

Design Patterns and UML

Class and Object diagrams

Linda Marshall

Department of Computer Science
University of Pretoria

17 August 2021

What are design patterns?

Definition 1

Patterns identify and specify abstractions that are above the level of single classes and instances, or of components.

Gamma et al (1995) [Gang of Four (GoF)]
(Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides)

Definition 2

Design Patterns constitute a set of rules describing how to accomplish certain tasks in the realm of software development.

Pree (1995)

Definition 3

Design Patterns focus more on reuse of recurring architectural design themes, while frameworks focus on detail design and implementation.

Coplien and Schmidt (1995)

Definition 4

A pattern addresses a recurring design problem that arises in specific design situations and presents a solution to it.

Buschmann et al (1996)

Definition 5

Design Patterns are recurring solutions to design problems you see over and over.

The Smalltalk Companion (1998)

Definition 6

Experienced OO developers build up a repertoire of general principles and idiomatic solutions that guide them in the creation of software. These may be called patterns.

Craig Larman(2006)

Definition 7

Design Patterns are programming tools to improve code to be:

- easier to implement, and
- easier to maintain.
- are good answers to common and specialised problems.
- define a common (programming language independent) programming model that standardise common programming tasks into recognisable forms, giving your projects better cohesiveness.

CG Lasater (2007)

When design patterns are applied we achieve:

- Improved maintainability of code
- Improved adaptability of code
- Improved reliability of code
- Programmers who are more effective in their work.

There are 23 classic patterns, identified by the GoF, categorised as:

- **Creational** - creation of objects
- **Behavioural** - interaction between objects
- **Structural** - composition of objects

- **Creational** - Factory Method, Abstract Factory, Builder, Prototype, Singleton
- **Behavioural** - Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy, Visitor
- **Structural** - Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy

The Sacred Elements of the Faith

the holy
origins

the holy
structures

107 FM Factory Method								139 A Adapter
117 PT Prototype	127 S Singleton					223 CR Chain of Responsibility	163 CP Composite	175 D Decorator
87 AF Abstract Factory	325 TM Template Method	233 CD Command	273 MD Mediator	293 O Observer	243 IN Interpreter	207 PX Proxy	185 FA Façade	
97 BU Builder	315 SR Strategy	283 MM Memento	305 ST State	257 IT Iterator	331 V Visitor	195 FL Flyweight	151 BR Bridge	

the holy
behaviors

Taken from <http://www.vincehuston.org/dp/>

Each design pattern is further categorised by the strategy used in the implementation.

The two implementation strategies are *Delegation* and *Inheritance*.

The relationships between objects are the main influencers of the delegation strategy, while the relationships between classes are the main focus of the inheritance strategy.

Category	Pattern	Strategy	
		Delegation	Inheritance
Creational	Factory Method		X
	Abstract Factory	X	
	Prototype	X	
	Builder	X	
	Singleton	X	
Behavioural	Memento	X	
	Template Method		X
	Strategy	X	
	State	X	
	Observer	X	
	Iterator	X	
	Mediator	X	
	Command	X	
	Chain of Responsibility	X	
	Interpreter		X
	Visitor	X	
Structural	Composite	X	
	Decorator	X	
	Adapter	X	X
	Bridge	X	
	Façade	X	
	Proxy	X	
	Flyweight	X	

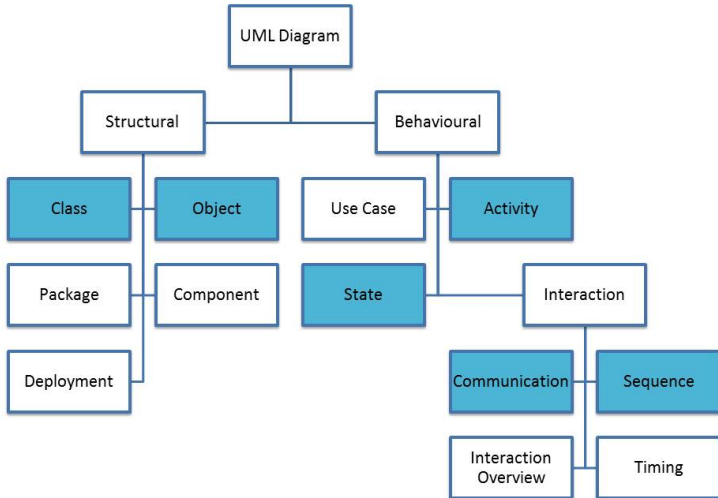
Unified Modelling Language (UML) is a standard notation for the modelling of real-world objects as a first step in developing an object-oriented design methodology.

<http://searchsoftwarequality.techtarget.com/definition/Unified-Modeling-Language>

UML *unifies* three OO methodologies:

- OMT - 1991 - James Rumbaugh
- OOA and OOD - early 1990's - while Grady Booch worked at Rationale
- OOSE - 1992 - Ivor Jacobson

The ideas of these “Three Amigos” and others, under the sponsorship of Rationale, lead to UML in 1995. UML 2 adapted by OMG in 2005. UML 2.5 released in June 2015.

