一、实验目的

（1）能根据待测软件的特点，选择合适的方法对软件进行黑盒测试(功能测试)；
（2）学习测试用例的书写。

二、实验内容

（一）实验 1：随机测试 VS 黑盒测试 VS 白盒测试

题目内容：

在游戏引擎开发中，检测物体碰撞是一项重要的基础功能。

为简单起见，我们这里只考虑二维平面空间的情况，并用RectManager 程序判断平面上任意两矩形的相交关系（A:不相交，B:相交：B1:相交为一个区域，B12:包含，B13:完全重合，B2:交点为1个点，B3:交点为1 条线段），如果相交，则同时给出相交部分的面积。

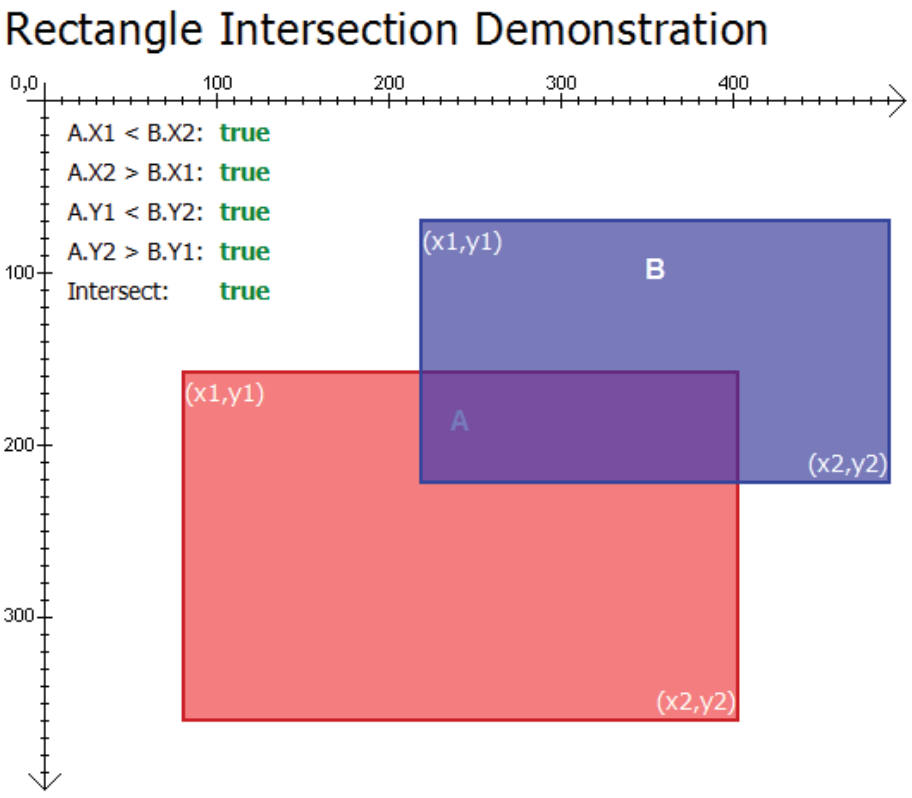
我们假设二维平面为iphone4 屏幕(960*640 分辨率)，且所有矩形的边都与坐标轴平行。

计算机图形学中，通常用左上角和右下角的坐标来表示一个矩形。

任意两个矩形的关系可借用这个工具来辅助分析：

http://silentmatt.com/rectangle-intersection/

坐标系请参照下图：



(1)请编写一简单程序，随机生成两个矩形的数据，请用这些随机数据对RectManager 进行测试。在测试用例生成程序中，可调用RectManager 中的方法直接驱动测试执行。
(2)请选择一种黑盒测试方法，设计相应的测试用例来测试程序；
程序运行命令行：java -jar RectManager.jar
(3)请分析RectManager 的实现源代码，利用**基本路径测试**方法，设计相应的测试用例来测试程序；只针对solve()方法进行测试。**为减少工作量，这里可不用处理复合条件。**
(4)在上述实验的基础上总结三种测试方法发现缺陷的能力上有何差别。

