University Computer Science Name: _____ Score: _____

1. Instance attributes come from –

   ☐ copying the class attributes

   ☐ assignments to self, usually in ___init___

   ☐ the caller of the method

   ☐ the class statement itself

2. Calling issue() on a copy that is already out –

   ☐ deletes the copy

   ☐ raises, and the state is unchanged

   ☐ returns False quietly

   ☐ silently issues it again

3. Member.___init___ starts with super().___init___(name) so that –

   ☐ the name is discarded

   ☐ the card number is inherited

   ☐ Person state exists before any Member line runs

   ☐ Member runs before Person

4. A Protocol names –

   ☐ the memory address of the object

   ☐ the test file for the class

   ☐ the behaviour any fitting type must provide

   ☐ the concrete class to inherit

5. except RenewalError catches a raised LimitError because –

   ☐ their messages match

   ☐ except catches everything

   ☐ LimitError is its subclass

   ☐ both are strings

6. A test that asserts the private field loan._days is a good behaviour test.

   ☐ True ☐ False

☐ True ☐ False

8. In Python, a leading underscore stops other code from changing the attribute.

   ☐ True ☐ False

9. Receipt.total() gave 100. The base then gains a loyalty line that takes off 5. DiscountReceipt.total() is written as super().total() − 10. What does it return now?

   _____

10. To be used where a Notifier Protocol is expected, a class must inherit from Notifier.

    ☐ True ☐ False

1. **assignments to self, usually in ___init___**

   ___init___ runs at construction and seeds each instance its own attribute values.

2. **raises, and the state is unchanged**

   The guard raises before any change is made – check-first means failed calls corrupt nothing.

3. **Person state exists before any Member line runs**

   The chain is base-first: skip super() and self._name does not exist yet when Member needs it.

4. **the behaviour any fitting type must provide**

   It is a structural contract: any object with the named methods fits, no inheritance needed.

5. **LimitError is its subclass**

   Handlers match by type through the hierarchy: a parent except catches every child.

6. **False**

   It pins storage, not a promise, so it breaks on refactors. Assert what the caller sees, such as the returned days.

7. **False**

   It stays on the class, and every instance reads that one shared value.

8. **False**

   It is only a convention that says "internal". Nothing stops a caller, so the discipline matters.

9. **85**

   The base now gives 95, and the override subtracts 10 from that. A copied body would still say 90.

10. **False**

    A Protocol is structural. Any class with a matching send method fits, with no inheritance.