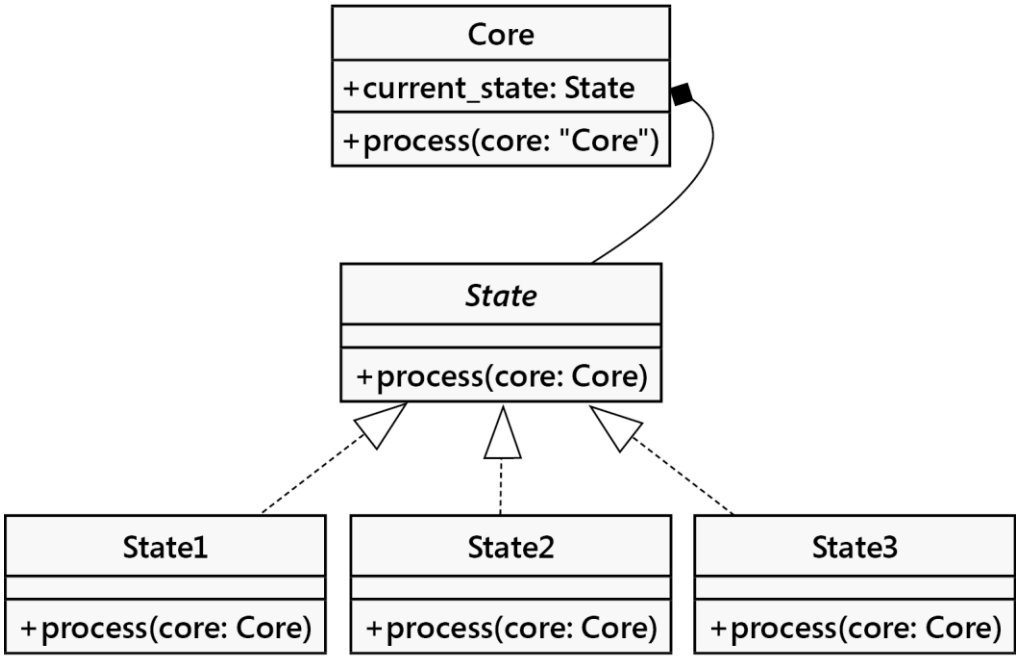


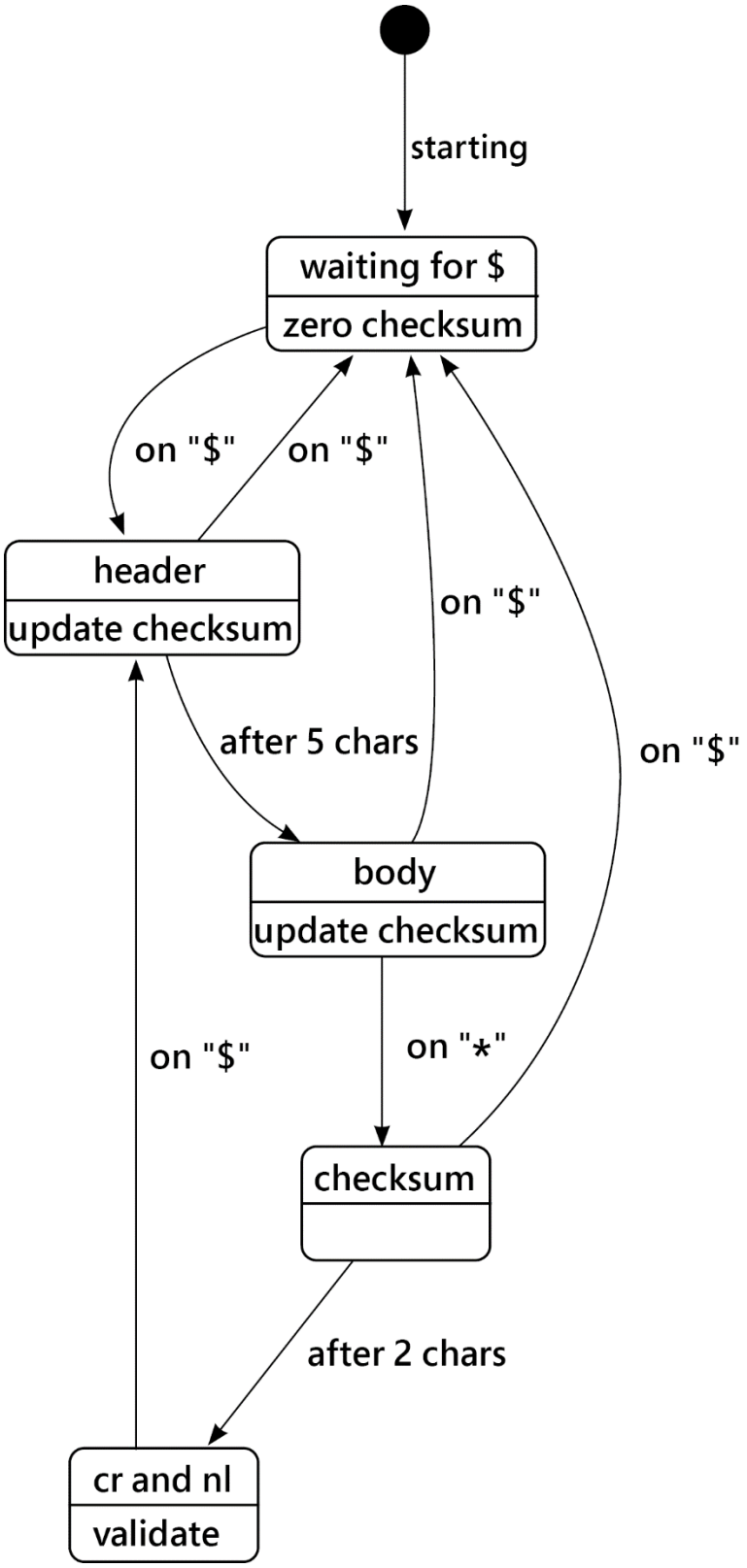

Observer
+observable: Observable
+__call__()
*
Observable
+observable: List[Observer]
+attach(Observer)
Observer2
+__call__()
Observer1
+__call__()
Core
+_observers: List[Observer]
+attach(Observer)
+someAction()

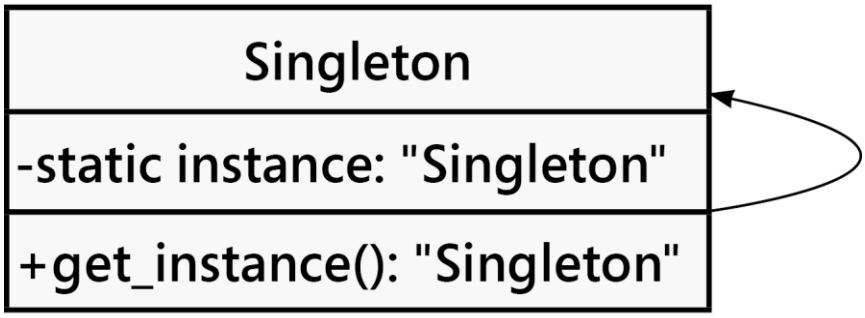

Core
+strategy: Strategy
+someAction()
Strategy
+someAction()
Implementation1
+someAction()
Implementation2
+someAction()