1、随机测试：

a)随机测试设计方案：

因为屏幕的范围是960*640, 且要考虑部分非法值情况，所以设计如下测试用例：
矩形各属性值范围：

Top	Bottom	Left	Right
640(+/-)6	640(+/-)6范围内且大于等于Top值	960(+/-)8	960(+/-)8范围内且大于等于Left值

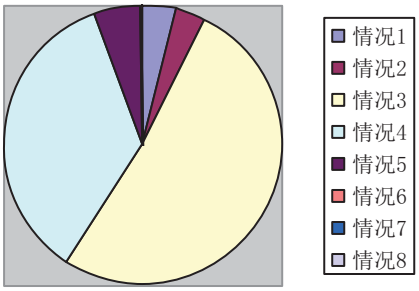
按上面的属性范围，随机生成测试用例：
测试用例数：1000000；

随机生成用例代码：

```
Random ra = new Random();
int la, lb;
la = ra.nextInt(960 + 10) - 5;
lb = ra.nextInt(960 + 10) - 5;
A.left = la < lb ? la : lb;
A.right = la > lb ? la : lb;
la = ra.nextInt(640 + 6) - 3;
lb = ra.nextInt(640 + 6) - 3;
A.top = la < lb ? la : lb;
A.bottom = la > lb ? la : lb;
```

b)随机测试统计：

序号	实验结果	出现次数	所在比例
1	Input error in Rectangle A	38612	0.038612
2	Input error in Rectangle B	37125	0.037125
3	矩形不相交！	512873	0.512873
4	矩形相交于一个区域！	356611	0.356611
5	矩形相交于一个区域且为包含关系！	51907	0.051907
6	矩形相交于一个区域且正好重合！	0	0.0
7	矩形相交于一个区域且交点为1个点！	6	0.000006
8	矩形相交于一个区域且交点为1条线段！	2867	0.002867



c)随机测试分析：

通过随机测试过程来看，我们发现实验结果的部分情况基本与理论上是相等的。但是也可以清楚地看到随机测试的不足，对于情况6（矩形相交于一个区域且正好重合）的出现次数为0，虽然已经进行了1000000此次数数据的生成，但是还是不能均匀的覆盖各种情况，也不能保证会出现大量重复的冗余的测试数据。所以，随机测试无法保证能测试到所有可能的情况，而大量的冗余、重复测试数据也是一种性能、资源的双重浪费。

2、黑盒测试：

选择等价类划分、正、反面测试、边界值分析法相结合的黑盒测试方法来进行程序测试。

1） **等价类划分**

等价类划分			
有效等价类		无效等价类	
1	不相交	9	输入含有负数
2	相交为一个区域	10	输入含有非数字非法字符
3	包含	11	输入值超过整型范围
4	完全重合	12	Left或right值超过960
5	交点为1个点	13	Top或bottom值超过640
6	交点为1条线段	14	Left值大于Right值
7	输入的数值构成一个点	15	Top值大于Bottom值
8	输入的数值构成一条线		

2） **边界值分析**

说明：由于进行的是黑盒测试，所以对每一个输入的参数都要进行边界值分析，这里采用6n+1的方法。

边界值分析							
数据名称	范围	MIN-1	MIN	MIN+1	MAX-1	MAX	MAX+1
A.Top	0——640	-1	0	1	639	640	641
B.Top	0——640	-1	0	1	639	640	641
A.Left	0——960	-1	0	1	959	960	961
B.Left	0——960	-1	0	1	959	960	961
A.Bottom	0——640	-1	0	1	639	640	641
B.Bottom	0——640	-1	0	1	639	640	641
A.Right	0——960	-1	0	1	959	960	961
B.Right	0——960	-1	0	1	959	960	961

3） **测试用例的生成：**

说明：通过上述边界值分析的方法，原则上要产生6*8+1=49个测试用例，但是由于本次试验为手工构造，所以在测试用例表中，省略部分用例内容。

序号	测试用例（Left,Top,Right,Bottom）	期望输出	实际输出	说明
001	A(10,10,20,20),B(30,30,40,40)	矩形不相交！	矩形不相交！	覆盖有效等价类1
002	A(10,10,20,20),B(15,15,40,40)	矩形相交于一个区域！	矩形相交于一个区域！	覆盖有效等价类2

