【案例12-2】：速度计算

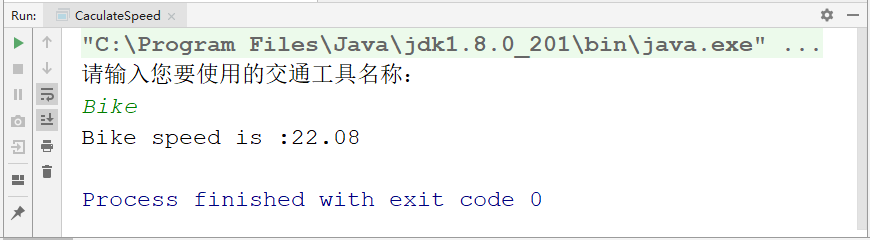
**【案例介绍】**

**1.案例描述**

本案例要求使用反射技术编写一个速度计算程序，计算某种交通工具的行驶速度。现有两种工具：Bike和 Plane，其中Bike的速度运算公式为：A\*B/C，Plane的速度运算公式为：A+B+C。

用户可通过输入交通工具名称选择自己想要使用的交通工具，选择交通工具之后，自动计算出该交通工具的行驶速度。此外，在未来如果增加第3种交通工具的时候，不必修改以前的任何程序，只需要编写新的交通工具的程序即可。

**2.运行结果**



运行结果

**【案例目标】**

* 学会分析“速度计算程序设计”实现的逻辑思路。
* 能够独立完成“速度计算程序设计”的源代码编写、编译及运行。
* 掌握反射机制的应用。

**【案例分析】**

（1）通过任务描述可知，有两种交通工具Plane和Bike：Plane类、Bike类。

（2）由于任务要求在未来如果增加第3种交通工具的时候，不必修改以前的任何程序，只需要编写新的交通工具的程序，故还需要编写一个接口Common，且Plane类和Bike类都继承Common接口。

（3）最后编写一个测试类CaculateSpeed，在main()方法中，编写程序，提示用户输入自己想要使用的交通工具，并利用反射来计算交通工具的速度。

**【案例实现】**

Common.java

1. public interface Common {
2. double getSpeed(double a,double b,double c);
3. }

上述代码中，定义了一个Common接口，在Common接口中定义了一个抽象方法getSpeed()。

Bike.java

1. public class Bike implements Common {
2. @Override
3. public double getSpeed(double a, double b, double c) {
4. return a\*b/c;
5. }
6. }

上述代码中，定义了一个Bike类并实现了Common接口，在Bike类中对getSpeed()方法方法进行了重写。

Plane.java

1. public class Plane implements Common {
2. @Override
3. public double getSpeed(double a, double b, double c) {
4. return a+b+c;
5. }
6. }

上述代码中，定义了一个Plane类并实现了Common接口，在Plane类中对getSpeed()方法进行了重写。

CaculateSpeed.java

1. import java.util.Scanner;
2. public class CaculateSpeed {
3. public static void main(String[] args){
4. Scanner in = new Scanner(System.in);
5. System.out.println("请输入您要使用的交通工具名称：");
6. String choice =in.nextLine();
7. String transport = "fanshe."+choice; //
8. double a = 23, b = 24, c = 25;
9. try {
10. Common newtransport = (Common)
11. Class.forName(transport).newInstance();
12. System.out.println(choice+" speed is :
13. "+newtransport.getSpeed(a,b,c));
14. } catch (InstantiationException e) {
15. e.printStackTrace();
16. } catch (IllegalAccessException e) {
17. e.printStackTrace();
18. } catch (ClassNotFoundException e) {
19. e.printStackTrace();
20. }
21. }
22. }

上述代码中，定义了一个测试类CaculateSpeed，其中第7行代码声明了一个transport对象，用于获取全限定类名；第10~11行代码是通过反射对transport对象进行实例化。