

May
14/05/2025
Wednesday

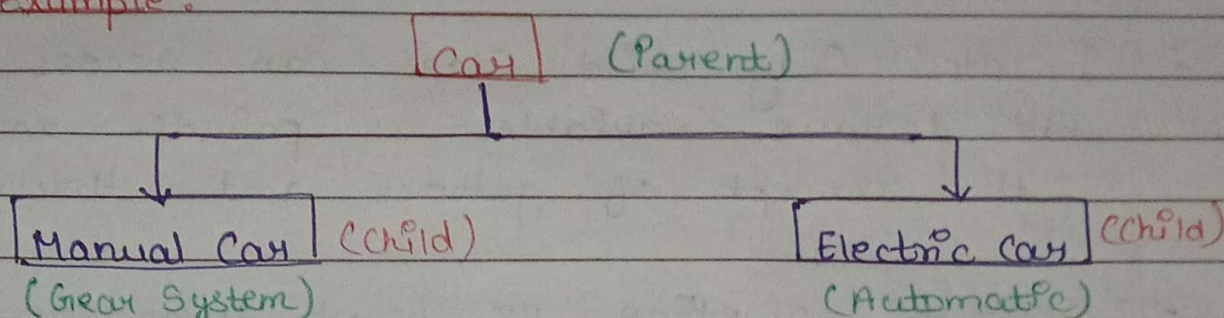
DAY-3 LECTURE-3

Inheritance and Polymorphism

Wake up determined
Go to bed
Satisfied

→ Inheritance : Parent - child Relationship. One object is related with another object

Example :



They both are car and exist in real life and relationship between them is parent-child and both exist in real world so we have represent in real life.

Characteristics

of Car

- brand
- model
- Is Engine On
- Current Speed

- Start engine()
- Stop engine()
- accelerate()
- break()

These all are exist in every car

Manual Car : Carry above features and some extra features.

- Current Gear
- Shift Gear

Electric Car : Specific features

- battery Percentage
- charged Battery

Example:

```

Class Car {
    // Model
    // brand etc
}
    
```

start engine();
stop();

Class Manual Car : public Car
{

// Shift Gear () {}
current Gear = 0;
}

Class Electric Car : public Car
{

// Charged Battery();
// Battery percentage();
}

Inherits all methods of Car
and have its specific methods
like shift Gear();

It can call all the
methods of Car

In both cases we don't need to redefine the
methods like start engine(), stop(). They both
can access it from Car class and they both
have their own specific methods.

Class Manual Car : Public Car

↓
Anyone can
access

or
Private Car

↓
Only class members
and class methods
can access

or
Protected Car

↓
Like Private no one
can access but its child
class can access its
properties

} Access
modifier

Syntax :

Class ManualCar : public Car { ... }

Accessibility Access Modifies Inheritance

Public

- Public member stay public
- Protected members stay protected

Protected

- Public and protected members become protected

Private

- All inherited members become private

Private members of parent class are never inherited



```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  /*
7  We know that real world Objects show inheritance relationship where we
8  have parent object and child object. child object have all the characters
9  or behaviours that parent have plus some additional characters/behaviours.
10 Like all cars in real world have a brand, model etc and can start, stop,
11 accelerate etc. But some specific cars like manual car have gear System
12 while other specific cars like Electric cars have battery system.
13
14 We represent this scenario of real world in programming by creating a parent class and
15 defining all the characters(variables) or behaviours(methods) that all cars
16 have in parent class. Then we create different child classes that inherits
17 from this parent class and define only those characters and behaviours
18 that are specific to them. Although objects of these child classes can
19 access or call parent class characters(variables) and behaviours(methods).
20 Hence providing code reusability.
21 */
22 class Car {
23 protected:
24     string brand;
25     string model;
26     bool isEngineOn;
27     int currentSpeed;
28
29 public:
30     Car(string b, string m) {
31         this->brand = b;
32         this->model = m;
33         isEngineOn = false;
34         currentSpeed = 0;
35     }
36
```

```

36
37 //Common methods for All cars.
38 void startEngine() {
39     isEngineOn = true;
40     cout << brand << " " << model << " : Engine started." << endl;
41 }
42
43 void stopEngine() {
44     isEngineOn = false;
45     currentSpeed = 0;
46     cout << brand << " " << model << " : Engine turned off." << endl;
47 }
48
49 void accelerate() {
50     if (!isEngineOn) {
51         cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
52         return;
53     }
54     currentSpeed += 20;
55     cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h" << endl;
56 }
57
58 void brake() {
59     currentSpeed -= 20;
60     if (currentSpeed < 0) currentSpeed = 0;
61     cout << brand << " " << model << " : Braking! Speed is now " << currentSpeed << " km/h" << endl;
62 }
63
64 virtual ~Car() {}
65 };
66
67 class ManualCar : public Car { // Inherits from Car
68 private:
69     int currentGear; //specific to Manual Car.
70
71 public:
72     ManualCar(string b, string m) : Car(b, m) {
73         currentGear = 0;
74     }
75

