Mixin: A mixin is a way to reuse code (methods and properties) in multiple classes without using inheritance. It allows a class to "mix in" behavior from one or more mixins, helping you share functionality between unrelated classes.

Key points:

- Mixins are defined using the mixin keyword.
- They cannot have constructors.
- You apply mixins to classes using the with keyword.
- Mixins help avoid code duplication and enable composition.

Ex: 1) mixin Logger

```
{
  void log(String msg)
  {
    print('Log: $msg');
  }
}
```

class Service with Logger

```
{
  void fetchData()
  {
    log('Fetching data...');
  }
}
```

void main()

```
{
  Service s = Service();
  s.fetchData();  // Log: Fetching data...
}
```

2) abstract class Performer

```
{
  void perform()
```

```
    {
      print("hello");
    }
  }
 abstract class Hero
{
  void action();
}
 mixin Dancer on Performer
 {
  @override
  void perform()
  {
     print("dance");
  }
}
mixin Singer on Hero
{
  void perform()
 {
    print("sing");
  }
  @override
  void action()
 {
    print("Acting");
  }
}
```

```
class Actor extends Performer with Dancer
{
  void dis()
  {
    perform();
  }
}
void main()
{
  Actor a = Actor();
  a.perform(); // dance
  //a.action();
}
```

static variable: Shared by all instances of a class. Belongs to the class itself, not to any object.

static method: Can be called without creating an object of the class. It also can't access non-static members directly.

Ex:

```
class Counter
    {
        static int count = 0;


       static void increment()
       {
          count++;
          print("Count is now: $count");
        }
```

```
}
void main()
 {
     Counter.increment();
     Counter.increment();
     print("Final count: ${Counter.count}"); // Count is now: 1
                                                    Count is now: 2
                                                    Final count: 2
  }
```