Contents

Introduction to UML

Object Orientated Design

- Criteria to be an object
  - Has Attributes
  - Has Behaviours
  - Can have an identity

Design

- Object-orientated programs work by making objects calloborate with each other, manipulating state and behaviour, in some particular order

  How do you express this and abstract your program to explain to someone who doesn't code

Communicating Design

- Design on a whiteboard mostly ends up as a set of cirles, boxes and lines
- We need a common language to communicate so every object can *do* the right thing at the right *time*

UML

- A common (mostly diagrammatic) language to describe a system
  - Structure
  - Behaviour

Tyoes

- Structural view
  - Class diagram
  - Package diagram
  - Component diagram
- Behavioural view
  - Activity diagram
  - Sequence diagram
  - Use Case diagram

Structural view

- Depicts at various leels of abstraction, the way code is arranged in a system

Types of UML Diagrams

UML Class Diagram Cheat Sheet

Generalization

Realization

Class
-privateMember: int
#protectedMember: char
+publicMember: bool
+publicMethod(): void
+publicVirtualMethod(parameter:int): void
+staticMethod(): void
-privateMethod(): bool
#protectedMethod(value:int): char

ParentClass
+virtualMethod(): void

is a

ChildClass
-kid: bool
+virtualMethod(): void

Interface
+virtualMethod(): void

implements

Class
-kid: bool
+virtualMethod(): void

Association

Class 1 1..*

1 Class 2

Association

Note

Aggregation

Class 1 1

has a

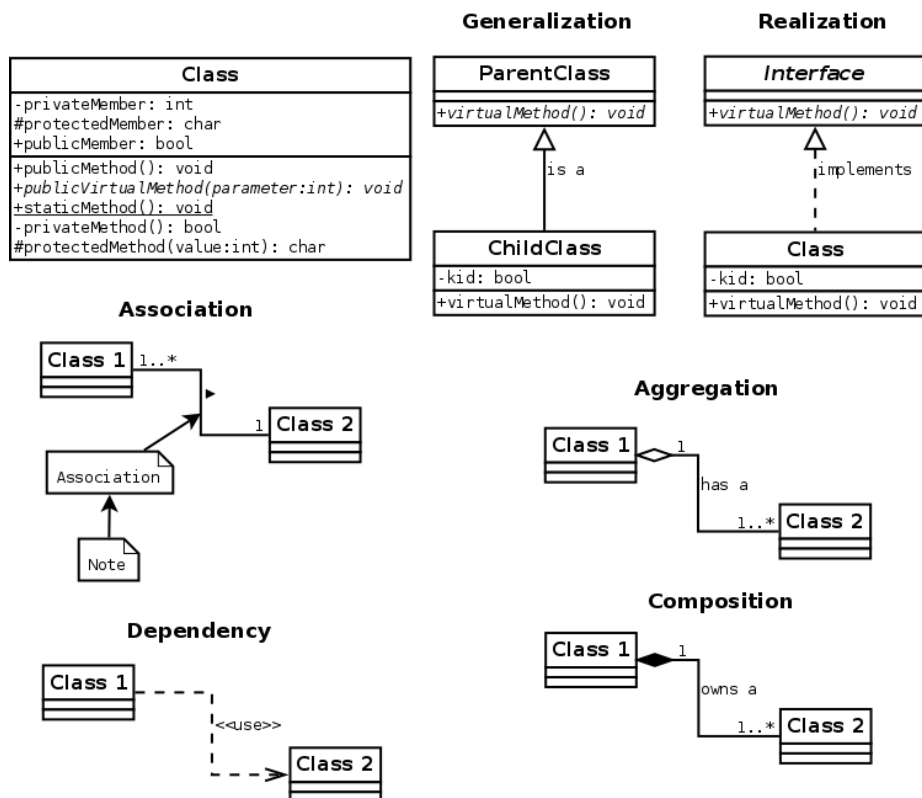1..* Class 2

Dependency

Class 1

<<use>>

Class 2

Composition

Class 1 1

owns a

1..* Class 2

Figure 1: UML Cheat Sheet

3