```

```

75
76 //Specialized method for Manual Car
77 void shiftGear(int gear) {
78     currentGear = gear;
79     cout << brand << " " << model << " : Shifted to gear " << currentGear << endl;
80 }
81 };
82
83 class ElectricCar : public Car { // Inherits from Car
84 private:
85     int batteryLevel; //specific to Electric Car.
86
87 public:
88     ElectricCar(string b, string m) : Car(b, m) {
89         batteryLevel = 100;
90     }
91
92 //specialized method for Electric Car
93 void chargeBattery() {
94     batteryLevel = 100;
95     cout << brand << " " << model << " : Battery fully charged!" << endl;
96 }
97 };
98
99
100
101 // Main Method
102 int main() {
103
104     ManualCar* myManualCar = new ManualCar("Suzuki", "WagonR");
105     myManualCar->startEngine();
106     myManualCar->shiftGear(1); //specific to manual car
107     myManualCar->accelerate();
108     myManualCar->brake();
109     myManualCar->stopEngine();
110     delete myManualCar;
111
112     cout << "-----" << endl;
113
114     ElectricCar* myElectricCar = new ElectricCar("Tesla", "Model S");
115     myElectricCar->chargeBattery(); //specific to electric car
116     myElectricCar->startEngine();
117     myElectricCar->accelerate();
118     myElectricCar->brake();
119     myElectricCar->stopEngine();
120     delete myElectricCar;
121
122     return 0;
123 }

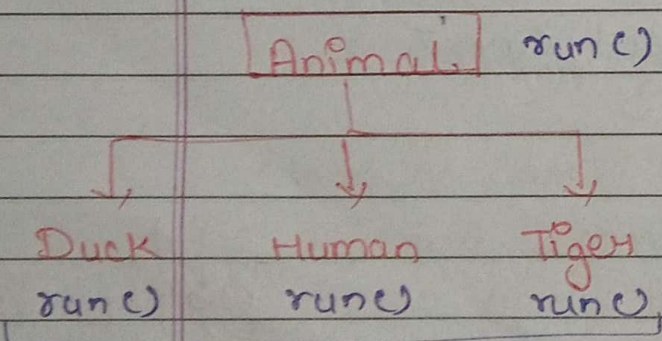
```


→ Polymorphism :

Poly → Many
Morphism → form

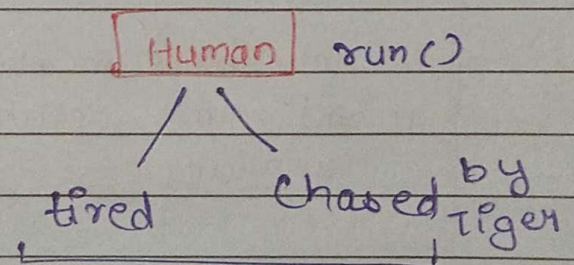
Some objects
have many form

Scenario - 1



Each performs run() differently

Scenario - 2



In both same human perform run differently

Polymorphism means same object perform differently based on situation

Polymorphism

Dynamic Polymorphism

→ Method Overriding

→ Example Scenario-1

Animal can be duck, human, tiger and they ~~have~~ ^{perform} different reaction on ~~different~~ ^{run}

