

Parent 1 (Example of Multiple Inheritance) **Parent 2**

```

class Simple Calculator {
    void add(a.b); ✓
    void subtract(a.b); ✓
};

class Scientific Calculator {
    void square-root(a); ✓
    void exponent(a.b); ✓
};

class Super Calculator : public Simple Calculator, public Scientific Calculator {
    object → { add(), subtract(), sq-root(), exponent(); }
};
  
```

child

Multilevel Inheritance:

```

class A {
    define method A();
};

class B : public A {
    method B();
};

class C : public B {
    method C();
};

A obj;
obj.method A(); (1)

B obj;
obj.m A(); (2)
obj.m B();

C obj;
obj.m A(), m B(), m C(); (3)
  
```

Polymorphism :→ The process by which a same entity can behave differently under different scenarios or circumstances, is called polymorphism.

Praveen

Classroom → Student
Food Court → Customer
Home → Son

poly → many **morph** → forms or shapes
Role or Character changes.

Two types of polymorphism:

① Static | Compile Time | Overloading (Same Class)

① no of parameters ② data type of parameters.

② Dynamic | Run Time | Overriding (Multiple Classes)

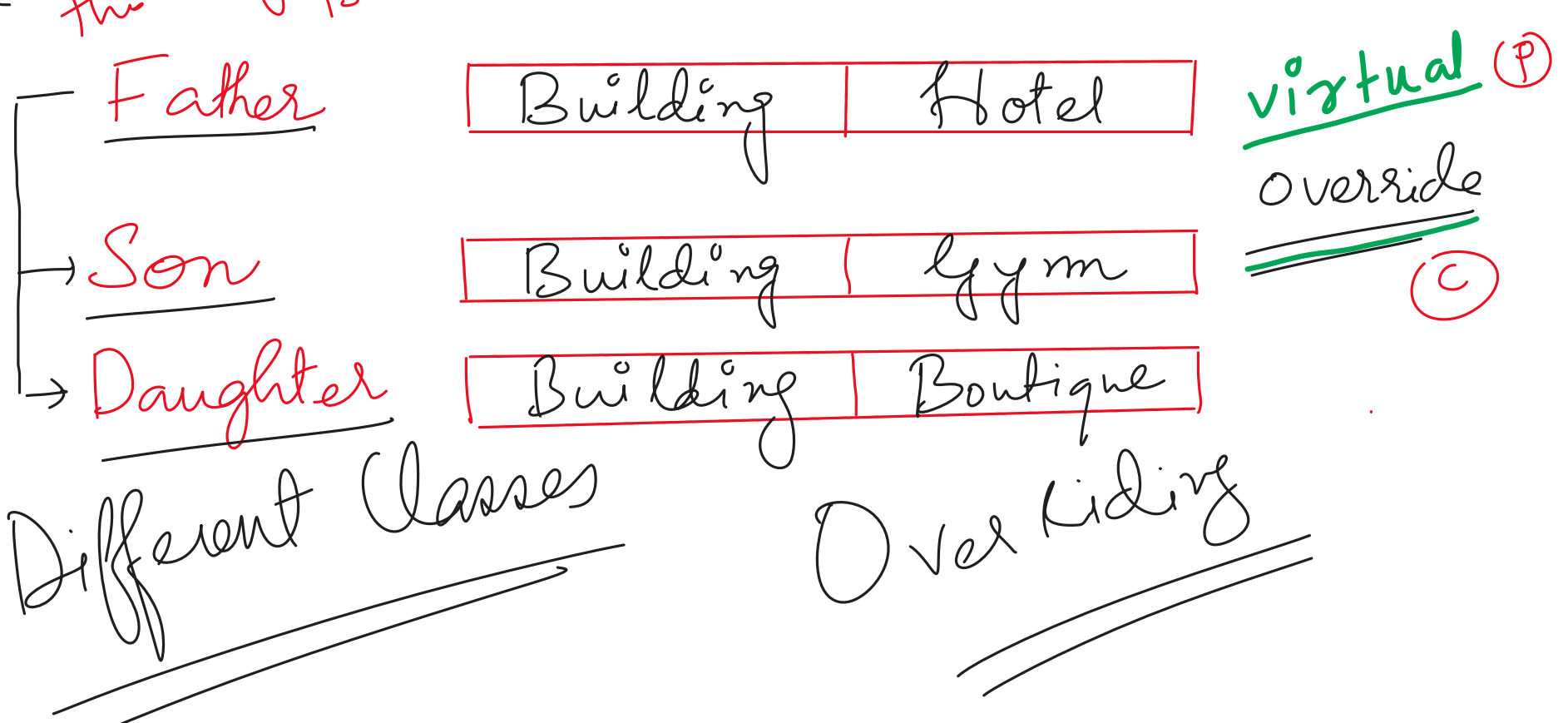
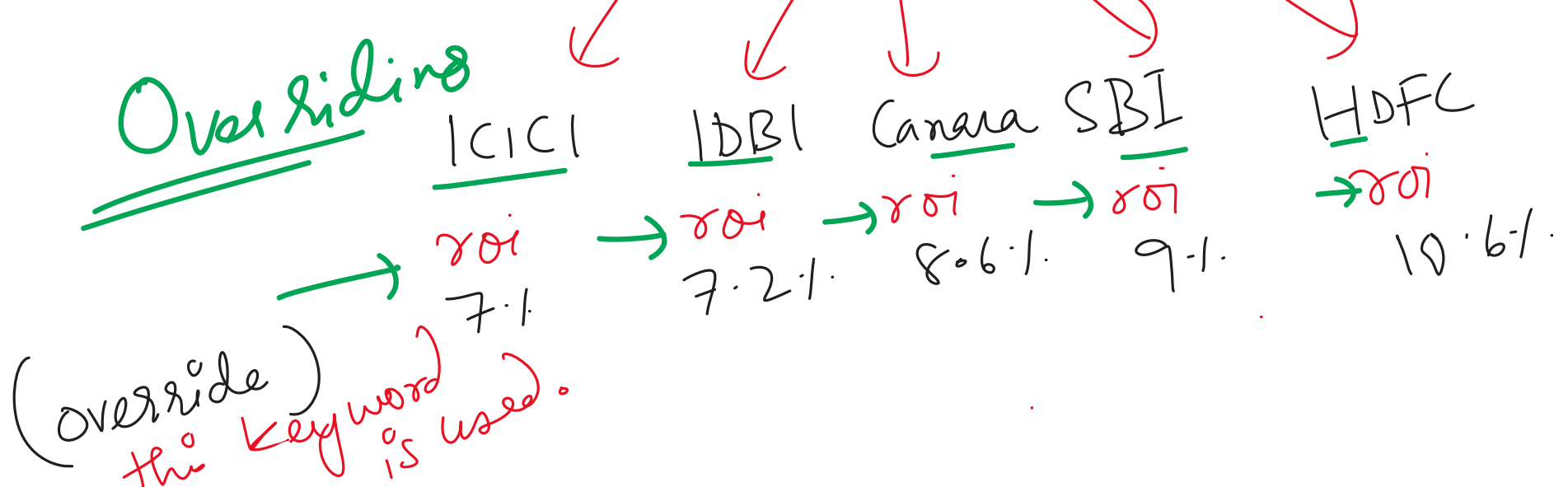
If we don't want to create objects to access the methods, we use the static key word.

```

class A {
    static void show();
};

main() {
    A::show();
};
  
```

Overriding :→



Showing → What is happening (functionality)

Hiding → How it is happening (implementation)

Hiding the implementation details & only showing the functionality to the "end user" is called data abstraction. For better User Exp (UI-UX)

- * Abstraction (pure virtual functions) **
- * Exception Handling (PBC SBC)
- * DMA (CPP (new & delete))
- * Standard Template Library (W3Schools)
- * DSA Roadmap → 20273 File Handling