

UML: Class Diagram

Diagramming your application classes

William Chan

Lead Platform Engineer, 605.tv, Capital One, FreeWheel (Comcast)

Objectives

- Learn how to sketch your application components in a class diagram
- Learn what components are in a class diagram

2

UML Diagram Diagrams

Lays out the sketches of classes, namely the follow

- **Public, private** and **protected** instance variables
- **Public, private** and **protected** instance methods
- The relationships of different classes

Use Case Diagram components

Components

- Packages - logical grouping of different classes that carry a common behavior
- Classes - classes that produces logical units of objects

Relations

- Extension - a class inherits from another class
- Composition - a class is composed of an object of another class

Note: *Terminology is taken from the PlantUML language*

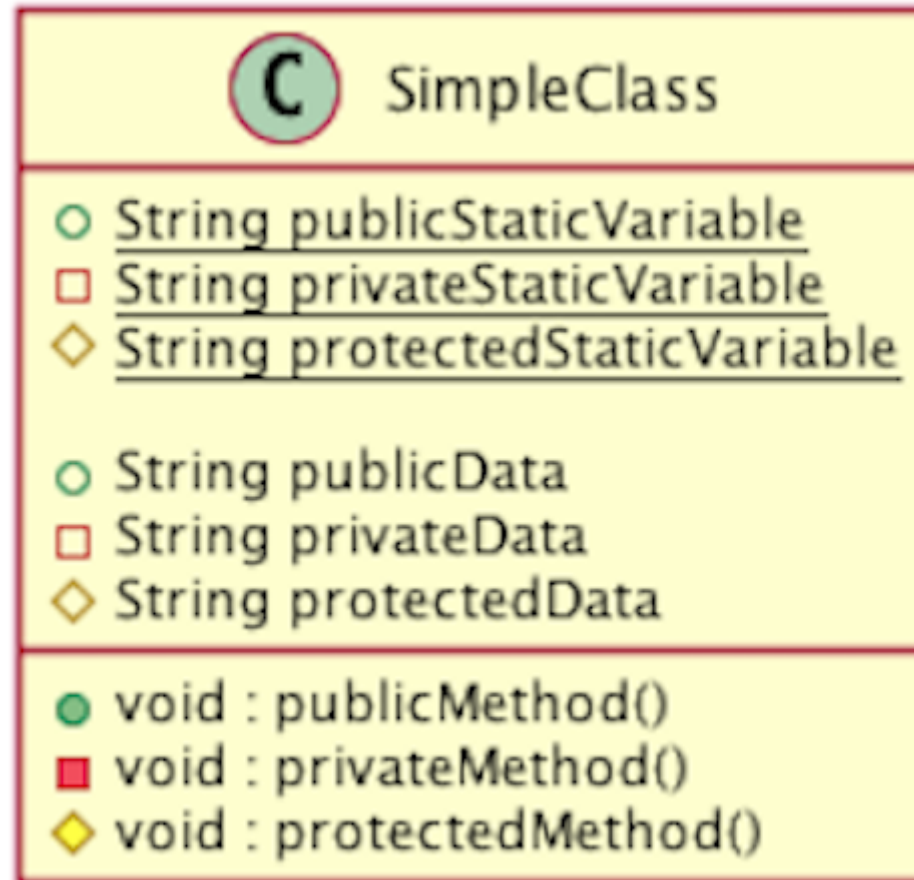
PlantUML Class Diagram Documentation

Documentation: <http://plantuml.com/class-diagram>

5

Diagram Tutorial

Simple Class Example











Simple Class Code

```
@startuml
class SimpleClass {
    + {static} String publicStaticVariable
    - {static} String privateStaticVariable
    # {static} String protectedStaticVariable

    + String publicData
    - String privateData
    # String protectedData

    + void : publicMethod()
    - void : privateMethod()
    # void : protectedMethod()
}
@enduml
```


Modifiers Legend

Character	Icon for field	Icon for method	Visibility
-			private
#			protected
~			package private
+			public

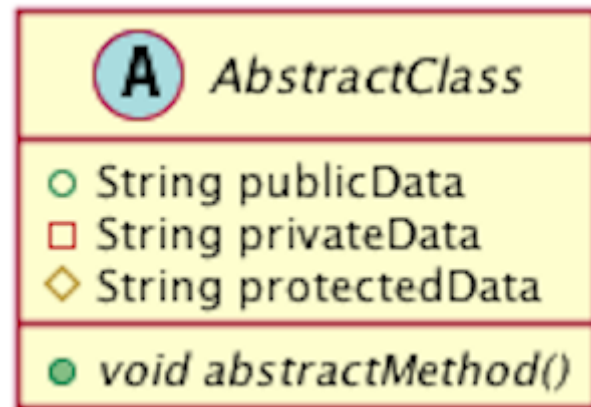
Abstract Classes and Interfaces

10

Abstract Class Example

An abstract class is a class with one or more abstract methods, methods without implementation where a class that extends it should implement the abstract method.