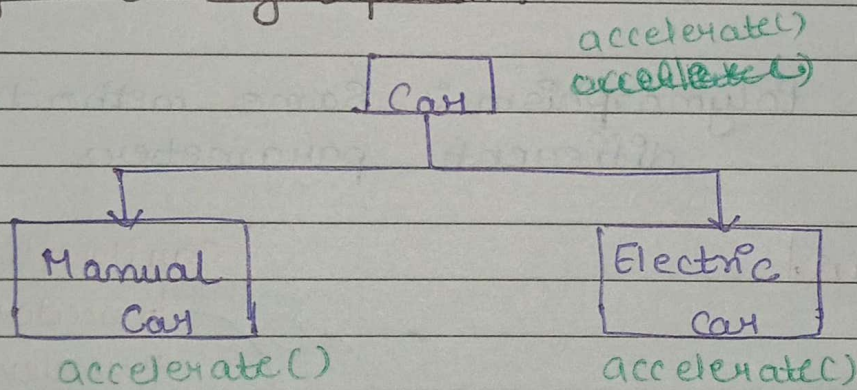
Static Polymorphism

→ Method Overloading

→ Example Scenario-2

One object human and one ~~parameter~~ ^{method} run(), if we change situation or parameter then it perform different

Dynamic Polymorphism:



- Car has method accelerate but
- Manual Car perform accelerate differently than the electric car.
- Manual Car has gear so it perform accelerate speedily or in electric car may be it perform accelerate slow.
- Both use method accelerate but in their own way

Class car {

virtual accelerate () = 0;

→ Abstract Class

}

← We only
define not
declare it so
it become
abstract
class



Manual car

have to



Electric car

They both declare
accelerate method

accelerate()

{

}

accelerate()

{

}

When we make object of manual car
and electric car they will give different
output if we call accelerate function

```

1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  /*
7   Dynamic Polymorphism in real life says that 2 Objects coming from same
8   family will respond to same stimulus differently. Like in real world Manual
9   car and Electric car will respond to accelerate() differently.
10
11  To represent this in programming, we create a parent class that defines all
12  characters and behaviours that are generic to all child classes and are also same in
13  all child classes but make those methods abstract(virtual) that are generic to all
14  child classes but all child class will behave differently. Then those child class
15  will provide implementation details of these abstract methods the way they want.
16  */
17  class Car {
18  protected:
19      string brand;
20      string model;
21      bool isEngineOn;
22      int currentSpeed;
23
24  public:
25      Car(string brand, string model) {
26          this->brand = brand;
27          this->model = model;
28          this->isEngineOn = false;
29          this->currentSpeed = 0;
30      }
31
32      //Common methods for All cars.
33      void startEngine() {
34          isEngineOn = true;
35          cout << brand << " " << model << " : Engine started." << endl;
36      }
37

```



```

37
38     void stopEngine() {
39         isEngineOn = false;
40         currentSpeed = 0;
41         cout << brand << " " << model << " : Engine turned off." << endl;
42     }
43
44     virtual void accelerate() = 0; // Abstract method for Dynamic Polymorphism
45     virtual void brake() = 0;     // Abstract method for Dynamic Polymorphism
46     virtual ~Car() {}             // Virtual destructor
47 };
48
49 class ManualCar : public Car {
50 private:
51     int currentGear;
52
53 public:
54     ManualCar(string brand, string model) : Car(brand, model) {
55         this->currentGear = 0;
56     }
57
58     //Specialized method for Manual Car
59     void shiftGear(int gear) {
60         currentGear = gear;
61         cout << brand << " " << model << " : Shifted to gear " << currentGear << endl;
62     }
63
64     // Overriding accelerate - Dynamic Polymorphism
65     void accelerate() {
66         if (!isEngineOn) {
67             cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
68             return;
69         }
70         currentSpeed += 20;
71         cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h" << endl;
72     }
73

```

```

73
74 // Overriding brake - Dynamic Polymorphism
75 void brake() {
76     currentSpeed -= 20;
77     if (currentSpeed < 0) currentSpeed = 0;
78     cout << brand << " " << model << " : Braking! Speed is now " << currentSpeed << " km/h" << endl;
79 }
80 };
81
82 class ElectricCar : public Car {
83 private:
84     int batteryLevel;
85
86 public:
87     ElectricCar(string brand, string model) : Car(brand, model) {
88         this->batteryLevel = 100;
89     }
90
91 //specialized method for Electric Car
92 void chargeBattery() {
93     batteryLevel = 100;
94     cout << brand << " " << model << " : Battery fully charged!" << endl;
95 }
96
97 // Overriding accelerate - Dynamic Polymorphism
98 void accelerate() {
99     if (!isEngineOn) {
100         cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
101         return;
102     }
103     if (batteryLevel <= 0) {
104         cout << brand << " " << model << " : Battery dead! Cannot accelerate." << endl;
105         return;
106     }
107     batteryLevel -= 10;
108     currentSpeed += 15;
109     cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h. Battery at "
110     << batteryLevel << "%." << endl;
111 }
112