003	A(10,10,40,40),B(15,15,30,30)	矩形相交于一个区域且为包含关系!	矩形相交于一个区域且为包含关系!	覆盖有效等价类3
004	A(10,10,40,40),B(10,10,40,40)	矩形相交于一个区域且正好重合!	矩形相交于一个区域且正好重合!	覆盖有效等价类4
005	A(10,10,40,40),B(40,40,60,60)	矩形相交于一个区域且交点为1个点	矩形相交于一个区域且交点为1个点	覆盖有效等价类5
006	A(10,10,40,40),B(40,10,60,40)	矩形相交于一个区域且交点为1条线段	矩形相交于一个区域且交点为1条线段	覆盖有效等价类6
007	A(10,10,10,10),B(40,20,60,40)	矩形不相交	矩形不相交	覆盖有效等价类7
008	A(10,20,10,30),B(40,20,60,40)	矩形不相交	矩形不相交	覆盖有效等价类8
009	A(-2,20,10,30),B(40,20,60,40)	Input error	Input error	覆盖无效等价类9
010	A(a,b,10,30),B(40,20,60,40)	Error	Error	覆盖无效等价类10
011	A(99999999999,2,1,3),B(4,2,6,4)	Error	Error	覆盖无效等价类11
012	A(10,20,1000,30),B(40,20,60,40)	Input error	Input error	覆盖无效等价类12
013	A(10,20,10,30),B(40,20,700,40)	Input error	Input error	覆盖无效等价类13
014	A(100,20,10,30),B(40,20,700,40)	Input error	Input error	覆盖无效等价类14
015	A(10,200,10,30),B(40,20,700,40)	Input error	Input error	覆盖无效等价类15
016	A(-1,10,10,10),B(40,20,60,40)	Input error	Input error	覆盖无效等价类10，A.Left最小值-1
017	A(0,10,10,30),B(40,20,60,40)	矩形相交于一个区域!	矩形相交于一个区域!	覆盖有效等价类2，A.Left最小值
018	A(1,10,10,30),B(40,20,60,40)	矩形相交于一个区域!	矩形相交于一个区域!	覆盖有效等价类2，A.Left最小值+1
019	A(959,10,10,30),B(40,20,60,40)	Input error	Input error	覆盖无效等价类14，A.Left最大值-1
020	A(960,10,10,30),B(40,20,60,40)	Input error	Input error	覆盖无效等价类14，A.Left最大值
021	A(961,10,10,30),B(40,20,60,40)	Input error	Input error	覆盖无效等价类14，A.Left最大值

				+1
◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦ ◦				
064	A(10,50,10,100),B(40,20,60,961)	Input error	Input error	覆盖无效等价类14，B.Right最大值+1

