一、填空题

1. 当希望一个类中的成员不能在类的外部访问时，应使用\_private 或 protected\_修饰符定义该成员。

2. 使用\_\_static\_\_修饰符定义的类成员，可以通过类直接访问而不需要创建对象后再访问。

3. 类的中一个成员是一个类的对象时，如果该成员没有被初始化，则该对象的初始值是\_null\_。

4. 在类的非静态成员函数中，使用\_\_\_this\_\_\_\_\_关键字来表示当前调用该函数的对象。

5. 假设A为一个类，则执行A [] array = new A[10];语句时，一共调用\_\_10\_\_\_次A的构造函数。

1. 单项选择题

1. 下面关于构造函数的说法不正确的是\_\_\_\_B\_\_\_\_\_。

A. 构造函数的调用时机是实例化对象时

B. 一个类必须且只能定义一个构造函数

C. 一个类可以不定义构造函数

D. 构造函数一定要和类名相同, 并且不能有返回值

2. 下列哪个修饰符可以使在一个类中定义的成员变量只能被同一包中的类访问？ B A．private B．无修饰符 C．public D．protected

3. 给出下列代码，如何使成员变量m 被方法fun（）直接访问\_\_\_\_C\_\_\_\_\_。  
 class Test   
 {   
 private int m;   
 public static void fun（）   
 {   
 ...   
 }   
 }   
 A．将private int m 改为protected int m   
 B．将private int m 改为 public int m   
 C．将private int m 改为 static int m   
 D．将private int m 改为 int m

4. 对于class A，如果在另一个包中的class B中，语句 A a = new A( ); a.m=10;成立，则下列定义正确的是\_\_\_C\_\_\_。

A．class A { int m; } B．class A {public int m; }

C．public class A{ public int m; } D．public class A { int m; }

5. 关于下面程序，说法正确的是\_\_\_\_C\_\_\_\_\_。

class AA{

private long i = 0;

AA(int i){

this.i =i;

}

String AA(long i) {

this.i = i;

return "i = " + this.i;

}

}

public class Test\_1\_6 {

public static void main(String[] args) {

System.out.println(new AA(10).AA(20));

}

}

A. 以上代码编译出错，一个类的构造函数不能有返回值；

B. 以上代码编译通过，输出结果为i = 10;

C. 以上代码编译通过，输出结果为i = 20;

D. 以上代码编译出错，一个类的构造函数不能重载；

6. 对于以下代码，说法正确的是\_\_\_\_A\_\_\_\_\_。

package homework.ch9.p1;

public class A {

private int i = 0;

protected int j = 0;

}

package homework.ch9.p2;

import homework.ch9.p1.A;

public class B extends A {

public void m() {

new A().j = 10;

this.j = 10;

}

}A．new A().j = 10; 有编译错误，在方法m里无法访问new A().j ;this.j无编译错误，在方法m里可以访问this.j;

B．new A().j = 10; 无编译错误，在方法m里可以访问new A().j ;this.j有编译错误，在方法m里不可以访问this.j;

C．new A().j = 10; 无编译错误，在方法m里可以访问new A().j ;this.j无编译错误，在方法m里可以访问this.j;

D．new A().j = 10; 有编译错误，在方法m里不可以访问new B().j ;this.j有编译错误，在方法m里不可以访问this.j;

三、判断对错题

1. 若a是类A的实例化对象，且a.fun();能顺利执行，则函数fun一定是实例方法。(错误)

2. protected修饰的类成员只能被其子类访问。(错误)

3. 类的静态变量被该类的所有实例共享。(正确)

4. Double类型的变量是值类型。(正确)

5. 当类的实例方法的形参变量与类的实例变量同名时，优先访问形参变量。(正确)

四、阅读下列程序，写出输出结果：

public class Circle

{

private double radius;

public static int count = 0;

public Circle(double r){

radius = r;

count ++;

}

public Circle(){

this(1.0);

}

public static void main(String[] args)

{

Circle c1 = new Circle ();

Circle c2 = new Circle (15.0);

c1. count ++;

c2. count ++;

Circle. count ++;

System.out.println("count = "+ count);