```



```

111     }
112
113     // Overriding brake - Dynamic Polymorphism
114     void brake() {
115         currentSpeed -= 15;
116         if (currentSpeed < 0) currentSpeed = 0;
117         cout << brand << " " << model << " : Regenerative braking! Speed is now " << currentSpeed << " km/h. Battery at " << batteryLevel << "%" << endl;
118     }
119 };
120
121
122 // Main function
123 int main() {
124     Car* myManualCar = new ManualCar("Suzuki", "WagonR");
125     myManualCar->startEngine();
126     myManualCar->accelerate();
127     myManualCar->accelerate();
128     myManualCar->brake();
129     myManualCar->stopEngine();
130
131     cout << "-----" << endl;
132
133     Car* myElectricCar = new ElectricCar("Tesla", "Model S");
134     myElectricCar->startEngine();
135     myElectricCar->accelerate();
136     myElectricCar->accelerate();
137     myElectricCar->brake();
138     myElectricCar->stopEngine();
139
140     // Cleanup
141     delete myManualCar;
142     delete myElectricCar;
143
144     return 0;
145 }

```

Static Polymorphism: Same methods
different parameters

Human

↓
run()

↳ IsInDanger()

↳ True

↳ False

Human has one method run

we pass (IsInDanger) as argument

if it is True then it will run

fast if it is false then it run
slow.

Car Model

↳ accelerate()

↳ accelerate(int
speed)

↳ Default

how much
we push paddle

then accelerate
according to that

Both are same
method but
parameter is different



```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  /*
7   Static Polymorphism (Compile-time polymorphism) in real life says that
8   the same action can behave differently depending on the input parameters.
9   For example, a Manual car can accelerate by a fixed amount or by a
10  specific amount you request. In programming, we achieve this via method
11  overloading: multiple methods with the same name but different signatures.
12  */
13
14  class ManualCar {
15
16  private:
17      string brand;
18      string model;
19      bool isEngineOn;
20      int currentSpeed;
21      int currentGear;
22
23  public:
24      ManualCar(string brand, string model) {
25          this->brand = brand;
26          this->model = model;
27          this->isEngineOn = false;
28          this->currentSpeed = 0;
29          this->currentGear = 0;
30      }
31
```

```

31
32 void startEngine() {
33     isEngineOn = true;
34     cout << brand << " " << model << " : Engine started." << endl;
35 }
36
37 void stopEngine() {
38     isEngineOn = false;
39     currentSpeed = 0;
40     cout << brand << " " << model << " : Engine turned off." << endl;
41 }
42
43 // Overloading accelerate - Static Polymorphism
44 void accelerate() {
45     if (!isEngineOn) {
46         cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
47         return;
48     }
49     currentSpeed += 20;
50     cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h" << endl;
51 }
52
53 void accelerate(int speed) {
54     if (!isEngineOn) {
55         cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
56         return;
57     }
58     currentSpeed += speed;
59     cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h" << endl;
60 }
61
62 void brake() {
63     currentSpeed -= 20;
64     if (currentSpeed < 0) currentSpeed = 0;
65     cout << brand << " " << model << " : Braking! Speed is now " << currentSpeed << " km/h" << endl;
66 }
67