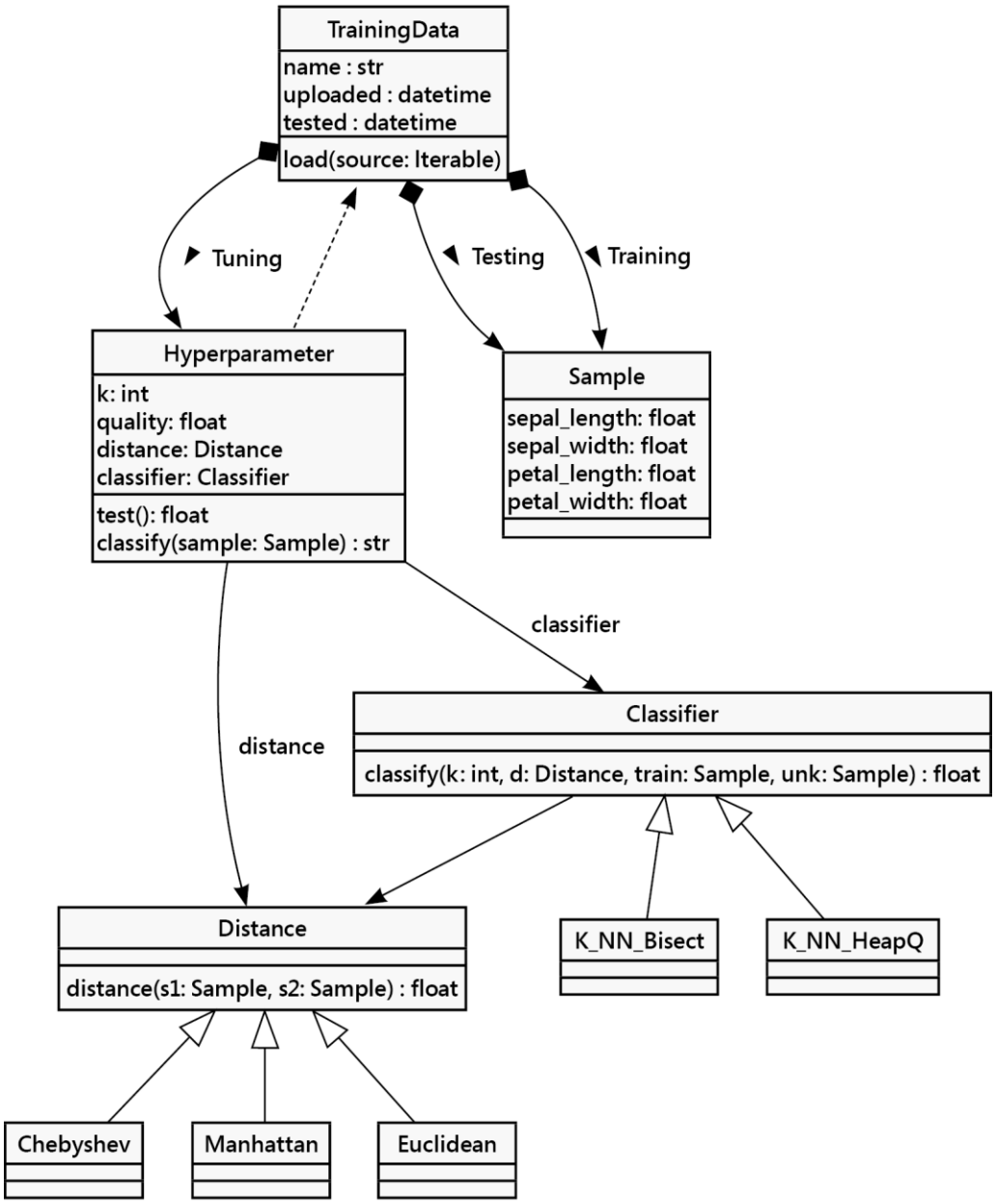

Core
+stuff_to_do: List[Command]
Command
+execute()
Command1
+execute()
Command2
+execute()


Group <n>
\d
d
Group <d>
\d
Group <a>
[dk+-]
\d

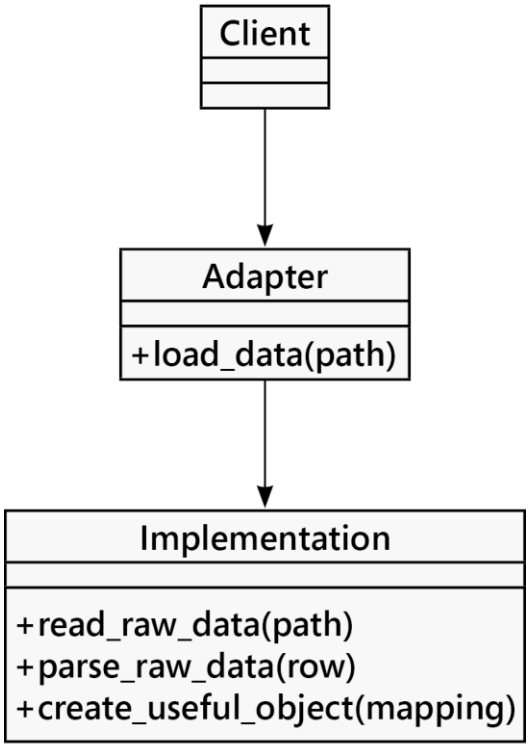

Core
+current_state: State
+process(core: "Core")
State
+process(core: Core)
State1
+process(core: Core)
State2
+process(core: Core)
State3
+process(core: Core)