3、白盒测试：

Solve函数源代码：

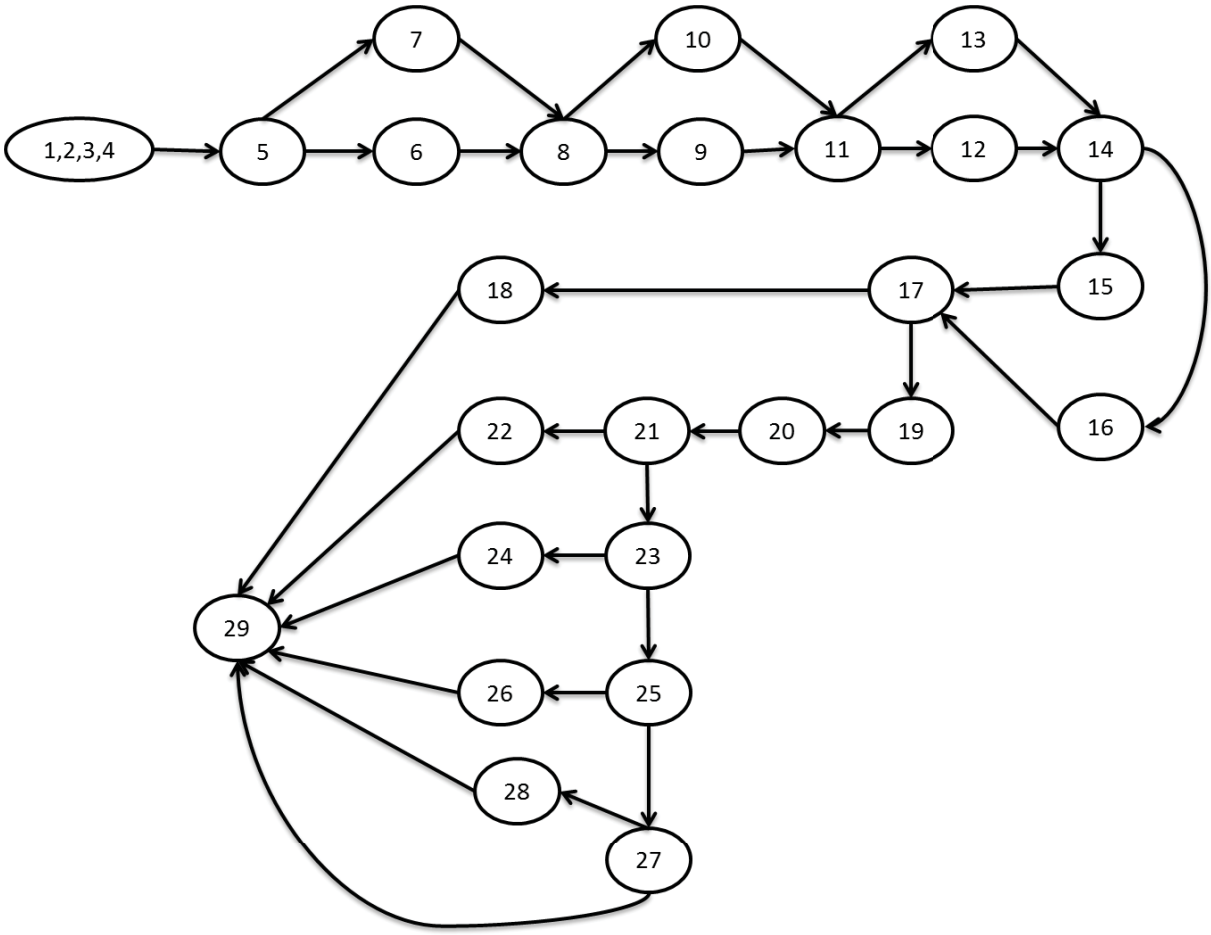
```
private void solve(Rect A, Rect B) {
1       int nMaxLeft = 0;
2       int nMaxTop = 0;
3       int nMinRight = 0;
4       int nMinBottom = 0;
5       if (A.left >= B.left) {
6               nMaxLeft = A.left;
        } else {
7               nMaxLeft = B.left;
        }
8       if (A.top >= B.top) {
9               nMaxTop = A.top;
        } else {
10              nMaxTop = B.top;
        }
11      if (A.right <= B.right) {
12              nMinRight = A.right;
        } else {
13              nMinRight = B.right;
        }
14      if (A.bottom <= B.bottom) {
15              nMinBottom = A.bottom;
        } else {
16              nMinBottom = B.bottom;
        }
17      if ((nMaxLeft > nMinRight) || (nMaxTop > nMinBottom)) {
18              nFlag = 0;

        } else {
19              nFlag = 1;
20              area = (nMinRight - nMaxLeft + 1) * (nMinBottom - nMaxTop + 1);

21              if ((B.left == A.left) && (B.right == A.right) && (B.top == A.top) &&
```

```
(B.bottom == A.bottom)) {
22                      nFlag = 3;
23              } else if (((nMaxLeft == A.left) && (nMinRight == A.right) && (nMaxTop
  == A.top) && (nMinBottom == A.bottom))
                                    || ((nMaxLeft == B.left) && (nMinRight == B.right) &&
  (nMaxTop == B.top) && (nMinBottom == B.bottom))) {
24                      nFlag = 2;
25              } else if ((nMaxLeft == nMinRight) && (nMaxTop == nMinBottom)) {
26                      nFlag = 4;
27              } else if (((nMaxLeft == nMinRight) && (nMaxTop < nMinBottom))
                                    || ((nMaxLeft < nMinRight) && (nMaxTop == nMinBottom)))
  {
28                      nFlag = 5;
                }
        }
29
}
```