```



```
61
62     void brake() {
63         currentSpeed -= 20;
64         if (currentSpeed < 0) currentSpeed = 0;
65         cout << brand << " " << model << " : Braking! Speed is now " << currentSpeed << " km/h" << endl;
66     }
67
68     void shiftGear(int gear) {
69         currentGear = gear;
70         cout << brand << " " << model << " : Shifted to gear " << currentGear << endl;
71     }
72 };
73
74 // Main function
75 int main() {
76     ManualCar* myManualCar = new ManualCar("Suzuki", "WagonR");
77     myManualCar->startEngine();
78     myManualCar->accelerate();
79     myManualCar->accelerate(40);
80     myManualCar->brake();
81     myManualCar->stopEngine();
82
83     // Cleanup
84     delete myManualCar;
85
86     return 0;
87 }
```



```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  // Base Car class
7  class Car {
8  protected:
9      string brand;
10     string model;
11     bool isEngineOn;
12     int currentSpeed;
13
14 public:
15     Car(string brand, string model) {
16         this->brand = brand;
17         this->model = model;
18         this->isEngineOn = false;
19         this->currentSpeed = 0;
20     }
21
22     //Common methods for All cars.
23     void startEngine() {
24         isEngineOn = true;
25         cout << brand << " " << model << " : Engine started." << endl;
26     }
27
28     void stopEngine() {
29         isEngineOn = false;
30         currentSpeed = 0;
31         cout << brand << " " << model << " : Engine turned off." << endl;
32     }
33 }
```



```

32     }
33
34     virtual void accelerate() = 0; // Abstract method for Dynamic Polymorphism
35
36     virtual void accelerate(int speed) = 0; //Abstract method for Static Polymorphism
37
38     virtual void brake() = 0; // Abstract method for Dynamic Polymorphism
39
40     virtual ~Car() {} // Virtual destructor
41 };
42
43 class ManualCar : public Car {
44 private:
45     int currentGear;
46
47 public:
48     ManualCar(string brand, string model) : Car(brand, model) {
49         this->currentGear = 0;
50     }
51
52     //Specialized method for Manual Car
53     void shiftGear(int gear) {
54         currentGear = gear;
55         cout << brand << " " << model << " : Shifted to gear " << currentGear << endl;
56     }
57
58     // Overriding accelerate - Dynamic Polymorphism
59     void accelerate() {
60         if (!isEngineOn) {
61             cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
62             return;
63         }
64         currentSpeed += 20;
65         cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h" << endl;
66     }
67

```

```

67
68 //overriding and overloading accelerate at the same time.
69 void accelerate(int speed) {
70     if (!isEngineOn) {
71         cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
72         return;
73     }
74     currentSpeed += speed;
75     cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h" << endl;
76 }
77
78 // Overriding brake - Dynamic Polymorphism
79 void brake() {
80     currentSpeed -= 20;
81     if (currentSpeed < 0) currentSpeed = 0;
82     cout << brand << " " << model << " : Braking! Speed is now " << currentSpeed << " km/h" << endl;
83 }
84 };
85
86 class ElectricCar : public Car {
87 private:
88     int batteryLevel;
89
90 public:
91     ElectricCar(string brand, string model) : Car(brand, model) {
92         this->batteryLevel = 100;
93     }
94
95     //specialized method for Electric Car
96     void chargeBattery() {
97         batteryLevel = 100;
98         cout << brand << " " << model << " : Battery fully charged!" << endl;
99     }
100
101 // Overriding accelerate - Dynamic Polymorphism
102 void accelerate() {
103     if (!isEngineOn) {
104         cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
105         return;

```



```

106     }
107     if (batteryLevel <= 0) {
108         cout << brand << " " << model << " : Battery dead! Cannot accelerate." << endl;
109         return;
110     }
111     batteryLevel -= 10;
112     currentSpeed += 15;
113     cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h. Battery at " << batteryLevel << "%" << endl;
114 }
115
116 // Overriding accelerate - Dynamic Polymorphism
117 void accelerate(int speed) {
118     if (!isEngineOn) {
119         cout << brand << " " << model << " : Cannot accelerate! Engine is off." << endl;
120         return;
121     }
122     if (batteryLevel <= 0) {
123         cout << brand << " " << model << " : Battery dead! Cannot accelerate." << endl;
124         return;
125     }
126     batteryLevel -= 10 + speed;
127     currentSpeed += speed;
128     cout << brand << " " << model << " : Accelerating to " << currentSpeed << " km/h. Battery at " << batteryLevel << "%" << endl;
129 }
130
131 // Overriding brake - Dynamic Polymorphism
132 void brake() {
133     currentSpeed -= 15;
134     if (currentSpeed < 0) currentSpeed = 0;
135     cout << brand << " " << model << " : Regenerative braking! Speed is now " << currentSpeed << " km/h. Battery at " << batteryLevel << "%" << endl;

```

```

136     }
137 };
138
139 // Main function
140 int main() {
141     Car* myManualCar = new ManualCar("Ford", "Mustang");
142     myManualCar->startEngine();
143     myManualCar->accelerate();
144     myManualCar->accelerate();
145     myManualCar->brake();
146     myManualCar->stopEngine();
147
148     cout << "-----" << endl;
149
150     Car* myElectricCar = new ElectricCar("Tesla", "Model S");
151     myElectricCar->startEngine();
152     myElectricCar->accelerate();
153     myElectricCar->accelerate();
154     myElectricCar->brake();
155     myElectricCar->stopEngine();
156
157     // Cleanup
158     delete myManualCar;
159     delete myElectricCar;
160
161     return 0;
162 }

```