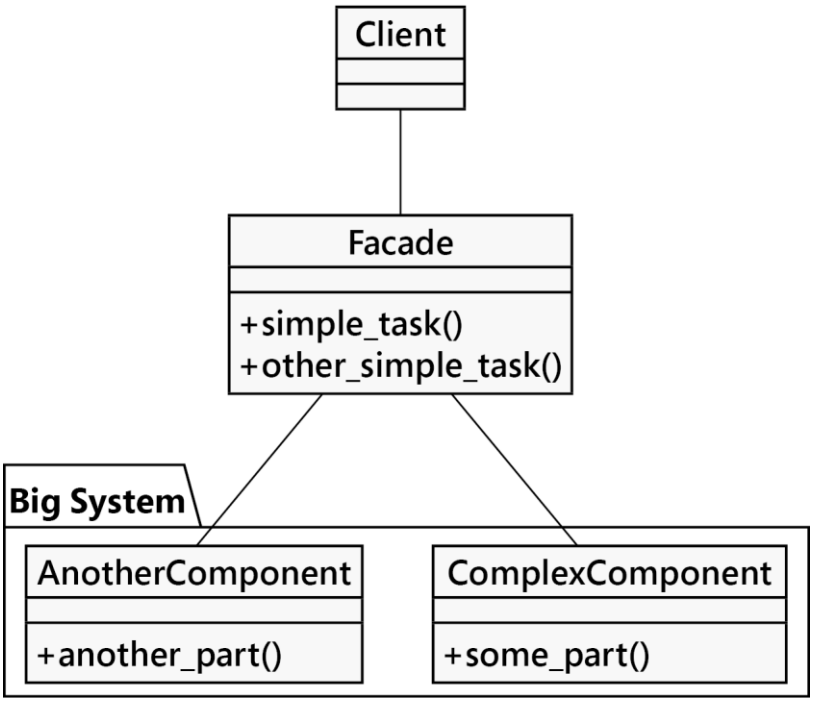

starting
waiting for $
zero checksum
on "$"
on "$"
header
update checksum
on "$"
after 5 chars
on "$"
body
update checksum
on "$"
on "*"
checksum
after 2 chars
cr and nl
validate

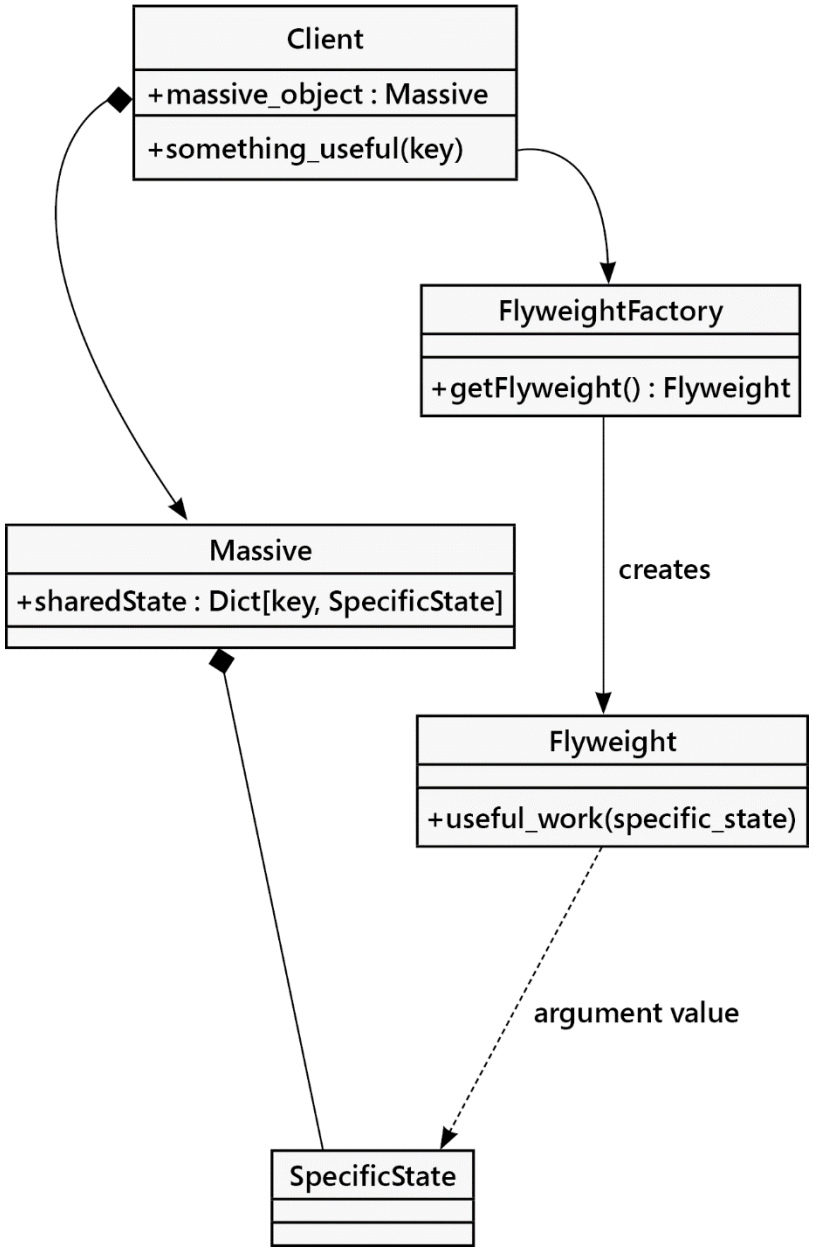

Singleton
-static instance: "Singleton"
+get_instance(): "Singleton"

