Software Design & Modelling 3 + 4

# What is object orientation?

* **Procedural paradigm:**
* Software is organized around the notion of procedures
* functions or routines
* Procedural abstraction
* Works as long as the data is simple
* Data abstractions
* Records and Structures
* Group together the pieces of data that describe some entity
* Helps reduce the system’s complexity
* **Object-oriented paradigm:**
* Organizing procedural abstractions in the context of data abstractions

# Example of Procedural Paradigms:

A white paper with black text

Description automatically generated

# Object-Oriented Paradigm:

* **An approach in which all computations are performed in the context of objects**
* Objects are instances of classes, which:
* are data abstractions
* contain procedural abstractions that operate on the objects
* A running program can be seen as a collection of objects collaborating to perform a given task

# A view of the two paradigm:

A diagram of a computer program

Description automatically generated

# Objects:

* Object is a chunk of structured data in a running software system
* Has properties
* Represent its state
* Has behaviors
* How it acts and reacts (possibly changing its state)
* May simulate the behavior of an object in the real world

# Example of Objects:

A group of black and white boxes with numbers and letters

Description automatically generated

# Classes:

* **A class is a unit of abstraction in an object oriented (OO) program**
* Represents similar objects
* Its instances
* A kind of software module
* Describes its instances’ structure (properties)
* Contains methods to implement their behavior

# Is something a class or an instance?

* Something should be a class if it could have instances
* Something should be an instance if it is clearly a single member of the set defined by a class

A white background with black text

Description automatically generated

# Instance Variables:

* Variables defined inside a class corresponding to data present in each instance
* Also called fields or member variables
* Attributes
* Simple data
* Associations
* Relationships with other important classes
* e.g. supervisor (also class Employee), coursesTaken (class Course)

# Objects & Instances:

**A screenshot of a computer program

Description automatically generated**

# Class Variables:

A white text with black text

Description automatically generated

# Operations & Methods:

* **Operation**
* A higher-level procedural abstraction (e.g., calculateArea)
* **Method**
* A procedural abstraction used to implement the behavior of a class
* Several different classes can have methods with the same name
* They implement the same abstract operation in ways suitable to each class
* e.g. calculating the area for a rectangle and a circle

