**Program 1:-**

class Test {

  int x = 10;

  Test(this.x);

}

class Test2 extends Test {

  Test2(super.x);

}

void main() {

  Test2 obj = Test2(10);

  Test obj2 = Test(30);

  obj.x = 19;

  print(obj.x);

  print(obj2.x);

}

**O/P:- 19**

**30**

**Explanation:-** after the passing the values its overrides and print the values

**Program 2:-**

class Test {

  int x = 30;

}

class Test2 extends Test {

  int x;

  Test2(this.x);

  void gun() {

    this.x = 8;

  }

  void fun() {

    print(this.x);

    print(super.x);

  }

}

void main() {

  Test2 obj = Test2(10);

  obj.gun();

  obj.fun();

}

**o/p:- 8**

**30**

**Exited**

**Explanation:-** after passing the value thorugh the obj it initializes the x with 10,but after calling obj.x it overrides the x with values 8 and after calling gun it prints the x and the super calls the parent where the again x is override with 30 then it prints 30

**Program3:-**

class Test {

  int x = 30;

  int y = 30;

}

class Test2 extends Test {

  int x;

  Test2(this.x);

  void gun() {

    this.x = 8;

    this.x = 19;

  }

  void fun() {

    print(super.x);

    print(super.y);

  }

}

void main() {

  Test2 obj = Test2(10);

  obj.gun();

  obj.fun();

}

**O/P:-**

30

30

**Explanation:-** after calling fun it call the parent class using the super keyword where the values of x is 30 and y is 30

**Program4:-**

class Test {

  int x;

  int? y;

  Test({required this.x, this.y});

}

class Test2 extends Test {

  Test2(int x, int y) : super(x: 88) {

    print(x);

  }

  void fun() {

    print(this.x);

    print(super.x);

  }

}

void main() {

  Test2 obj = Test2(19, 20);

  obj.fun();

}

**O/P:-**

19

88

88

**Explanation:-** after passing values through obj to the constructor x is printed as 19 and after that super is called which overrides the x with 88 and after calling fun it again print the x

**Program 5:-**

class Test {

  int x = 20;

  String str = "Core2web";

  void parentMethod() {

    print(x);

    print(str);

  }

}

class Test2 extends Test {

  int x = 10;

  String str = "Incubator";

  void childMethod() {

    print(x);

    print(str);

  }

}

void main() {

  Test2 obj = new Test2();

  obj.parentMethod();

  obj.childMethod();

}

**O/P:-**

**10**

**Incubator**

**10**

**Incubator**

**Explanation:-** after calling the method it prints the values which are in child class due to inheritance

**Program 6:-**

class Test {

  int? x;

  Test(this.x) {

    print("in test");

  }

}

class Test2 extends Test {

  int? y;

  Test2(this.y, int x) : super(x);

}

class Test3 extends Test2 {

  int? z;

  Test3(this.z, int y, int x) : super(y, x) {

    print("in test3");

  }

}

void main() {

  Test3 obj = Test3(10, 20, 30);

}

**O/P:-**

**in test**

**in test3**

**Explanation:-** after passing the values through objects it calls the parent constructor using the super that’s why first print in test and aftrt that in test3

**Program7:-**

class Demo1 {

  int x;

  Demo(this.x);

  }

class Demo2 extends Demo1 {

  Demo2(super.x);

  void fun(){

    print(x);

  }

}

void main() {

 // Demo obj1 = Demo2()

  Demo1 obj2 = Demo2(10);

  obj2.fun();

}

**Error: The non-abstract class 'Demo1' is missing implementations for these members: - Demo1.Demo Try to either - provide an implementation, - inherit an implementation from a superclass or mixin, - mark the class as abstract, or - provide a 'noSuchMethod' implementation. class Demo1 { ^^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart:3:3: Context: 'Demo1.Demo' is defined here. Demo(this.x); ^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart:7:7: Error: The non-abstract class 'Demo2' is missing implementations for these members: - Demo1.Demo Try to either - provide an implementation, - inherit an implementation from a superclass or mixin, - mark the class as abstract, or - provide a 'noSuchMethod' implementation. class Demo2 extends Demo1 { ^^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart:3:3: Context: 'Demo1.Demo' is defined here. Demo(this.x); ^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart:3:8: Error: Field formal parameters can only be used in a constructor. Try removing 'this.'. Demo(this.x); ^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart:9:3: Error: The superclass, 'Demo1', has no unnamed constructor that takes no arguments. Demo2(super.x); ^^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart:18:8: Error: The method 'fun' isn't defined for the class 'Demo1'. - 'Demo1' is from '../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart'. Try correcting the name to the name of an existing method, or defining a method named 'fun'. obj2.fun(); ^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart7.dart:2:7: Error: Field 'x' should be initialized because its type 'int' doesn't allow null. int x;**

**Explanation :-** x is not been initialized in the whole code that’s why its giving error

**Program 8:-**

class Parent {

  Parent() {

    print("in parent constructor");

  }

}

class Child extends Parent {

  Child() {

    super();

    print("in child constructor");

  }

}

void main() {

  Child obj = new Child();

}

**Error: Superclass has no method named 'call'. super(); ^^^^**

**Explanation:-** parent class must have call to use super

**Program 9:-**

class Test {

  int? x;

  static int y=20;

  Test.initX(this.x)

    static void changeY(){

           y=30;

    }

}

class Test2 extends Test {

  Test2(int x) : super.initX(x);

}

void main() {

  Test2 obj = Test2(40);

  Test2.changeY();

  print(Test2.y);

}

**Error: Expected '{' before this. static void changeY(){ ^^^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart9.dart:15:9: Error: Member not found: 'Test2.changeY'. Test2.changeY(); ^^^^^^^ ../../../../Desktop/DART/6-1-24/Inheritance/dart9.dart:16:15: Error: Member not found: 'y'. print(Test2.y);**

**Expalnation:-wrong syntax of super constructor**

**Program 10:-**

abstract class Demo1 {

  factory Demo1() {

    return Demo2();

  }

}

class Demo2 implements Demo1 {

  Demo2() {

    print("Demo2");

  }

}

void main() {

  Demo1 obj = new Demo1();

}

**o/p:-Demo2**

**Explanation:-after creating the obj the factory constructor has been called which returns the demo2 constructor**