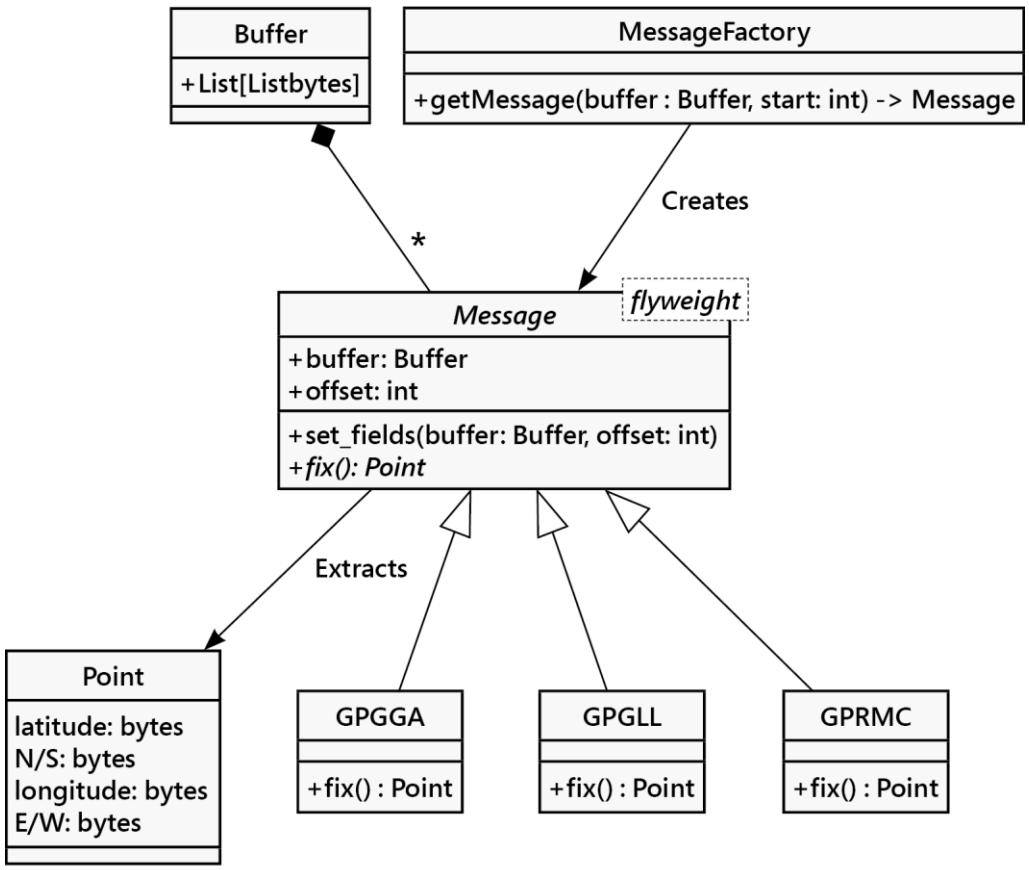

TrainingData
name : str
uploaded : datetime
tested : datetime
load(source: Iterable)
Tuning
Testing
Training
Hyperparameter
k: int
quality: float
distance: Distance
classifier: Classifier
test(): float
classify(sample: Sample) : str
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
classifier
distance
Classifier
classify(k: int, d: Distance, train: Sample, unk: Sample) : float
Distance
distance(s1: Sample, s2: Sample) : float
K_NN_Bisect
K_NN_HeapQ
Chebyshev
Manhattan
Euclidean