程序流图：



通过流图不难计算出基本路径的条数（环复杂度）：

边数-节点数+2=34-26+2=10条

基本路径测试用例：

序号	测试用例(left,top,right,bottom)	路径
001	A(20,20,30,30),B(10,10,15,30)	1,2,3,4->5->6->8->9->11->13->14->15->17->18->29
002	A(10,20,18,30),B(20,65,30,40)	1,2,3,4->5->7->8->10->11->12->14->16->17->18-29
003	A(10,20,40,30),B(20,25,30,30)	1,2,3,4->5->7->8->10->11->12->14->15->17->19->20->21->23->25->27->29
004	A(10,20,30,40),B(10,20,30,40)	1,2,3,4->5->6->8->9->11->12->14->15->17->19->20->21->22->29
005	A(10,20,30,50),B(10,20,30,40)	1,2,3,4->5->6->8->9->11->12->14->16->17->19->20->21->23->24->29
006	A(10,20,30,40),B(10,20,30,50)	1,2,3,4->5->6->8->9->11->12->14->15->17->19->20->21->23->24->29
007	A(10,20,30,40),B(30,40,50,60)	1,2,3,4->5->7->8->10->11->12->14->15->17->19->20->21->23->25->26->29
008	A(30,40,50,60),B(10,20,30,40)	1,2,3,4->5->6->8->9->11->13->14->16->17->19->20->21->23->25->26->29
009	A(10,20,30,40),B(30,20,60,40)	1,2,3,4->5->7->8->9->11->12->14->15->17->19->20->21->23->25->27->28->29
010	A(30,20,60,40),B(10,20,30,40)	1,2,3,4->5->6->8->9->11->13->14->15->17->19->20->21->23->25->27->28->29

4、三种测试方法的综合分析：

通过三种不同的测试方法，可以分析得到，对于随机测试，虽然构造用例方法简单、测试用例数量大，但是无法在有限的测试用例中涵盖所有不同的方面，而且还会出现大量冗余、重复的测试用例，遗漏掉大量边界值所需考虑的特殊情形，所以随机测试局限性很大，发现缺陷的能力很弱。

而对于黑盒测试方案中，对问题进行了综合的分析后，划分等价类，并结合正、反面测试以及边界值分析法设计出了很多有针对性的测试用例。这些用例较全面覆盖了输入的各种情形，有较强的发现错误的能力。

最后进行了白盒测试，在本次实验中，选择的白盒测试方法是基于基本路径的测试方法，所以通过流图设计出了10条测试用例，基本路径虽然对程序的每个判断阶段进行的覆盖，但是仍然忽略了很多诸如边界值、非法值等特殊情形，缺乏发现缺陷的能力。

综上，在本次实验中黑盒测试方案是发现问题能力最强的，而随机测试和白盒测试均有很大的不足。但是，这样不代表黑盒测试就是最有效的方法，在实际的项目中，只有综合运用3中测试技术才能更好的发现程序的缺陷。

（二）实验 2：蜕变测试问题

在很多软件测试活动中，人们发现给出一个测试用例的期望输出是一件很困难的事情，在一些情况下，人们甚至无法给出测试用例的预期输出。这种测试预测输出无法获得的问题，就称为测试

预言问题(Test Orcale Problem)。

为解决Test Orcale 问题，1998 年T.Y. Chen 提出的蜕变测试的概念。Sin.exe 是一个用某种数值计算方法求解数学函数f(x)=sin(x)的程序。请设计测试用例，实现对该程序的黑盒测试。

