1. Given

**class** Foo {

**class** Bar{ }

}

**class** Test {

**public static void** main (String [] args){

Foo f = **new** Foo();

/\* Line 10: Missing statement ? \*/

}

}

which statement, inserted at line 10, creates an instance of Bar?

1. Foo.Bar b = **new** Foo.Bar();
2. Foo.Bar b = f.**new** Bar();
3. Bar b = **new** f.Bar();
4. Bar b = f.**new** Bar();
5. Which statement is true about a static nested class?
6. You must have a reference to an instance of the enclosing class in order to instantiate it.
7. It does not have access to nonstatic members of the enclosing class.
8. It's variables and methods must be static.
9. It must extend the enclosing class.

3). What modifiers can be used with inner classes?

1. abstract
2. public
3. static
4. final
5. Which statement(s) about inner classes is (are) TRUE?
6. It must be declared final
7. It can only be used in enclosing class
8. Enclosing class can also be inner class
9. It is member of enclosing class
10. Which statement is true about an inner class?
11. You must have a reference to an instance of the enclosing class in order to instantiate it.
12. It does not have access to non-static members of the enclosing class.
13. It's variables and methods must be static.
14. It must extend the enclosing class.
15. Given

**class** Foo {

**static** class Bar{ }

}

**class** Test {

**public static void** main (String [] args){

Foo f = **new** Foo();

/\* Line 10: Missing statement ? \*/

}

}

which statement, inserted at line 10, creates an instance of Bar?

1. Foo.Bar b = **new** Foo.Bar();
2. Foo.Bar b = f.**new** Bar();
3. Bar b = **new** f.Bar();
4. Bar b = f.**new** Bar();
5. Given

**class** OuterClass {

**public** **int** var = 0;

**public class** InnerClass {

**public void** output(){

/\* Line 5: missing statement \*/

}

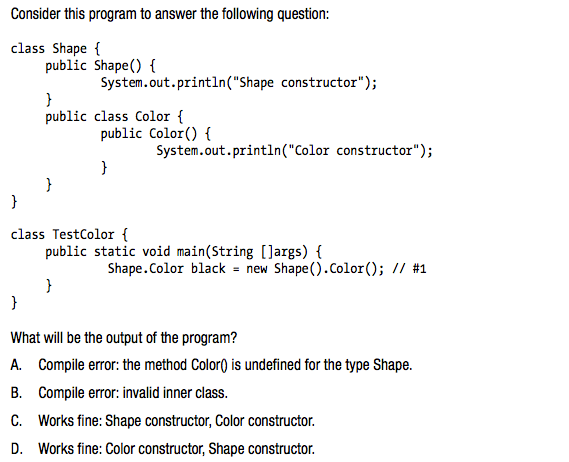
}

}

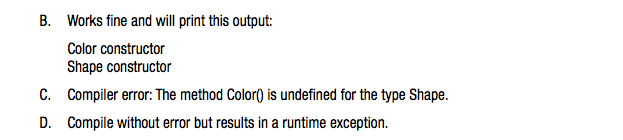
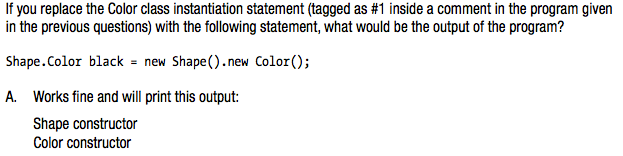
Which statement(s) inserted at line 5 will output var?

1. System.out.println(OuterClass.var);
2. System.out.println(OuterClass.this.var);
3. System.out.println(var);
4. System.out.println(this.var);
5. Consider inner interface. Which access level can it have?
6. public
7. private
8. protected
9. package private

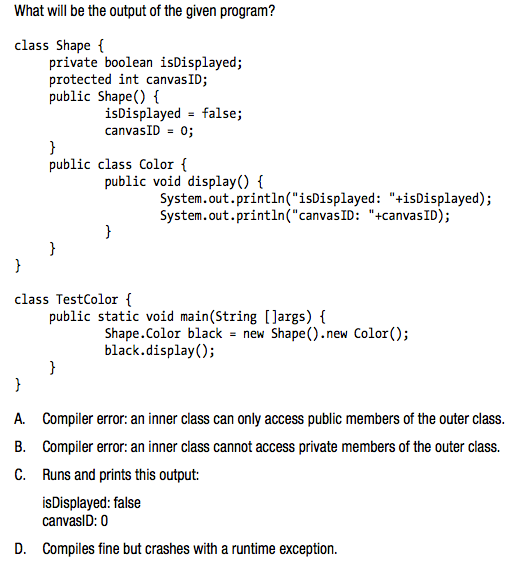
9)



10)



11)



12) Результат выполнения?

**class** Outer {

**protected int** x = 5;

**private class** Inner **extends** Outer {

**public** Inner() {

x = 3;

}

**public void** method() {

System.out.println(Outer.**this**.x);

System.out.println(x);

}

}

**public static void** main(String[] args) {

**new** Outer().**new** Inner().method();

}

}

1. 3 5
2. 3 3
3. 5 3
4. 5 5
5. Compilation error

13) Результат выполнения?

**class** Outer2 {

**static public class** Inner {

**public class** InInner {

**public int** y;

}

}

}

**class** Test {

**public static void** main(String[] args) {

System.out.println(Outer2.Inner.**new** InInner().y);

}

}

1. 0
2. Initialization error (Compilation error)
3. Cannot find symbol (Compilation error)

14). В каких строках ошибки компиляции?

**class** Mult {

**public** Mult getMultiplier(int x) { //1

**return new** Mult() { //2

**public void** multiply() { //3

System.out.println(x \* 2); //4

}

};

}

}

**class** BlaBla {

**public static void** main(String[] args) {

**new** Mult().getMultiplier(3).multiply(); //5

}

}

1. 1, 2
2. 4
3. 4 5
4. 5
5. 2

15) Будет ли ошибка компиляции?

**interface** Interface {

**class** Clazz {

**interface** Interface2 {

**int** x = 4;

}

}

**class** MyClass **implements** **Clazz.Interface2** {

}

}

**class** InterfaceTest {

**public static void** main(String[] args) {

System.out.println(**new** Interface.MyClass().x);

}

}

1. ага
2. не

16) Что произойдет?

**public class** A {

**public static** class X {

**public static class** Y {

**public static** String Z = "life is good";

}

**public static** C Y;

}

**public static class** C {

**public static** String Z = "life is pain";

}

**public static void** main(String[] args) {

System.out.println(X.Y.Z);

}

}

1. Compile error
2. Runtime error
3. Output: “life is good”
4. Output: “life is pain”