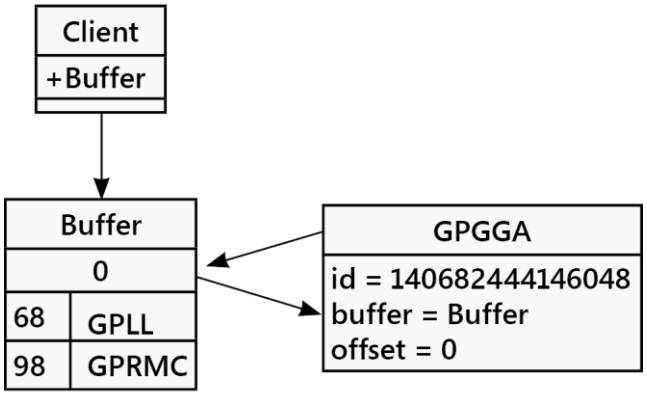
Chapter 12: Advanced Design Patterns

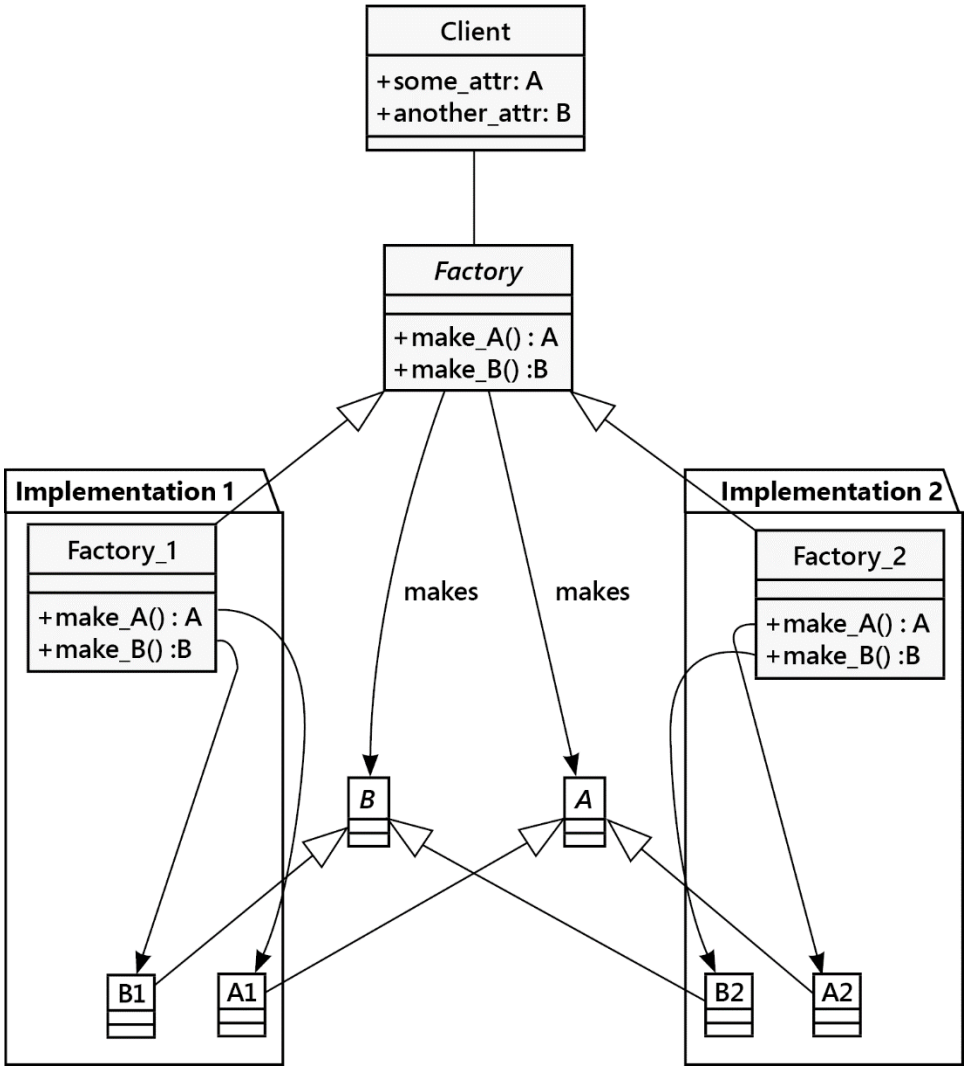


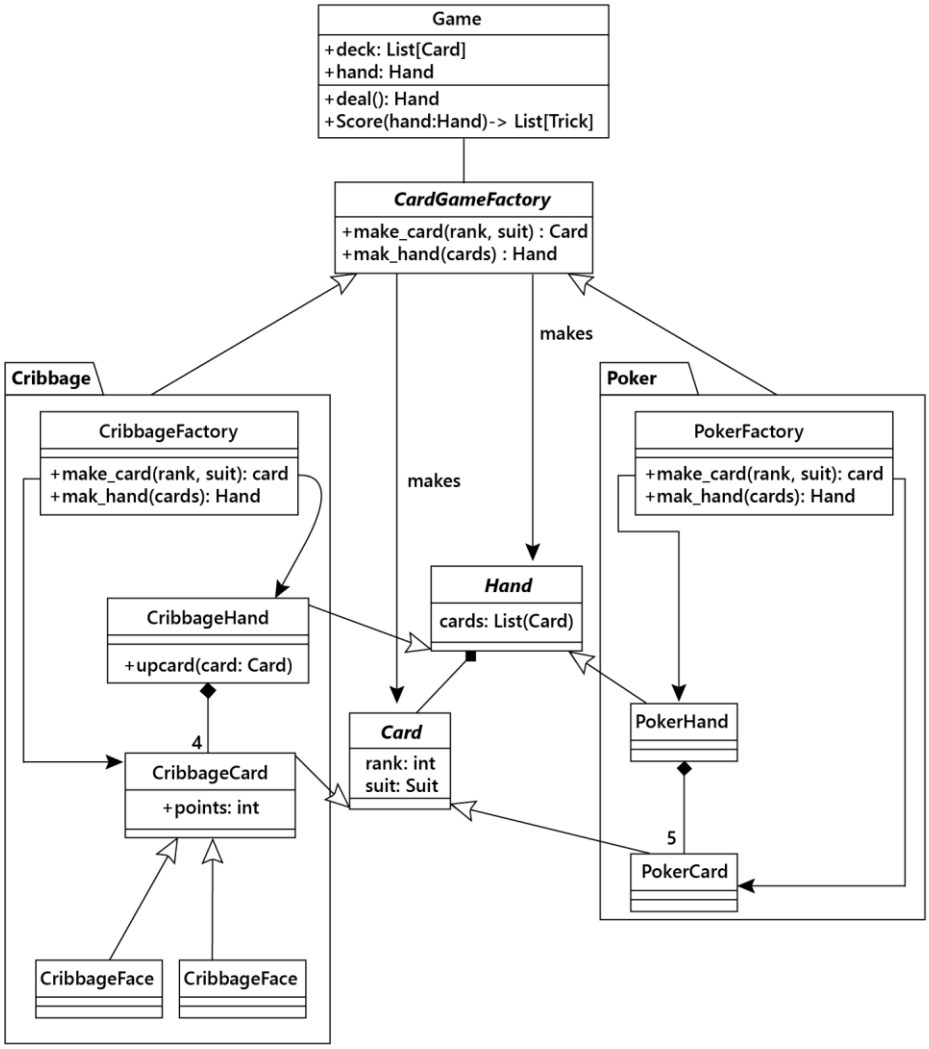


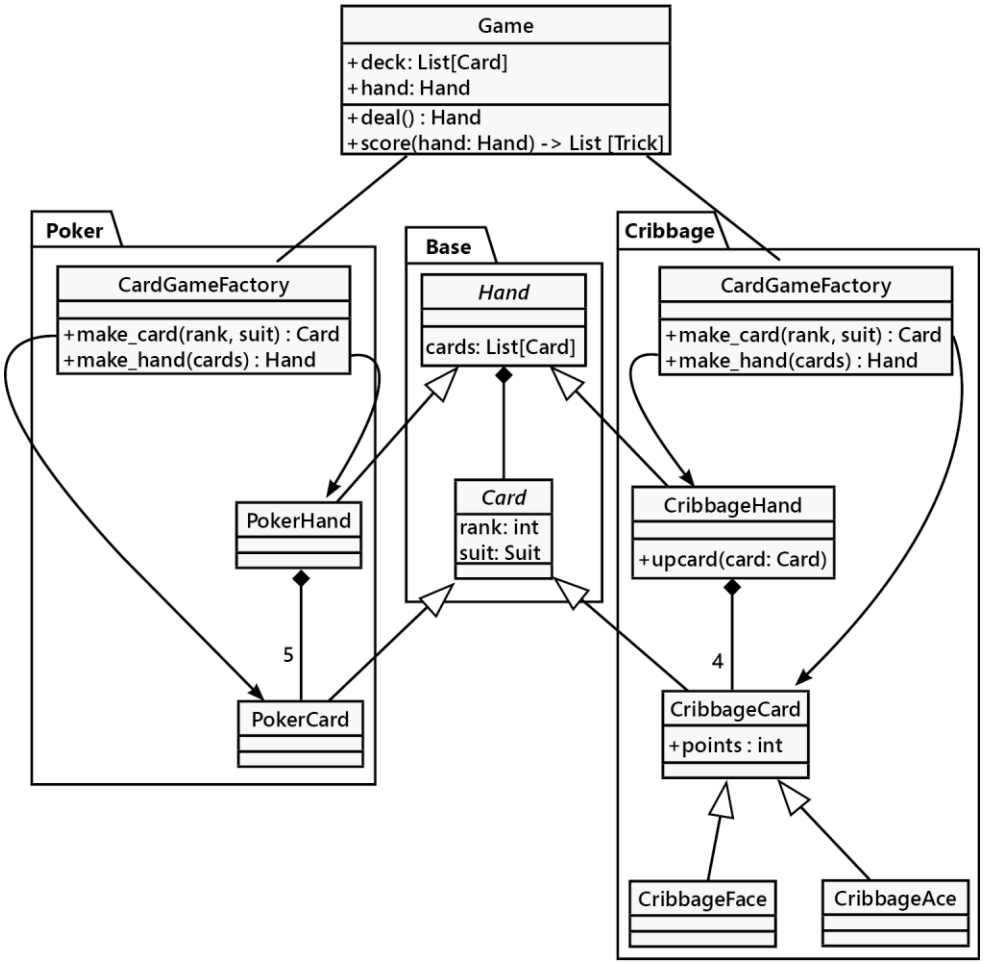

Client
+massive_object : Massive
+something_useful(key)
FlyweightFactory
+getFlyweight() : Flyweight
creates
Massive
+sharedState : Dict[key, SpecificState]
Flyweight
+useful_work(specific_state)
argument value
SpecificState

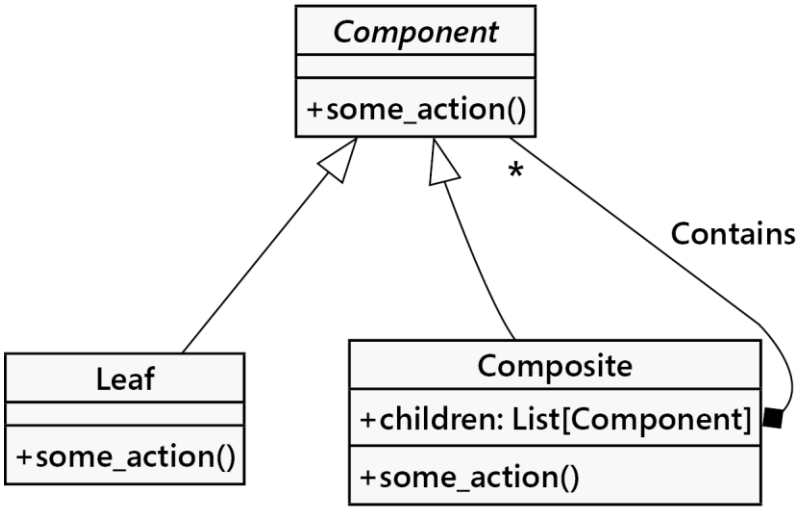

Buffer
+List[Listbytes]
MessageFactory
+getMessage(buffer : Buffer, start: int) -> Message
Creates
*
Message
flyweight
+buffer: Buffer
+offset: int
+set_fields(buffer: Buffer, offset: int)
+fix(): Point
Extracts
Point
latitude: bytes
N/S: bytes
longitude: bytes
E/W: bytes
GPGGA
+fix() : Point
GPGLL
+fix() : Point
GPRMC
+fix() : Point