要求：

(1) 给出每个测试用例设计的理由；

(2) 这个实验中展示的测试用例输出值无法预测的情形还可能出现在哪些实际应用中。

解答：

（1）给出每个测试用例设计的理由

问题分析：

f(x)=sin(x)是典型的orcale问题，即我们在测试过程中无法确定执行的结果与期望的结果是否相同。所以，我们需要利用蜕变测试的思想解决本次测试问题。蜕变测试方法通过检查程序的多个执行结果之间的关系来测试程序的，不需要构造预期的输出。

解决方案：

1、 首先构造蜕变关系MR（输入之间的关系R，输出之间的关系Rf，MR（R，Rf））

2、 生成原始测试用例

3、 根据R，在原始测试用例基础上生成衍生的测试用例

4、 检查原始和衍生的测试用例的输出是否满足Rf，从而得到测试结果。

关键性问题：蜕变的关系的构造

Sin函数的蜕变关系	
序号	蜕变关系MR
001	Sin(X+k*360)=Sin(X)　k为整数
002	Sin(X+180)=-Sin(X)
003	Sin(-X)=-Sin(X)
004	Sin(X-180)=Sin(X)
005	Sin(360-X)=-Sin(X)
006	Sin(X)^2+Sin(90-X)^2=1

测试用例构造

序号	原始测试用例 X	衍生测试用例 Y	预期结果	构造理由
001	90	无	1	特殊值，已知预期结果
002	0	无	0	特殊值，已知预期结果
003	180	无	0	特殊值，已知预期结果
004	270	无	-1	特殊值，已知预期结果
005	360	无	0	特殊值，已知预期结果
006	-90	无	-1	特殊值，已知预期结果
007	-180	无	0	特殊值，已知预期结果

008	-270	无	1	特殊值，已知预期结果
009	-360	无	0	特殊值，已知预期结果
010	17(一般值)	360+17	f(X)=f(Y)	蜕变关系Sin(X+k*360)=Sin(X)
		180+17	f(X)=-f(Y)	Sin(X+180)=-Sin(X)
		-17	f(X)=-f(Y)	Sin(-X)=-Sin(X)
		-17-180	f(X)=-f(Y)	Sin(X-180)=-Sin(X)
		360-17	f(X)=-f(Y)	Sin(360-X)=-Sin(X)
		90-17	f(X)^2+f(Y)^2=1	Sin(X)^2+Sin(90-X)^2=1

说明：对于一般值的情况，可以综合利用蜕变关系来间接的进行测试，预测运行结果的正确性。

（2）这个实验中展示的测试用例输出值无法预测的情形还可能出现在哪些实际应用中。

三、实验体会

实验思考：

（1）通过测试，是否发现程序中存在的缺陷？

（2）经过一系列的实验，请谈谈你所感受的软件测试中存在哪些挑战性的难题。

实验体会：

（1） 发现程序中存在缺陷。当输入值为非整数时，程序会出现严重的错误，直接崩掉。例如：输入字符 a 或者实数 123.123 均产生严重的错误。当输入的整数值大于 int 所能容纳的位数时，也会出现错误，例如 101010101011 程序会立刻崩掉。

（2） 在软件测试课程学习过程中，我主要学习了白盒测试和黑盒测试，在学这门课之前对软件测试的概念仍然是很模糊的，通过几次测试实验后，我对软件测试方法有了一定的了解，也在进行测试过程中遇到了各种各样的难题。

我认为，在白盒测试中，最大的难题是要要求测试人员对程序代码要有一定的了解，并且要细心对代码进行走查，画流图、测试路径覆盖等方法都要求对程序代码仔细的阅读，不可大意，因而白盒测试设计测试用例难度比较大，而且比较繁琐。而对于黑盒测试而言，我觉得，测试人员不必了解程序实现代码，但是难题是要正确构造等价类、边界值分析等问题，这些特殊情形是保证黑盒测试有效地有力保障，另外一个难题是要考虑的十分全面的情况下，要减少冗余、重复的数据出现，保障测试的效率。