# Naming Convention:

A close-up of a computer screen

Description automatically generated

# Member Access Modifiers:

A white background with black text

Description automatically generated

# Inheritances:

A white background with black text

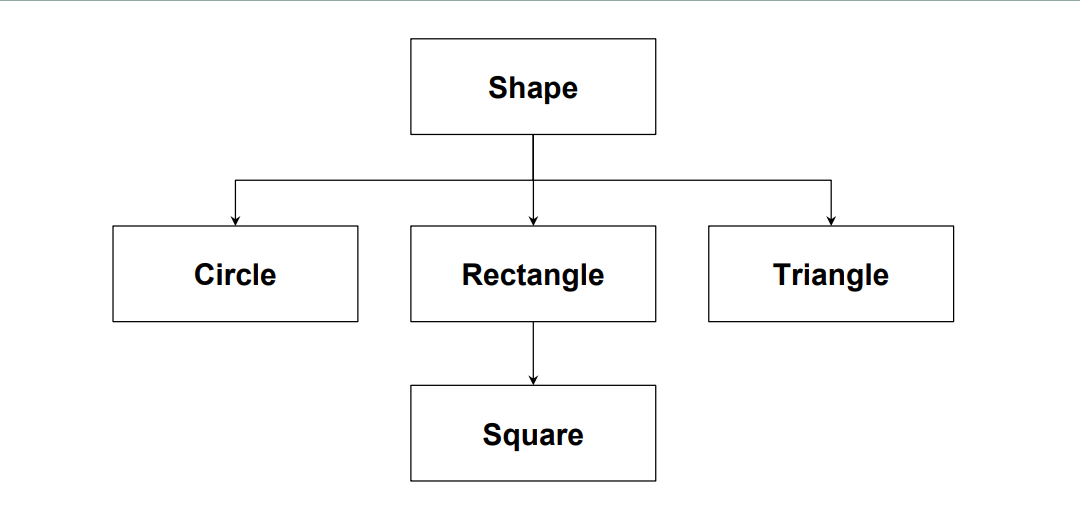
Description automatically generated

# This is a rule:

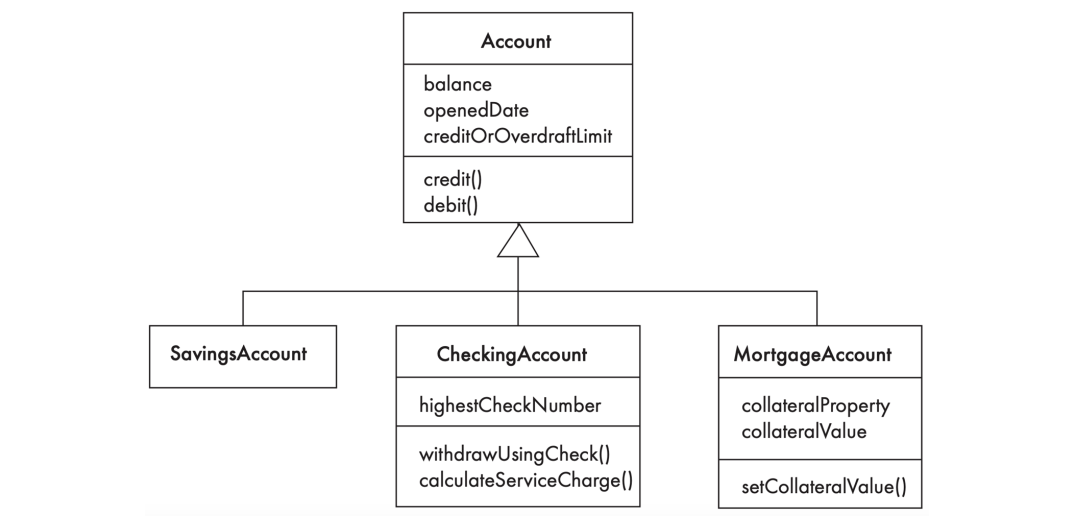
A white text with black text

Description automatically generated

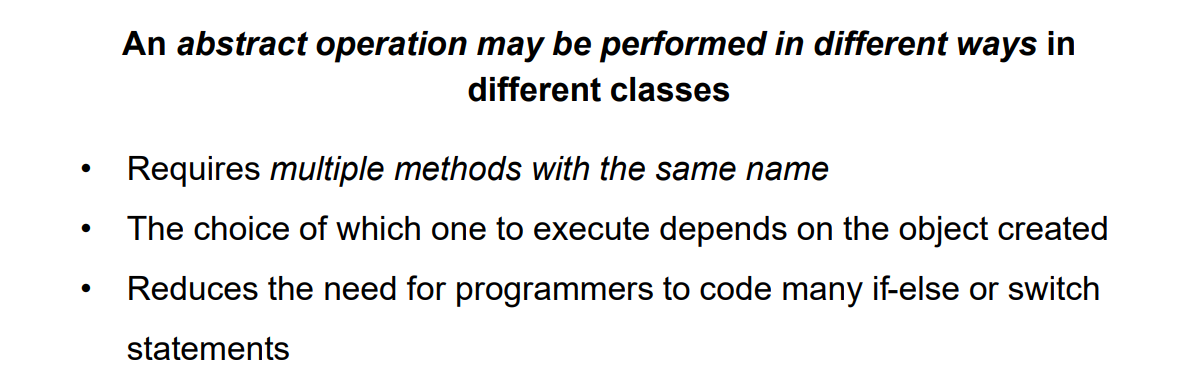
# Example of Inheritance Hierarchy:



# Inherited features must make sense:



# Polymorphism:



# Overriding:

A diagram of a diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

# A hierarchy of class inheritance with polymorphism:

A diagram of a computer

Description automatically generated

# Constructors:

A white text with black text

Description automatically generated

# Abstracts Classes and Methods:

A white background with black text

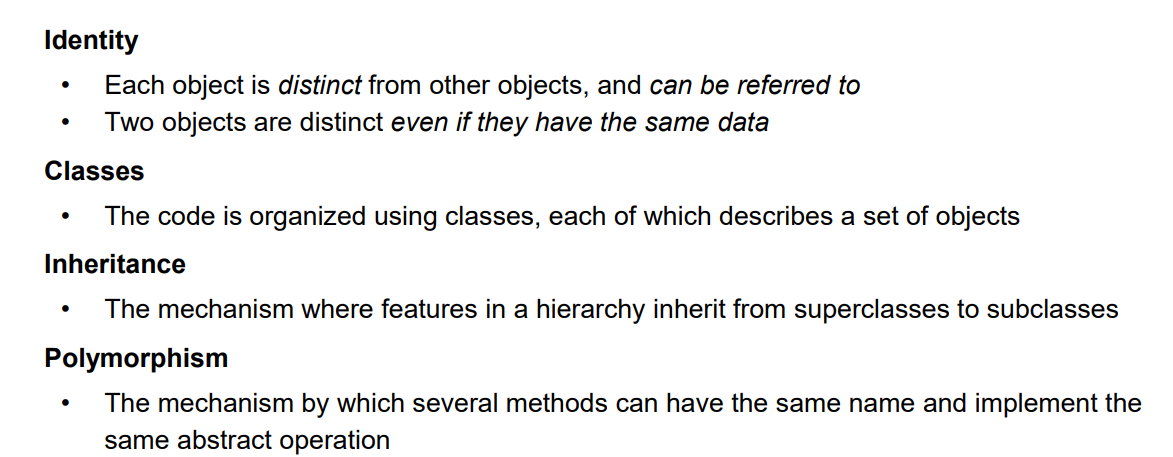
Description automatically generated

# Cohesion and Coupling:

A white background with black text

Description automatically generated

# Concepts that define object orientation:



A white background with black text

Description automatically generated