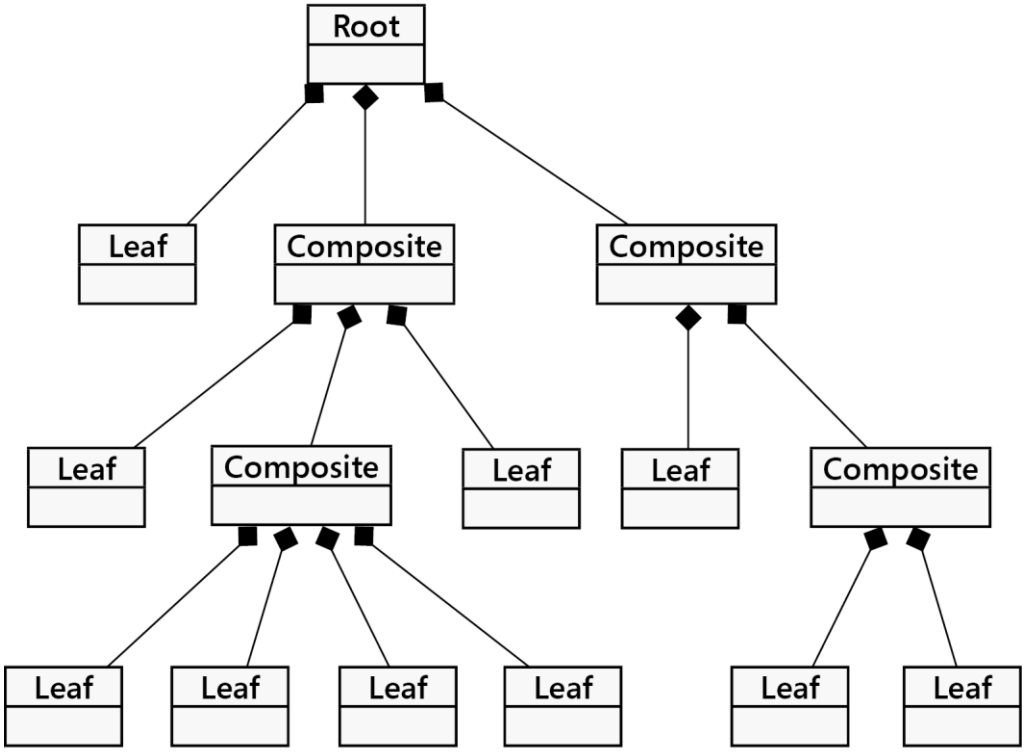

Client
+Buffer
Buffer
0
68
GPLL
98
GPRMC
GPGGA
id = 140682444146048
buffer = Buffer
offset = 0

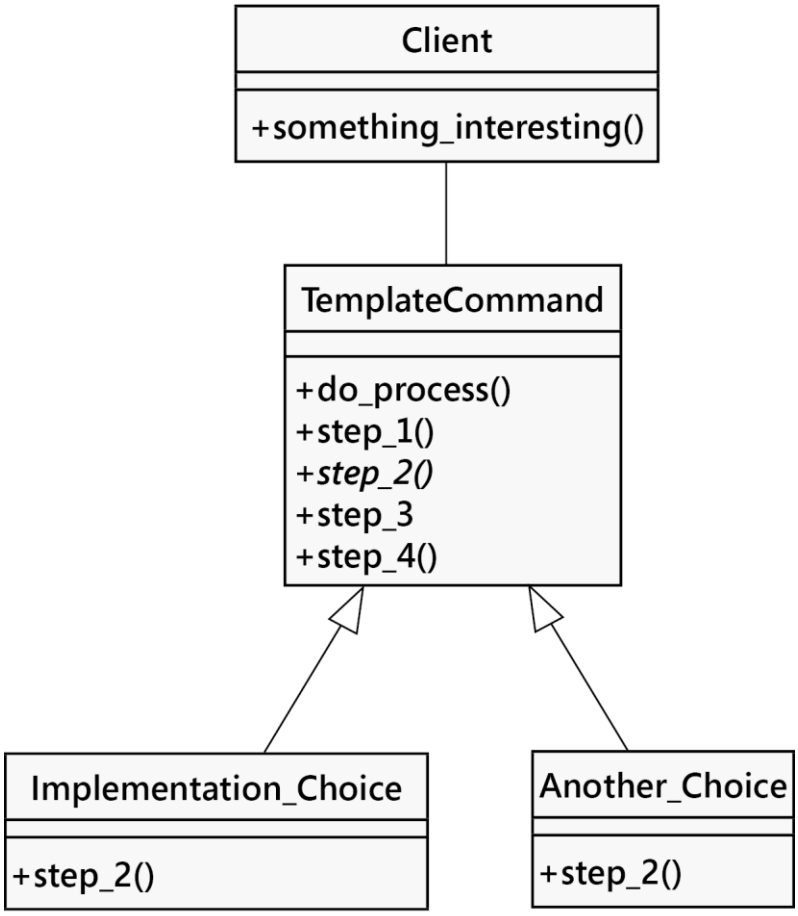

Client
+some_attr: A
+another_attr: B
Factory
+make_A() : A
+make_B() :B
Implementation 1
Factory_1
+make_A() : A
+make_B() :B
makes
makes
Implementation 2
Factory_2
+make_A() : A
+make_B() :B
B
A
B1
A1
B2
A2