Note: objects can not be created from abstract classes



Abstract Class Code

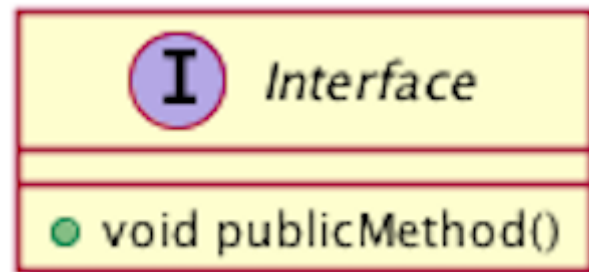
```
@startuml
abstract class AbstractClass {
    + String publicData
    - String privateData
    # String protectedData

    + {abstract} void abstractMethod()
}
@enduml
```

12

Interface Example

Interfaces don't have implementations but is a contract for which the class that **implements** it needs to adhere to, namely the parameter and return types.



Interface Code

```
@startuml
interface Interface {
    + void publicMethod()
}
@enduml
```

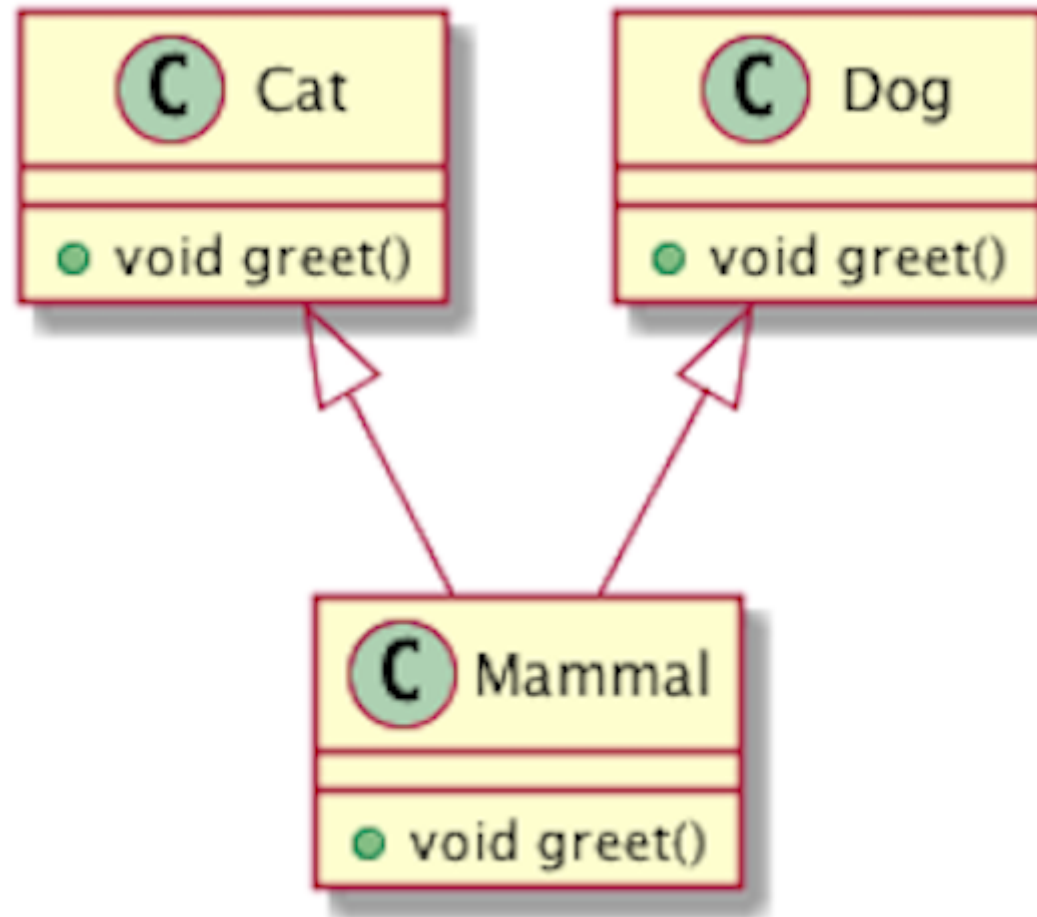
14

Class Relations

15

Class Inheritance Example

Inheritance represents the **is a** relationship. For instance, a cat or a dog is a mammal.