}

}

输出结果为\_\_\_\_\_count = 5\_\_\_\_\_\_\_\_\_\_。

五、编程题

1. 教材9.8题。

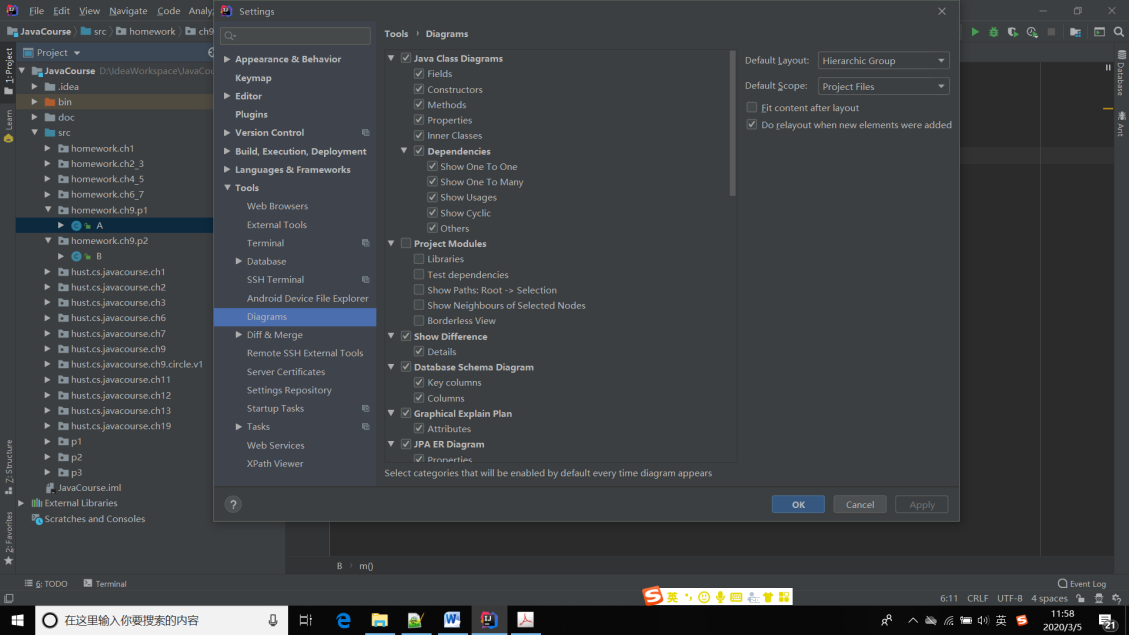
风扇类Fan。设计一个名为Fan的类来表示一个风扇。这个类包括：

1. 三个名为 SLOW、MEDIUM和FAST而值为1、2和3的常量，表示风扇的速度
2. 一个名为speed的int类型私有数据域，表示风扇的速度（默认值为SLOW）
3. 一个名为on的boolean类型私有数据域，表示风扇是否打开（默认值false）
4. 一个名为radius的double类型私有数据与，表示风扇的半径（默认值为5）
5. 一个名为color的string类型数据域，表示风扇的颜色（默认值为blue）
6. 这四个数据域的访问器和修改器
7. 一个创建默认风扇的无参构造方法
8. 一个名为 toString() 的方法返回描述风扇的字符串。如果风扇是打开的，那么该方法在一个组合的字符串中返回风扇的速度、颜色和半径。如果风扇没有打开，该方法就会返回一个由 “fan is off” 和风扇颜色及半径组合成的字符串

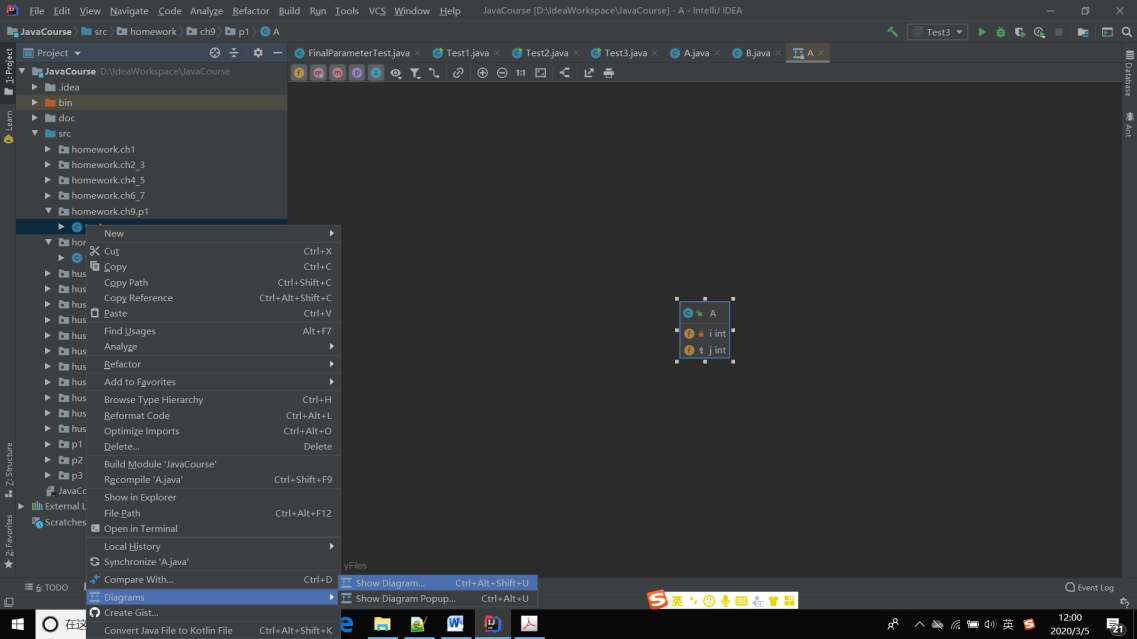
画出该类的UML图并实现这个类。编写一个测试程序，创建两个Fan对象。将第一个对象设置为最大速度、半径为10、颜色为yellow、状态为打开。将第二个对象设置为中等速度、半径为5、颜色为blue、状态为关闭。通过调用他们的toString 方法显示这些对象。

