# Beyond the Textbook (Severance - Chapter 14)

## Creating Custom Python Classes

## Classes and Responsibilities

- Classes are **blueprints** for creating programming objects.
- Programming objects have both **data** and **behavior**.
- Data is carried by the **instance variables** (**fields**).
- Behavior is implemented by **methods**.
- Class responsibilities are divided between data responsibilites: **to know**
- And behavior responsibilities: **to do**

## Data Classes

- When learning to create custom Python classes, we often concentrate on classes that have **high data responsibilities** (instance variables) and **low behavior responsibilites** (methods).
- These are called **data classes**.
- Their purpose is to hold a related set of data facts:
  - a record from a data file
  - a row from a relational database table
  - a row from a spreadsheet
- They make processing data records easier - particularly when sorting.

## Python Dataclasses

- Introduced in Python 3.7 with the `dataclasses` package.
- Python dataclasses use the `@dataclass` **decorator**.
- Basic service methods of the class are generated automatically:
  - `__init__` method
  - `__repr__` method
  - `__eq__` method
- See:
  - *my_states.py*
  - *create_state_area_reports.py*
- Python dataclasses feature even supports class hierarchies.

## Class Hierarchies

- **Superclasses** and **subclasses** can be combined to create a type hierarchy.
- Superclass: `BankAccount`
- Subclasses: `SavingsAccount`, `CheckingAccount`, `TimeDepositAccount`
- Superclass implements common instance variables and methods.
- Subclasses implement additional instance variables and/or methods.
- Subclass methods can override superclass methods.
- Subclass methods can add further functionality to superclass methods.
- Differing behavior by same-named methods in different class is known as **polymorphism**.
- Instances of subclasses know their proper behavior.

## Vehicle Class Hierarchy - Precursor Version

- Vehicle
    - Car
    - Truck
- This is a first version of the class code and the client code.
- This design has too much coupling between class code and client code.
- Each time we introduce a new subtype, we break the client code.
- See:
    - *my_vehicles_precursor.py*
    - *create_vehicle_registration_invoices_precursor.py*

## Vehicle Class Hierarchy - Starter Version

- Vehicle
  - Car
  - Truck
- This is a refactored version of both class code and client code.
- It has much lower coupling between class code and client code.
- New subtypes may be introduced into class code without breaking client code.
- See:
  - *my_vehicles_starter.py*
  - *create_vehicle_registration_invoices_starter.py*

## Vehicle Class Hierarchy - Coding Assignment Version

- Vehicle
  - Car
  - Truck
  - Motorcycle
  - Snowmobile (Challenge)
- These are expansions of the class hierarchy to demonstrate the ease of adding new subtypes in the refactored architecture.

**Last Revised 2022-10-24**