Class Inheritance Code

```
@startuml
class Mammal {
    + void greet()
}

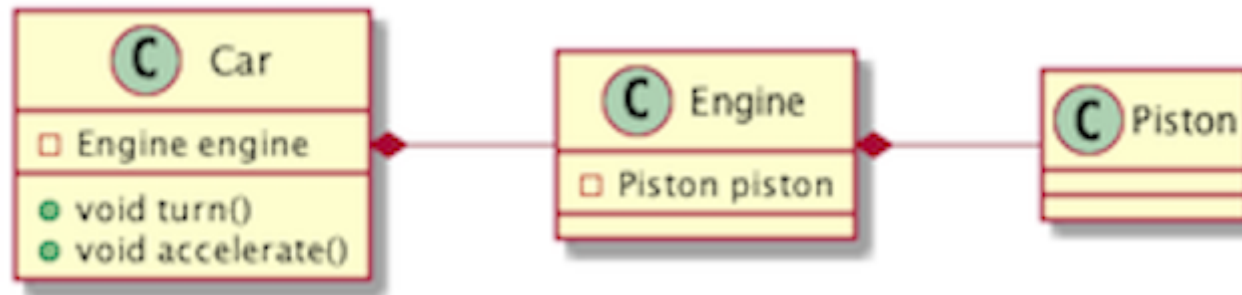
class Cat {
    + void greet()
}

class Dog {
    + void greet()
}

Cat <|-- Mammal
Dog <|-- Mammal
@enduml
```

Class Composition Example

Composition represents the **has an instance** relationship. For instance, a car has an engine.



18

Class Composition Code

```
@startuml
left to right direction

class Piston
class Engine {
    - Piston piston
}

class Car {
    - Engine engine

    + void turn()
    + void accelerate()
}

Car *-- Engine
Engine *-- Piston

@enduml
```

Packages

20

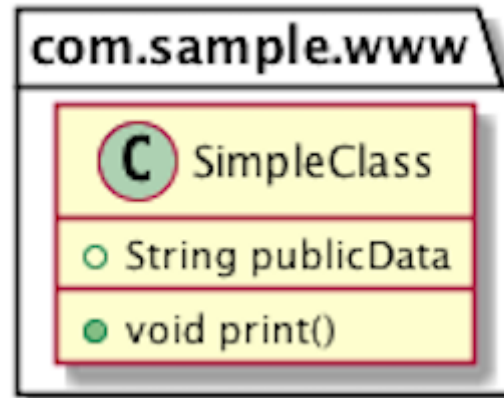
What are packages?

Packages are a way to organize class that are similar in functionality because it works with the same domain.

For instance, an application may have third party API integrations and can group all APIs by functional aspect. They may also be grouped to a specific domain, such as a package for Twitter integration and another package for Microsoft integrations.

21

Packages Example



Packages Code

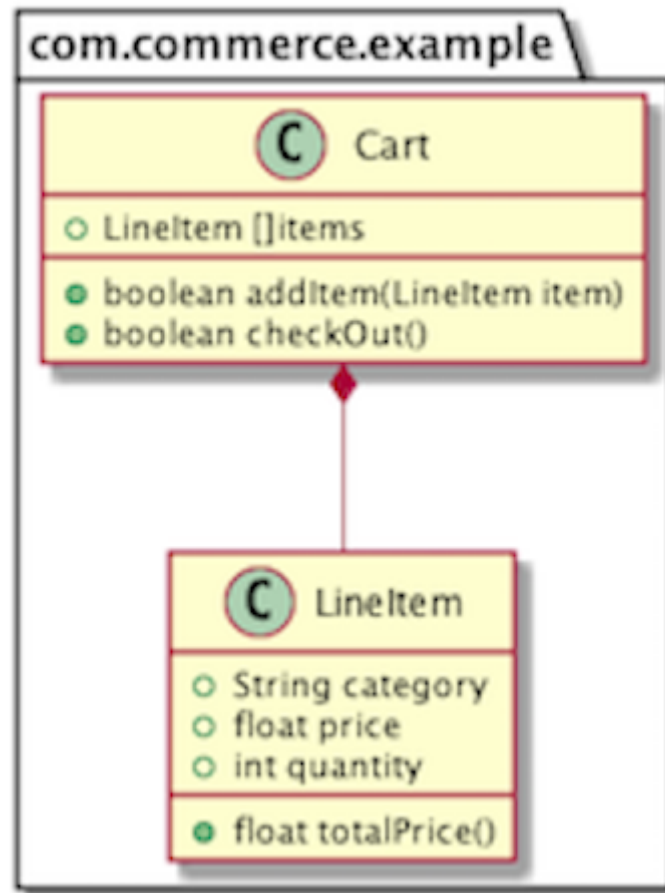
```
@startuml
namespace com.sample.www {
    class SimpleClass {
        + String publicData
        + void print()
    }
}
@enduml
```

23

A More Complete Example

24

UML Example: E-commerce Diagrams



UML Example: E-commerce Diagram PlantUML Code

```
@startuml
namespace com.commerce.example {
    class LineItem {
        + String category
        + float price
        + int quantity

        + float totalPrice()
    }

    class Cart {
        + LineItem []items
        + boolean addItem(LineItem item)
        + boolean checkOut()
    }

    Cart *-- LineItem
}
@enduml
```

Summary

- Class diagrams help you visualize the components that make up your application

27

Thank you

William Chan

Lead Platform Engineer, 605.tv, Capital One, FreeWheel (Comcast)

<http://linkedin.com/in/wchan2> (<http://linkedin.com/in/wchan2>)