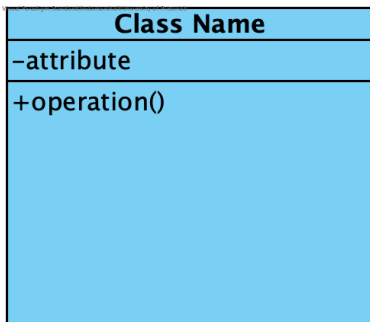


A *Class diagram* is a UML structural diagram which shows the structure of the designed system at the level of classes and interfaces. Class diagrams show features, constraints and relationships – associations and generalisations etc. – of the classes and interfaces.

A class is a classifier which describes a set of objects that share the same:

- features
- constraints
- semantics (meaning)

Features (or fields or members) of a class are **attributes** (fields) and **operations** (or functions or methods).



+ (public), # (protected) and – (private)
are used to specify the **visibility** of the
respective features.

*Always list features in order from public to
private in each of the sections of the class.*

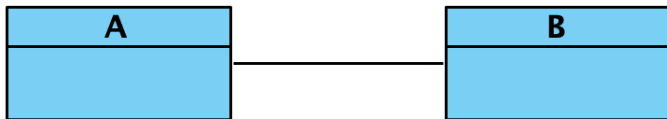
Static members are underlined.

SearchService
-config : Configuration
-engine : SearchEngine
+search(query : SearchRequest) : SearchResult
<u>+createEngine() : SearchEngine</u>

Set the Scope to “classifier” in Visual Paradigm

Associations are a type of relationship between objects of classes that model **delegation**. An object of a class makes use of an object of another class. These associations are modelled in a class diagram to show how objects of the classes will interact with one another.

Visual Paradigm Standard(lindamarshall@University of Pretoria)



There is an association between class A and class B.

An association exists when an object:

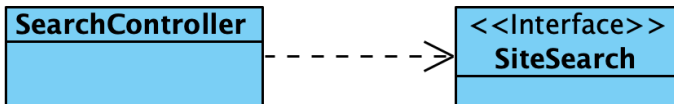
- accepts an object of another class as a parameter in one of its operations.
- instantiates an object of another class in one of its operations, or
- of a class holds an attribute of another class.

Different levels of coupling can be represented by associations. These include:

- Dependency (*uses-a* relationship)
- Aggregation (*has-a* relationship)
- Composition (*owns-a* relationship)

Dependency (uses-a)

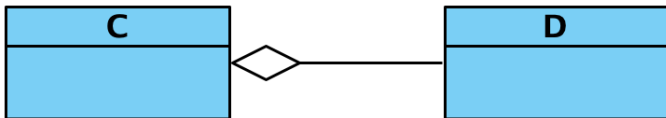
Visual Paradigm Standard (Indamarschal/University of Pretoria)



Interface SiteSearch is used (required) by
SearchController

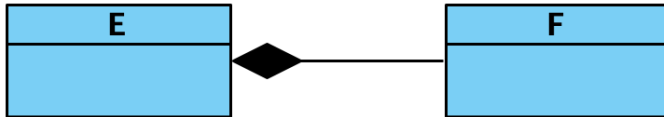
Aggregation (has-a)

Visual Paradigm Standard (lindamarshall@University of Pretoria)



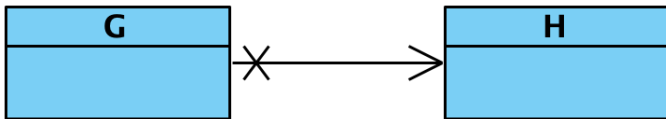
Composition (owns-a)

Visual Paradigm Standard(lindamarshall@University of Pretoria)



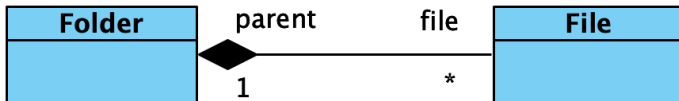
Navigatibility and direction

Visual Paradigm Standard (lindamarshall@University of Pretoria)



Multiplicity and Roles

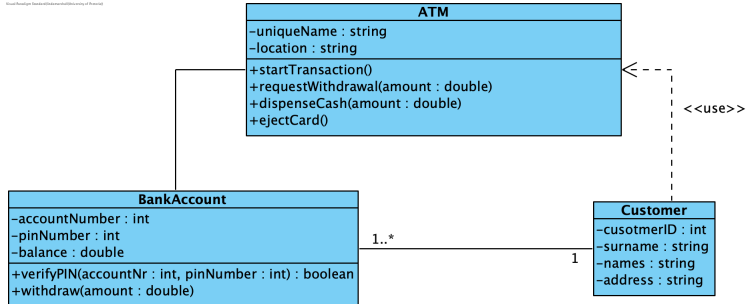
Visual Paradigm Standard (ledamars hall) (University of Pretoria)



Multiplicity cont.