要求：除了提交工程文件外，请给出运行结果屏幕截图并放在Word文档里提交。

Tips：在IDEA里可以产生实现好的类的UML图。首先点击settings



然后选择一个类，点击鼠标右键。可以将生成的Diagram导出到图片里。



代码：

public class Fan {

static final int SLOW = 1;

static final int MEDIUM = 2;

static final int FAST = 3;

private int speed;

private boolean on;

private double radius;

private String color;

Fan() {

this.speed = SLOW;

this.on = false;

this.radius = 5;

this.color = "blue";

}

public void setSpeed(int sp) {

this.speed = sp;

}

public int getSpeed() {

return this.speed;

}

public void setOn(boolean o) {

this.on = o;

}

public boolean getOn() {

return this.on;

}

public void setRadius(double r) {

this.radius = r;

}

public double getRadius() {

return this.radius;

}

public void setColor(String c) {

this.color = c;

}

public String getColor() {

return this.color;

}

public String toString() {

String res;

if(this.on) {

res = "fan: speed " + this.speed;

} else {

res = "fan is off";

}

res += ", color " + this.color + ", radius " + this.radius;

return res;

}

public static void main(String[] args) {

Fan fan1 = new Fan();

Fan fan2 = new Fan();

fan1.setSpeed(Fan.FAST);

fan1.setRadius(10);

fan1.setColor("yellow");

fan1.setOn(true);

fan2.setSpeed(Fan.MEDIUM);

fan2.setRadius(5);

fan2.setColor("blue");

fan2.setOn(false);

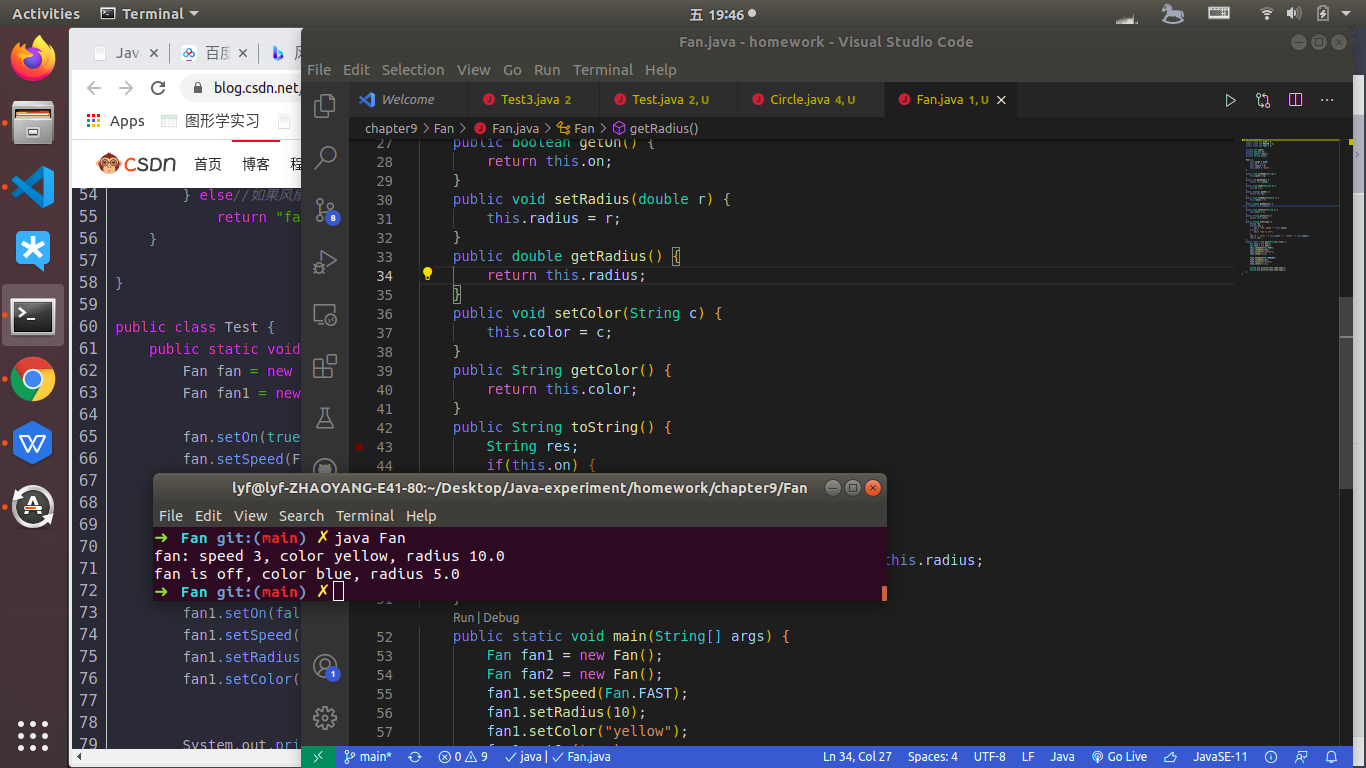
System.out.println(fan1.toString());

System.out.println(fan2.toString());

}

}

运行测试：



UML图：使用plantUML绘制