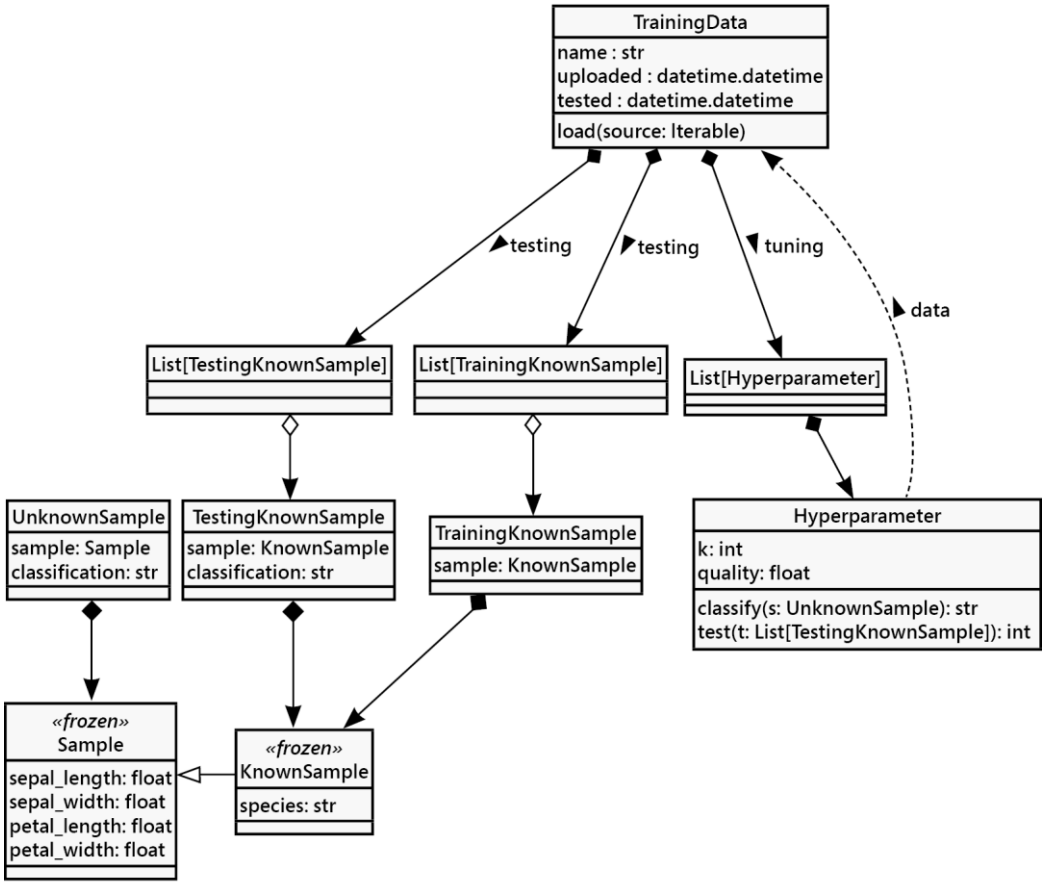

Game
+deck: List[Card]
+hand: Hand
+deal(): Hand
+Score(hand:Hand)-> List[Trick]
CardGameFactory
+make_card(rank, suit) : Card
+mak_hand(cards) : Hand
makes
makes
Cribbage
CribbageFactory
+make_card(rank, suit): card
+mak_hand(cards): Hand
CribbageHand
+upcard(card: Card)
4
CribbageCard
+points: int
CribbageFace
CribbageFace
Hand
cards: List(Card)
Card
rank: int
suit: Suit
Poker
PokerFactory
+make_card(rank, suit): card
+mak_hand(cards): Hand
PokerHand
5
PokerCard


Game
+deck: List[Card]
+hand: Hand
+deal() : Hand
+score(hand: Hand) -> List [Trick]
Poker
CardGameFactory
+make_card(rank, suit) : Card
+make_hand(cards) : Hand
PokerHand
5
PokerCard
Base
Hand
cards: List[Card]
Card
rank: int
suit: Suit
Cribbage
CardGameFactory
+make_card(rank, suit) : Card
+make_hand(cards) : Hand
CribbageHand
+upcard(card: Card)
4
CribbageCard
+points : int
CribbageFace
CribbageAce


Component
+some_action()
*
Contains
Leaf
+some_action()
Composite
+children: List[Component]
+some_action()


Root
Leaf
Composite
Composite
Leaf
Composite
Leaf
Leaf
Composite
Leaf
Leaf
Leaf
Leaf
Leaf
Leaf


Client
+something_interesting()
TemplateCommand
+do_process()
+step_1()
+step_2()
+step_3
+step_4()
Implementation_Choice
+step_2()
Another_Choice
+step_2()


TrainingData
name : str
uploaded : datetime.datetime
tested : datetime.datetime
load(source: Iterable)
testing
testing
tuning
data
List[TestingKnownSample]
List[TrainingKnownSample]
List[Hyperparameter]
UnknownSample
sample: Sample
classification: str
TestingKnownSample
sample: KnownSample
classification: str
TrainingKnownSample
sample: KnownSample
Hyperparameter
k: int
quality: float
classify(s: UnknownSample): str
test(t: List[TestingKnownSample]): int
«frozen»
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float
«frozen»
KnownSample
species: str

