Unified Modelling Language

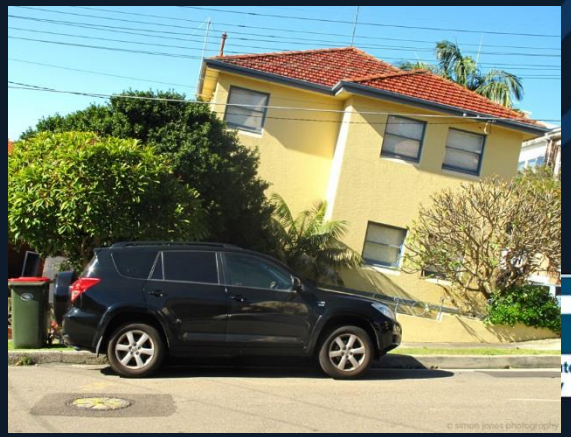
Visualising code

Contents

- Why UML?
- What is UML?
- Class diagrams
  - Classes
  - Attributes
  - Operations
  - Associations
  - Aggregation
  - Composition

The Case For Blueprints

- Imagine building a house without a blueprint
  - Measurements will be wrong.
  - A poor foundation can mean an unstable structure.
  - Each worker will have a different vision of the final product.
  - A lot of work will have to be redone.



Why Use UML?

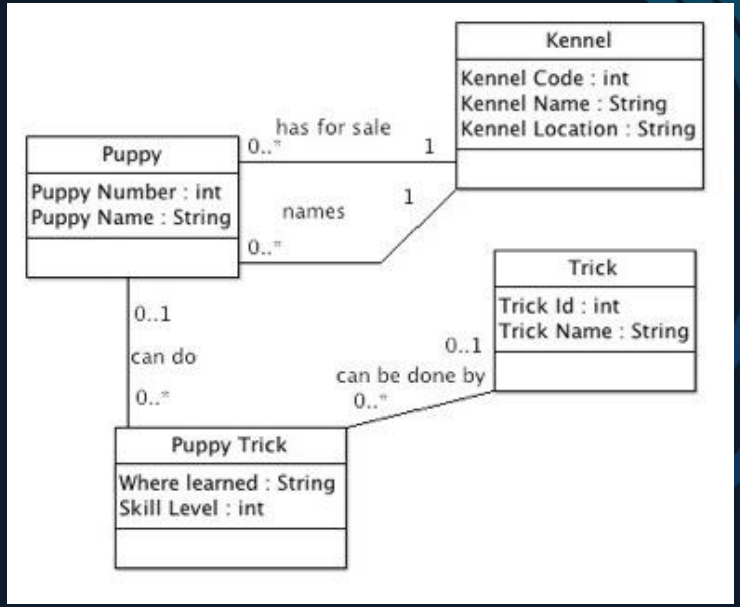
- Building software without a blueprint is similar
  - Unforeseen problems can lead to a lot of code refactoring.
  - Refactoring takes time.
  - Less time means fewer features and less polish.
  - Refactoring increases the chance of bugs.
  - New team members won't know how things should work.
  - Lack of planning leads to complex code that is difficult to work with and re-use

What is UML?

- UML diagrams are blueprints for software
  - It's an industry standard.
  - Like pseudo code it's independent of any language.
  - Helps to visualise the application.
  - Highlights problems before implementation.
  - New members can quickly grasp the software design.
  - Helps to break an application down into tasks.

UML Class Diagrams

- UML has many different types of diagrams, but the most useful the class diagram.
- This outlines all the classes (or objects) in our game and how they will relate to one another.
- This is one of the best ways to outline the architecture of the code

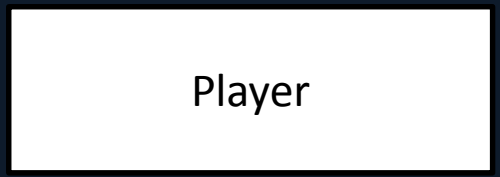


Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

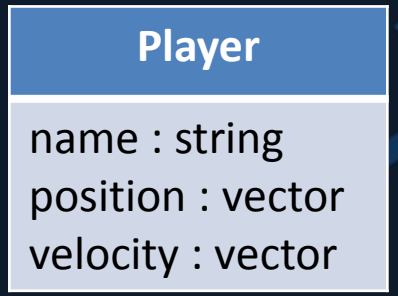
Classes

- In the previous lecture we practised identifying the objects that make up a problem
- Classes are the blueprints for these objects.
- The simplest way of representing classes in UML is with a box:



