Part Two

1. Arrange the classes found in the room into inheritance hierarchies. Generalize to find useful base classes.

We can group the objects in the room into two main base classes: Furniture and ElectronicDevice.

Under Furniture, we have:

- Chair
- Whiteboard

Under ElectronicDevice, we have:

- Laptop
- Keyboard
- Air Conditioner

All of these can be considered subclasses of a more general base class called Object, which represents any physical entity in the room.

2. Ensuring that the principle of substitutability is maintained, **identify where derived classes override members inherited from the base class**. Are there any instances where this principle is violated?

The principle of sustainability or the **Liskov Substitution Principle (LSP)** says: *A subclass should be substitutable for its base class.*

In our hierarchy, ElectronicDevice is a base class, and its subclasses like Laptop, Keyboard, and Air Conditioner can override general behaviors such as “power on” or “shutdown” to perform device-specific actions.

This respects the principle of substitutability as stated above, meaning any object of a subclass (like Laptop) can be used wherever the base class (ElectronicDevice) is expected, without breaking the behavior.

However, if a class like Whiteboard were incorrectly placed under ElectronicDevice (even though it’s not electronic), that would violate substitutability. For example, trying to “power on” a non-electronic whiteboard wouldn’t make sense.

Thus, to avoid violating the principle, Whiteboard should instead belong under Furniture, where it fits more appropriately.

So:

- Substitutability is *maintained when subclasses logically extend the base class and override its behavior in a consistent way.*

- Substitutability is *violated when a subclass doesn't truly fit the role or behavior expected from the base class.*

3. Identify one instance of inheritance polymorphism, and one instance of parametric polymorphism.

- *Inheritance polymorphism:*
    - An example is when both the Laptop and the Air Conditioner are treated as types of ElectronicDevice. You can give the same instruction like “power on” to any electronic device, and each device will respond in its own way. That’s polymorphism through inheritance.
- *Parametric polymorphism:*
    - This happens when a function or method works with different types of inputs. For example, if you had a list of RoomObjects, it could contain Chairs, Laptops, and Whiteboards, and the same method could handle all of them, regardless of their type.

4. Create a multiple-inheritance scenario which results in repeated inheritance.

- For example, let’s say we have a class called Device.
- We also have two other classes: SmartDevice and OfficeEquipment, both of which inherit from Device.
- Then you create a class SmartWhiteboard that inherits from both SmartDevice and OfficeEquipment.

Now, SmartWhiteboard ends up inheriting from Device twice — once through each parent — causing repeated inheritance.

5. State whether the following should be related by inheritance or composition:

PAIR	**RELATIONSHIP TYPE**	**REASON**
a. room, house	Composition	A house contains rooms.
b. boat, fleet	Composition	A fleet is made up of boats.
c. cruiser, boat	Inheritance	A cruiser is a specific type of boat.
d. living room, house	Composition	A house contains a living room.
e. son, child	Inheritance	A son is a specific type of child.

f. noun, word	Inheritance	A noun is a specific type of word.
g. word, sentence	Composition	A sentence is made up of words.
h. child, family	Composition	A family has children, but a child is not a type of family.