Chapter 13: Testing Object-Oriented Programs

20Object-Oriented%20Programming%204e

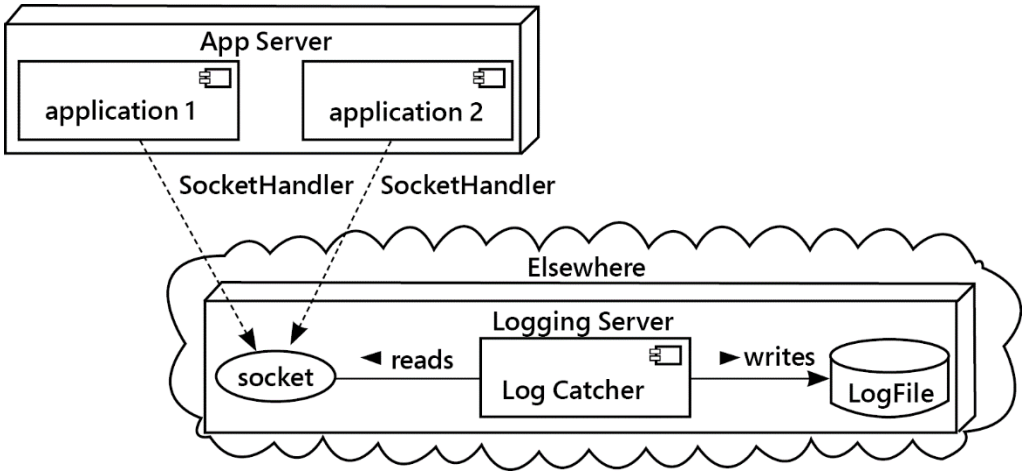
Coverage for **src/stats.py** : 42%

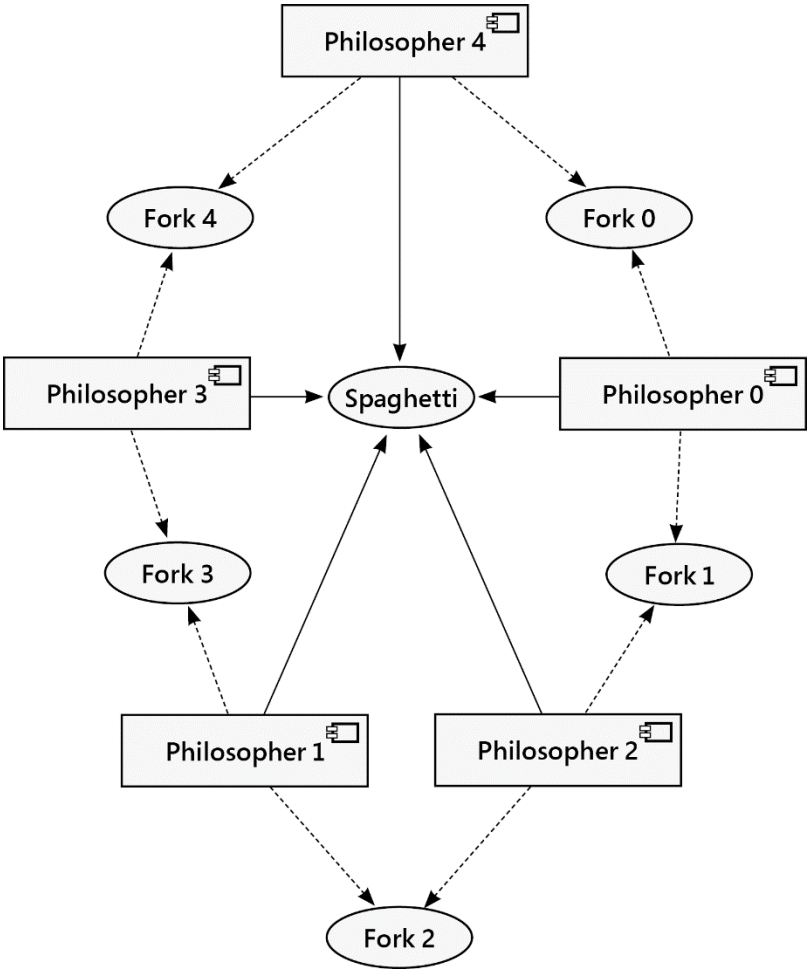
19 statements | 8 run | 11 missing | 0 excluded

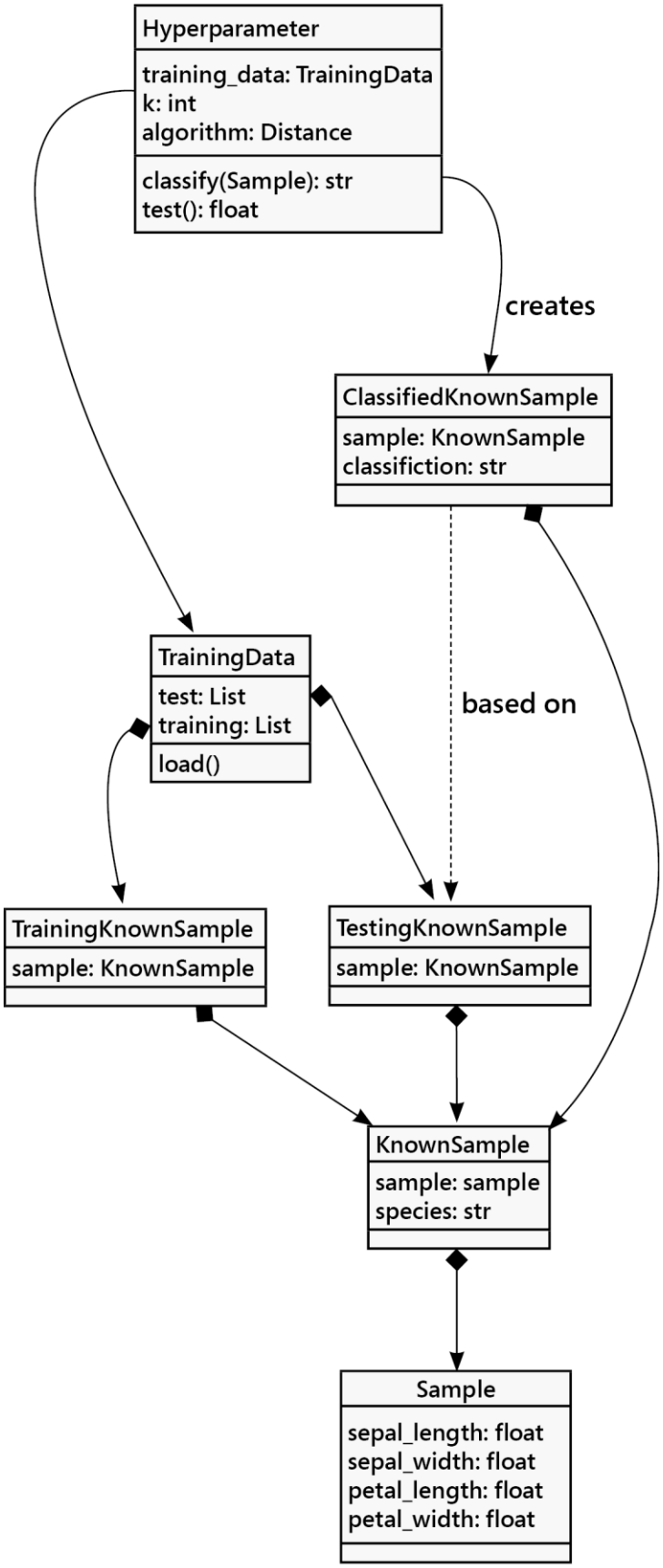
```
1  """
2  Python 3 Object-Oriented Programming
3
4  Chapter 13.  Testing Object-Oriented Programs.
5  """
6  import collections
7  from typing import DefaultDict, List, Optional
8
9
10 class StatsList(List[Optional[float]]):
11     """Stats with None objects rejected"""
12
13     def mean(self) -> float:
14         clean = list(filter(None, self))
15         return sum(clean) / len(clean)
16
17     def median(self) -> float:
18         clean = list(filter(None, self))
19         if len(clean) % 2:
20             return clean[len(clean) // 2]
21         else:
22             idx = len(clean) // 2
23             return (clean[idx] + clean[idx - 1]) / 2
24
25     def mode(self) -> List[float]:
26         freqs: DefaultDict[float, int] = collections.defaultdict(int)
27         for item in filter(None, self):
28             freqs[item] += 1
29         mode_freq = max(freqs.values())
30         modes = [item for item, value in freqs.items() if value == mode_freq]
31         return modes
```

« index coverage.py v5.4, created at 2021-02-01 20:51 -0500

Chapter 14: Concurrency






Hyperparameter
training_data: TrainingData
k: int
algorithm: Distance
classify(Sample): str
test(): float
creates
ClassifiedKnownSample
sample: KnownSample
classifiction: str
TrainingData
test: List
training: List
load()
based on
TrainingKnownSample
sample: KnownSample
TestingKnownSample
sample: KnownSample
KnownSample
sample: sample
species: str
Sample
sepal_length: float
sepal_width: float
petal_length: float
petal_width: float