abstract class A

{

abstract void test1();

abstract void test2();

abstract void test3();

abstract void test4();

}

interface B

{

}

interface C

{

int i = 10;

String s1 = "abc";

double j = 1.2;

void test1();

void test2();

void test3();

}

abstract interface D

{

static int i = 10;

public static String s1 = "abc";

final static public double j = 1.2;

void test1();

public void test2();

abstract public void test3();

}

interface E

{

int i;

}

Console:

E.java:3: error: = expected

int i;

^

1 error

interface F

{

protected int i = 10;

}

Console:

F.java:3: error: modifier protected not allowed here

protected int i = 10;

^

1 error

interface G

{

private void test1();

}

Console:

G.java:3: error: modifier private not allowed here

private void test1();

^

1 error

interface H

{

H()

{

}

}

Console:

H.java:3: error: <identifier> expected

H()

^

1 error

interface I

{

void test()

{

}

}

Console:

I.java:4: error: interface abstract methods cannot have body

{

^

1 error

interface J

{

static

{

}

{

}

}

Console:

J.java:4: error: illegal start of type

{

^

J.java:5: error: = expected

}

^

J.java:5: error: ';' expected

}

^

J.java:9: error: class, interface, or enum expected

}

^

4 errors

interface A

{

}

class K

{

public static void main(String args[])

{

A a1 = new A();

System.out.println("done");

}

}

Console:

K.java:8: error: A is abstract; cannot be instantiated

A a1 = new A();

^

1 error

interface A

{

}

class L

{

public static void main(String args[])

{

A a1 = null;

A a2 = null;

a1 = a2;

System.out.println("done");

}

}

interface A

{

}

class M

{

A obj;

M(A obj)

{

}

A test1()

{

return null;

}

void test2(A obj)

{

}

public static void main(String[] args)

{

System.out.println("done");

}

}

Console:

done

interface A

{

int i = 10;

}

class N

{

public static void main(String[] args)

{

System.out.println(A.i);

}

}

Console:

10

interface A

{

void test1();

}

class B implements A

{

public void test1()

{

System.out.println("from test1()");

}

}

class O

{

public static void main(String[] args)

{

B b1 = new B();

b1.test1();

System.out.println("done");

}

}

Console:

from test1()

done

interface A

{

void test1();

}

class B implements A

{

void test1()

{

System.out.println("from test1()");

}

}

class P

{

public static void main(String[] args)

{

B b1 = new B();

b1.test1();

System.out.println("done");

}

}

Console:

P.java:7: error: test1() in B cannot implement test1() in A

void test1()

^

attempting to assign weaker access privileges; was public

1 error

interface A

{

void test1();

void test2();

}

class Q implements A

{

public void test1()

{

}

public static void main(String[] args)

{

System.out.println("done");

}

}

Console:

Q.java:6: error: Q is not abstract and does not override abstract method test2() in A

class Q implements A

^

1 error

interface A

{

void test1();

void test2();

}

abstract class R implements A

{

public void test1()

{

}

public static void main(String[] args)

{

System.out.println("done");

}

}

Console:

done

interface A

{

void test1();

void test2();

}

class S implements A

{

public void test1()

{

System.out.println("S.test1()");

}

public void test2()

{

System.out.println("S.test2()");

}

public static void main(String[] args)

{

S s1 = new S();

s1.test1();

s1.test2();

System.out.println("done");

}

}

Console:

S.test1()

S.test2()

done

interface A

{

void test1();

void test2();

void test3();

}

abstract class B implements A

{

public void test1()

{

System.out.println("B.test1()");

}

}

abstract class C extends B

{

public void test2()

{

System.out.println("C.test1()");

}

}

class T extends C

{

public void test3()

{

System.out.println("T.test1()");

}

public static void main(String args[])

{

T t1 = new T();

t1.test1();

t1.test2();

t1.test3();

System.out.println("done");

}

}

Console:

B.test1()

C.test1()

T.test1()

done