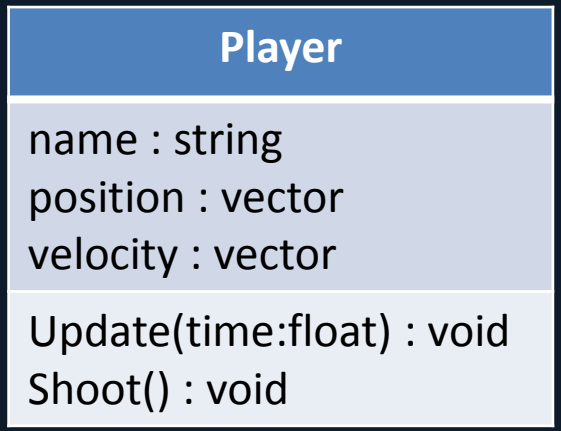
Attributes

- Attributes describe the data contained in a class
- These are the variables!
- They are placed under the class name in a class diagram, and are often listed with a type



Operations

- Operations are the functions or methods that belong to a class.
- These are placed below the attributes in the diagram
- We can also specify the return types of these functions, though it is not necessary.



Associations

- Associations represent the relationships between classes
- Classes are associated if:
  - Class A sends a message to class B
  - Class A owns or creates an instance of class B
  - Class A calls a function belonging to class B
- In short, if any part of class A needs to know about any part of class B
- These are represented by a solid line with no arrow

Associations

- Associations can be labelled
- We can also use the associations to specify the amount of one class in relation to the other
- In the example on the right, one player is going to use many bullets. So the relationship is 1 to many.

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Aggregation

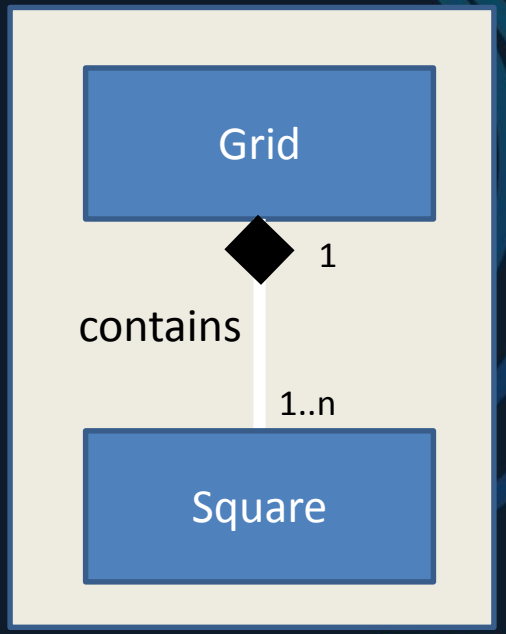
- An aggregation is a more specific kind of association between two classes.
- We use it when one class is part of another class, but the two classes could still possibly exist on their own.
- For example a player has bullets, but both the player and the bullets could still exist if they are not linked together.
- Aggregation is represented by an open diamond on the side of the class that contains the other class.

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Composition

- Composition is more specific than aggregation, however in this case class A *owns* class B and class B could not exist without class A. In this case, Grid contains one or more Squares.
- This is represented by a closed diamond.

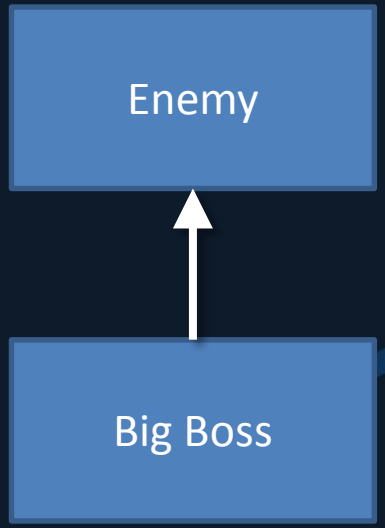


Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Generalisation

- This is another relationship that can happen between classes.
- This relationship occurs when one class is based on another, it contains some of the same attributes and operations.
- This is represented by a solid line with an open arrow pointing to the class that forms the base of the relationship
- This is also referred to as an “is a” relationship. If you can say class B is a class A then it’s correct. In this case, Big Boss *is an* Enemy.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- UML is a great way to visualise a project and share the vision with all team members.
- Also a great way to share the project concepts with non-technical team members and share holders.
- Class diagrams represent the architecture of a system – all the classes and their relationships between them.

References

- Stevens, P 2006, *Using UML: Software Engineering with Objects and Components*, Addison Wesley

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX