```
@FunctionalInterface
interface Supplier<T> {
    T get();
}

public class Car {
    // Supplier是jdk1.8的接口，这里和lambda一起使用了
    // 函数式接口Supplier：无参数，返回一个结果。
    public static Car create(final Supplier<Car> supplier) {
        return supplier.get();
    }

    public Car(){}

    public static void collide(final Car car) {
        System.out.println("Collided " + car.toString());
    }

    public void follow(final Car another) {
        System.out.println("Following the " + another.toString());
    }

    public void repair() {
        System.out.println("Repaired " + this.toString());
    }

    @Override
    public int hashCode() {
        return super.hashCode();
    }
}

import java.util.Arrays;
import java.util.List;

public class MethodReference implements Hello, Vehicle{

    public static void main(String[] args) {
        // 构造器引用
        final Car car = Car.create(Car::new);
        Car car1 = new Car();
        System.out.println(car);
        final List<Car> cars = Arrays.asList(car,car1);
        // 静态方法调用
        cars.forEach(Car::collide);
```

```
        cars.forEach(Car::repair);
        cars.forEach(Car.create(Car::new)::follow);
        //new MethodReference().desc();
    }

    // 本来实现的接口中的默认方法不需要重写，但是这两个接口中有相同默认方法，必须
重写
    public void desc() {
        Vehicle.super.desc();
    }
}


interface Hello{
    // public String print();
    static void test(){
        System.out.println("interface static method");
    }

    default void desc(){
        System.out.println("我是一辆车");
    }
}

interface Vehicle{
    default void desc(){
        System.out.println("我是一辆交通工具");
    }
}
```