

Test 1

00	<code>public class Sei {</code>
01	<code> char f() { return '6'; }</code>
02	<code> public static void main(String e[]) {</code>
03	<code> Sei a = new Sei();</code>
04	<code> Sei b = new Sette();</code>
05	<code> Sette c = new Sette();</code>
06	<code> System.out.print(a.f() + " " + b.f() + " " + c.f() + "</code>
07	<code>");</code>
08	<code> char ch[] = {'A', 'B', 'A', 'B', 'A', 'B'};</code>
09	<code> int i1 = 0, i2 = 2, i3 = 4;</code>
10	<code> if (a.equals(b)) i1++;</code>
11	<code> if (b.equals(a)) i2++;</code>
12	<code> if (c.equals(b)) i3++;</code>
13	<code> System.out.println(ch[i1] + " " + ch[i2] + " " +</code>
14	<code>ch[i3]);</code>
15	<code> } }</code>
16	<code>class Sette extends Sei {</code>
17	<code> char f() { return '7'; }</code>
18	<code> public boolean equals(Object a) {</code>
19	<code> return (a instanceof Sei);</code>
20	<code> }</code>

Test 2

00	<code>public class Uno {</code>
01	<code> static Collection c=new HashSet();</code>
02	<code> public static void main(String a[]) {</code>
03	<code> Collection c=new LinkedList();</code>
04	<code> Uno u=new Uno();</code>
05	<code> c.add(u); c.add(u);</code>
06	<code> u.f();</code>
07	<code> System.gc();System.runFinalization();</code>
08	<code> System.out.println(c.size());</code>
09	<code> }</code>
10	<code> void f() {</code>
11	<code> A a=new A("K");</code>
12	<code> A b=new A("L");</code>
13	<code> c.add(b);</code>
14	<code> }</code>
15	<code> class A {</code>
16	<code> String s="";</code>
17	<code> A(String s) {this.s=s; System.out.print(this);}</code>
18	<code> public String toString(){return s;}</code>
19	<code> public void finalize(){System.out.print(this);}</code>
20	<code> } }</code>

Test 3

01	<code>public class Due {</code>
02	<code> static Collection<Due> s=new HashSet<Due>();</code>
03	<code> int k,j;</code>
04	<code> Due(int k, int j) {this.k=k; this.j=j;}</code>
05	<code> public boolean equals(Object d){</code>
06	<code> return k-j==((Due)d).j-((Due)d).k;</code>
07	<code> }</code>
08	<code> public int hashCode(){return 1;}</code>
09	<code> public static void main(String[] m){</code>
10	<code> s.add(new Due(1,2)); s.add(new Due(0,1));</code>
11	<code> s.add(new Due(2,1)); s.add(new Due(1,0));</code>
12	<code> System.out.print(s.size());</code>
13	<code> for (Due x:s){System.out.print(x.k+" "+x.j);}</code>
14	<code> }</code>
15	<code> public static void main(String m){</code>
16	<code> s.add(new Due(1,0));</code>
17	<code> System.out.print(s.size());</code>
18	<code> } }</code>

Test 4

01	<code>public class Due {</code>
02	<code> static Collection<Due> s=new HashSet<Due>();</code>
03	<code> static int k,j;</code>
04	<code> Due(int k, int j) {this.k=k; this.j=j;}</code>
05	<code> public boolean equals(Object d){</code>
06	<code> return k-j==((Due)d).j-((Due)d).k;</code>
07	<code> }</code>
08	<code> public int hashCode(){return 1;}</code>
09	<code> public static void main(String[] m){</code>
10	<code> s.add(new Due(1,2)); s.add(new Due(0,1));</code>
11	<code> s.add(new Due(2,1)); s.add(new Due(1,0));</code>
12	<code> System.out.print(s.size());</code>
13	<code> for (Due x:s){System.out.print(x.k+" "+x.j);}</code>
14	<code> } }</code>

Test 5

01	<code>public class Due {</code>
02	<code> Collection<Due> s=new HashSet<Due>();</code>
03	<code> static int k,j;</code>
04	<code> Due(int k, int j) {this.k=k; this.j=j;}</code>
05	<code> public boolean equals(Object d){</code>
06	<code> return k-j==((Due)d).j-((Due)d).k;</code>
07	<code> }</code>
08	<code> public int hashCode(){return 1;}</code>
09	<code> public static void main(String[] m){</code>
10	<code> s.add(new Due(1,2)); s.add(new Due(0,1));</code>
11	<code> s.add(new Due(2,1)); s.add(new Due(1,0));</code>
12	<code> System.out.print(s.size());</code>
13	<code> for (Due x:s){System.out.print(x.k+" "+x.j);}</code>
14	<code> } }</code>

Test 6

00	<code>public class Tre {</code>
01	<code> class A {</code>
02	<code> public A(int k) {System.out.print(k);};</code>
03	<code> public void finalize() {System.out.print("A");}</code>
04	<code> }</code>
05	<code> class B extends A {</code>
06	<code> public B(int k) {System.out.print(k);};</code>
07	<code> public void finalize() {System.out.print("A");}</code>
08	<code> }</code>
09	<code> public static void main (String z[]){</code>
10	<code> new Tre();</code>
11	<code> }</code>
12	<code> Tre(){</code>
13	<code> A a=new B(3);</code>
14	<code> B b=(B)a;</code>
15	<code> a=null;</code>
16	<code> b=new B(3);</code>
17	<code> System.gc(); System.runFinalization();</code>
18	<code> } }</code>

Test 7

01	<code>#include <iostream></code>
02	<code>using namespace std;</code>
03	<code>int x[] = {-2, -1, 0, 1, 2};</code>
04	<code>void f(int* x, int y[]) {</code>
05	<code> x[*y] = -y[*x];</code>
06	<code>}</code>
07	<code>int main(int argc, char** argv) {</code>
08	<code> int * p = x + 1;</code>
09	<code> f(p, p);</code>
10	<code> for (int * s = x; s < x + 5; s++) {</code>
11	<code> cout << *s;</code>
12	<code> }</code>
13	<code> return 0;</code>
14	<code>}</code>

Test 8

01	<code>public class F{</code>
02	<code> int x=2;</code>
03	<code> F(int x) {</code>
04	<code> f(x);</code>
05	<code> f();</code>
06	<code> System.out.println(x);</code>
07	<code> }</code>
08	<code> void f() { x++; System.out.print(x);}</code>
09	<code> void f(int x) { this.x++; x--;System.out.print(x);}</code>
10	<code> public static void main(String arg[]) {</code>
11	<code> F x=new F(9);</code>
12	<code> }}}</code>