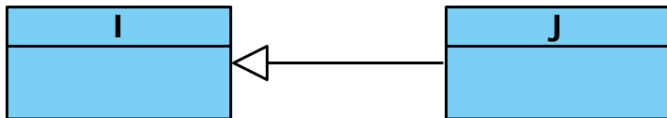
Multiplicity	Meaning
n	Exactly n
$*$	0 or more
$0..n$	0 to n
$0..*$	0 or more
$m..n$	At least m and at most n

Example

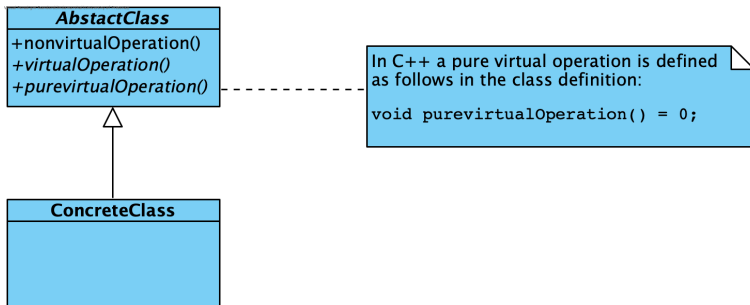


Generalisation is a relationship for modelling **inheritance** and specifically *public* inheritance. It is the manifestation of the *is-a* relationship.

Visual Paradigm Standard(lindamarshall@University of Pretoria)

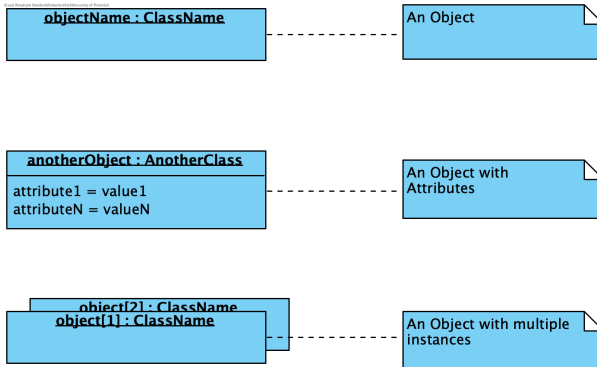


Operations in a class are defined as non-virtual, virtual or pure virtual.

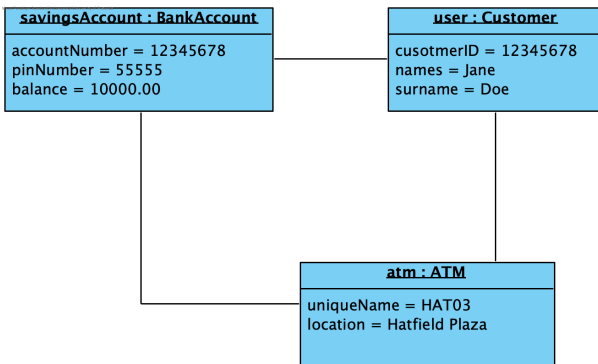


Object diagrams are derived from class diagrams:

- and are therefore dependent on class diagrams
- represent an instance of a class or classes - static view or snapshot of the system at the current moment
- therefore, concrete in nature - represent the real-world versus class diagrams which are abstract and represent the blueprint

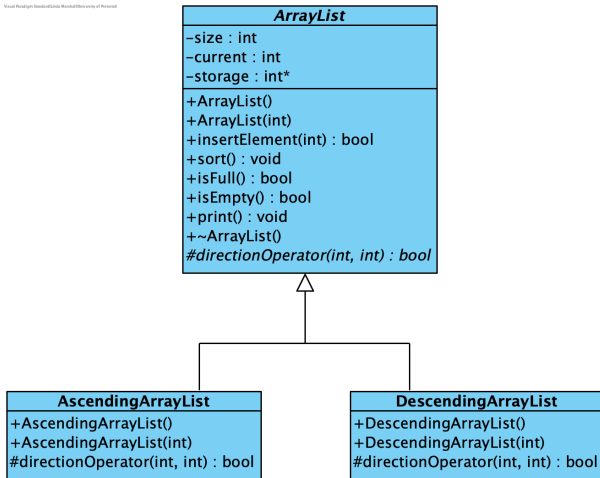


“Name” section is underlined



Use the same basic relationships as class diagrams (associations and delegation)

Visual Paradigm Standard Edition, Marshall University of Portland



arr : DescendingArrayList

```
current = 0  
size = -1  
storage = int*[50]
```

```
DescendingArrayList arr(50)  
// OR  
ArrayList* arr = new DescendingArrayList(50);
```

- Tackling Design patterns, Chapters 1, 2 and 6.

`https://www.cs.up.ac.za/cs/
lmarshall/TDP/TDP.html`

- `http://www.uml-diagrams.org`
- `https://www.youtube.com/watch?
v=UI6lqHOVHic&t=338s`