Design Patterns

UNIVERSITY OF ALBERTA

Course Glossary

We have attempted to identify the most important terms in the course here. Look here if you encounter a term without a definition in context. If you need further explanation, you may have to return to the lesson where we first introduced the term, or do some independent study.

---

abstract superclass - a class which must have subclasses in order to be instantiated.

Adapter pattern - a design pattern that connects two incompatible interfaces by fitting between them and providing a compatible interface to both.

Agile development - a family of software development methodologies that encourage adaptivity, cross-functional collaboration, evolutionary development, early delivery, and continuous improvement.

Anti-Pattern - a commonly-encountered attempted solution to a problem which is counterproductive or ineffective.

behavioural patterns - design patterns that focus on how objects distribute work.

boundary object - an object which interfaces with objects outside the current system or with users.

Chain of Responsibility pattern - a behavioural pattern for allowing multiple entities to handle requests.

code reuse - using existing code to build new software.

code smell - a symptom of bad code.

Command pattern - a behavioural pattern for encapsulating requests as objects.

comments - annotations to code that clarify its purpose. Can be considered a code smell if comments are overused to compensate for a lack of clarity in the code.

Composing objects principle - an alternative method of code reuse that uses aggregation rather than inheritance.

composite class - a class which is designed to contain other classes, including objects of the same type.

Composite pattern - a design pattern for composing nested structures of objects and dealing with these objects uniformly.

concrete instantiation - the act of instantiating a class (usually with the new operator in Java).

control object - generally, an object whose role is to coordinate the actions of other objects.

Controller - coordinates the functions of the model and view components. A control object.

creational patterns - design patterns that involve instantiating concrete objects.

data class - a code smell/anti-pattern; a very small class with little more than data.

data clump - a code smell/anti-pattern; a group of frequently-associated data that would be better encapsulated as an object.

Decorator pattern - a design pattern allowing for added behaviour through wrapping.

design pattern - a general solution to a commonly occurring problem in software design. Not characterized by specific code, but rather, best practices for common issues.

Dependency Inversion principle - a design principle striving for high-level dependencies based on abstract superclasses.

divergent change - a code smell/anti-pattern; developers find themselves changing a class in different, often contradictory ways. Indicates poor separation of concerns.

duplicated code - a code smell/anti-pattern; the same code, sometimes with small changes, is found in multiple places in the software. Indicates that the developer could make better use of generalization.

entity object - an object that encapsulates what the software is interested in representing. Usually provides methods for interacting with the data therein.

Facade pattern - a design pattern that involves encapsulating a complex system in a simple interface.

Factory Method pattern - a pattern that delegates concrete instantiation to a method of a subclass.

Factory Object - any object whose main purpose is concrete instantiation, i.e. actually creating objects

feature envy - a code smell/anti-pattern; two classes that would be better together, usually indicated by frequently invoking each other's methods.

Gang of Four - the four co-authors of the textbook: '*Design Patterns: Elements of Reusable Object-Oriented Software',* a seminal work in object-oriented design and the source of many design patterns.

generalization - a design principle dictating that behaviour should be generalized where possible. General solutions can be applied more widely.

inappropriate intimacy - a code smell/anti-pattern; two classes that should have a further degree of separation.

information hiding - a design principle dictating that information that is not essential outside a class should be hidden from other classes. Sometimes synonymous with encapsulation.

inheritance - when a subtype or subclass inherits behaviours and methods from the supertype or superclass.

Interface Segregation principle - a design principle that recommends having smaller, more specific interfaces so that clients are not dependent on methods that they do not use.

large class - a code smell/anti-pattern; a class is very large. Indicates that there may need to be better separation of concerns. What is considered "large" may vary considerably depending on complexity.

lazy creation/lazy initialization - instantiating an object when it is needed rather than at startup, improving boot time.

leaf class - a class that cannot or should not be subtyped. Can be enforced with the *final* keyword in Java.

long method - a code smell/anti-pattern; a method is very large. Indicates that there may need to be better separation of concerns. What is considered large may vary considerably, e.g. initializing visual elements is typically code-heavy.

long parameter list - a code smell/anti-pattern; a method has a long list of parameters, making the method difficult to use properly. Can sometimes be fixed by passing parameter objects that encapsulate common parameters.

Mediator pattern - a behavioural pattern in which a central object coordinates the activities of many objects.

message chains - a code smell/anti-pattern; a chain of object returns and methods calls is strung together. A violation of the principle of least knowledge.

MVC pattern - stands for Model, View, Controller. A pattern that separates data, display, and control functions into separate roles.

Model - the data or 'backend' component of the MVC pattern. An entity object.

Observer pattern - a behavioural pattern for event handling.

Open/Closed principle - a design principle dictating that an object should be open to extension, but closed to modification.

pattern language - a method of describing good design practices or patterns in a field of expertise.

polymorphism - the ability to interact with objects of different types in the same way. Usually achieved through inheritance or through interfaces in Java.

primitive obsession - a code smell/anti-pattern; inappropriate or extensive use of primitive data types.

principle of least knowledge - a design principle stating that classes should know about and interact with as few other classes as possible. Also called the Law of Demeter. Similar to the "need to know" principle in organizations.

Refactoring - the process of changing code so that external behaviours and interfaces are unchanged but internal structure is improved.

refused bequest - a code smell; subclasses are inheriting methods that they do not need. Could be an indication that the method should be defined in subclasses.

Separation of concerns - a design principle dictating that a program should separate different concerns or functions into separate objects or processes. Allows for easier maintenance and loose coupling.

shotgun surgery - a change in one place that necessitates many other changes; a symptom of tight coupling.

Singleton pattern - a design pattern that enforces a single instantiation of a class that is globally accessible.

speculative generality - a code smell/anti-pattern; defining a superclass or interface before it is needed.

State pattern - a behavioural pattern for handling requests taking into account the current state of the object.

structural patterns - design patterns that describe how objects are connected to one another.

subclass - a class that inherits from a parent class, called a superclass.

subtype - a data type that inherits from a parent type, called a supertype.

switch statements - a code smell/anti-pattern; using switch statements to dictate behaviour where polymorphism is a better solution.

Template Method pattern - a behavioural pattern that allows for similar behaviour to be inherited by multiple subclasses.

View - the display or user-interface component of the MVC pattern. A boundary object or objects because they are responsible for interfacing with the user.