CPSC 250 - Programming for Data Manipulation
Test 3

Instructions: Answer all questions on this test paper. You may not use any devices. Write clearly and show your work where needed.

Part 1 – Multiple Choice (10 points)

Circle the best answer. Each question is worth 2 points.

1. What will happen if a derived class does **not** call `super().__init__()` in its constructor?
   a) The base class is automatically initialized
   b) The base class's constructor is ignored
   c) Python creates a default constructor for it
   d) It has no effect on object creation
2. Which of the following is **true** about overriding methods in a derived class?
   a) You must use the same method name but different parameters
   b) You must redefine all parent methods
   c) The method in the base class is permanently deleted
   d) The derived method replaces the base version when called from an instance
3. In object-oriented programming, what is **polymorphism** most commonly used for?
   a) To allow different object types to respond to the same method call
   b) To copy attributes from parent to child classes
   c) To convert private attributes to public
   d) To compare two unrelated types for equality

4. Consider the following:

```
class A:
    def speak(self):
        return "A speaks"

class B(A):
    def speak(self):
        return "B speaks"

class C(A):
    def speak(self):
        return "C speaks"

class D(B, C):
    pass

print(D().speak())
```

What is printed?

a) A speaks
b) B speaks
c) C speaks
d) Error due to ambiguity

5. What is the main purpose of using **encapsulation** in object-oriented design?

a) To allow multiple classes to inherit the same method
b) To ensure subclasses always override base methods
c) To restrict direct access to some parts of an object
d) To define attributes in the constructor

Part 2 – Find the Errors (10 points)

Each question contains at least two errors related to inheritance or object design. Circle the errors and explain briefly what is wrong.

Question 1:

```
class Vehicle:
    def __init__(self, speed):
        speed = speed

class Bike(Vehicle):
    def __init__(self, speed, gear):
        self.gear = gear
```

Question 2:

```
class Tool:
    def __init__(self, name):
        self.name = name

class Hammer(Tool):
    def __init__(self, name, weight):
        Tool.__init__()
        self.weight = weight
```

Part 3 – Code Writing (30 points)

Answer each of the following questions clearly. Be sure to use correct syntax and indentation.

1. (10 points) – Inheritance with `super()` and `__str__`

Write a class `Book` with attributes `title` and `author`. Then write a subclass `Textbook` that adds a `subject` attribute. Use `super()` to initialize the base class, and override `__str__` to return:

```
"Calculus by Stewart [Subject: Math]"
```

2. (10 points) – `__eq__` and `__lt__`

Create a class `Movie` with attributes `title` and `rating` (a float from 0 to 10).

- Override `__eq__` so two movies are equal if their titles match
- Override `__lt__` so movies can be compared by rating

3. (10 points) – Mixin + Inheritance

Write a mixin class `TimestampMixin` that defines a method `timestamp()` which prints "Logged at some time". Create a base class `Document` with an attribute `filename`. Write a class `PDFDocument` that inherits from both and includes a `print_info()` method that prints the filename and calls `timestamp()`.

Part 4 – Comment the Code (10 points)

Write meaningful comments on the lines marked below. Don't just rewrite the code — explain what each part is doing and why.

```
class Animal:
    def __init__(self, species="unknown"):       # (1)
        self._species = species

    def get_species(self):                        # (2)
        return self._species

    def speak(self):                              # (3)
        return "..."

class Dog(Animal):
    def speak(self):                              # (4)
        return "Woof!"
```