1. Given:

```
class Plane {
static String s = "-";
public static void main(String[] args) {
new Plane().s1();
System.out.println(s);
}
void s1() {
try { s2(); }
catch (Exception e) { s += "c"; }
}
void s2() throws Exception {
s3(); s += "2";
s3(); s += "2b";
}
void s3() throws Exception {
throw new Exception();
} }
```

What is the result?
A. -
B. -c
C. -c2
D. -2c
E. -c22b
F. -2c2b
G. -2c2bc
H. Compilation fails.

2. Given:

```
class Emu {
static String s = "-";
public static void main(String[] args) {
try {
throw new Exception();
} catch (Exception e) {
try {
try { throw new Exception();
} catch (Exception ex) { s += "ic "; }
throw new Exception(); }
catch (Exception x) { s += "mc "; }
finally { s += "mf "; }
} finally { s += "of "; }
System.out.println(s);
} }
```

What is the result?
A. -ic of
B. -mf of
C. -mc mf
D. -ic mf of
E. -ic mc mf of
F. -ic mc of mf
G. Compilation fails.

3. Given:

```
3. class SubException extends Exception { }
4. class SubSubException extends SubException { }
5.
6. public class CC { void doStuff() throws SubException { } }
7.
8. class CC2 extends CC { void doStuff() throws SubSubException { } }
9.
10. class CC3 extends CC { void doStuff() throws Exception { } }
11.
12. class CC4 extends CC { void doStuff(int x) throws Exception { } }
13.
14. class CC5 extends CC { void doStuff() { } }
```

What is the result? (Choose all that apply.)
A. Compilation succeeds
B. Compilation fails due to an error on line 8

C. Compilation fails due to an error on line 10
D. Compilation fails due to an error on line 12
E. Compilation fails due to an error on line 14

4. Given:

```
3. public class Gotcha {
4. public static void main(String[] args) {
5. // insert code here
6.
7. }
8. void go() {
9. go();
10. }
11. }
```

And given the following three code fragments:

```
I. new Gotcha().go();
II. try { new Gotcha().go(); }
catch (Error e) { System.out.println("ouch"); }
III. try { new Gotcha().go(); }
catch (Exception e) { System.out.println("ouch"); }
```

When fragments I - III are added, independently, at line 5, which are true? (Choose all that apply.)
A. Some will not compile
B. They will all compile
C. All will complete normally
D. None will complete normally
E. Only one will complete normally
F. Two of them will complete normally

5. Given:

```
1. public class Frisbee {
2. // insert code here
3. int x = 0;
4. System.out.println(7/x);
5. }
6. }
```

And given the following four code fragments:

```
I. public static void main(String[] args) {
II. public static void main(String[] args) throws Exception {
III. public static void main(String[] args) throws IOException {
IV. public static void main(String[] args) throws RuntimeException {
```

If the four fragments are inserted independently at line 4, which are true? (Choose all that apply.)
A. All four will compile and execute without exception
B. All four will compile and execute and throw an exception
C. Some, but not all, will compile and execute without exception
D. Some, but not all, will compile and execute and throw an exception
E. When considering fragments II, III, and IV, of those that will compile, adding a try/catch block around line 6 will cause compilation to fail

6. Given:

```
2. class MyException extends Exception { }
3. class Tire {
4. void doStuff() { }
5. }
6. public class Retread extends Tire {
7. public static void main(String[] args) {
8. new Retread().doStuff();
9. }
10. // insert code here
11. System.out.println(7/0);
12. }
13. }
```

And given the following four code fragments:

```
I. void doStuff() {
II. void doStuff() throws MyException {
III. void doStuff() throws RuntimeException {
IV. void doStuff() throws ArithmeticException {
```

When fragments I - IV are added, independently, at line 10, which are true? (Choose all that apply.)

A. None will compile
B. They will all compile
C. Some, but not all, will compile
D. All of those that compile will throw an exception at runtime
E. None of those that compile will throw an exception at runtime
F. Only some of those that compile will throw an exception at runtime

7. Given:

```
2. abstract class Tool {
3. int SKU;
4. abstract void getSKU();
5. }
6. public class Hammer {
7. // insert code here
8. }
```

Which line(s), inserted independently at line 7, will compile? (Choose all that apply.)

A. void getSKU() { ; }
B. private void getSKU() { ; }
C. protected void getSKU() { ; }
D. public void getSKU() { ; }

8. Given:

```
5. abstract class Thing { static String s = ""; Thing() { s += "t "; } }
6. class Steel extends Thing {
7. Steel() { s += "s "; }
8. Steel(String s1) {
9. s += s1;
10. new Steel();
11. }
12. }
13. public class Tungsten extends Steel {
14. Tungsten(String s1) {
15. s += s1;
16. new Steel(s);
17. }
18. public static void main(String[] args) {
19. new Tungsten("tu ");
20. System.out.println(s);
21. } }
```

What is the result?

A. s tu s tu s
B. t s tu t s t s
C. t s tu t t s tu t s
D. t tu t s tu t t tu t s tu t s
E. Compilation fails.
F. An exception is thrown at runtime.

9. Given:

```
1. abstract class Vibrate {
2. static String s = "-";
3. Vibrate() { s += "v"; }
4. }
5. public class Echo extends Vibrate {
6. Echo() { this(7); s += "e"; }
7. Echo(int x) { s += "e2"; }
8. public static void main(String[] args) {
```

```
System.out.print("made " + s + " ");
}
static {
Echo e = new Echo();
System.out.print("block " + s + " ");
} }
```

What is the result?

A. made -ve2e
B. block -ee2v
C. block -ve2e
D. made -eve2 block -eve2
E. made -ve2e block -ve2e
F. block -ve2e made -ve2e
G. block -ve2e made -ve2eve2e
H. Compilation fails

10. Which declare a compilable abstract class? (Choose all that apply.)

A. public abstract class Canine { public Bark speak(); }
B. public abstract class Canine { public Bark speak() { } }
C. public class Canine { public abstract Bark speak(); }
D. public class Canine abstract { public abstract Bark speak(); }

11. Given:

```
public class Glank implements Vonk { Jooker[] j; }
abstract class Bostron { String yoodle; Bostron b; }
interface Protefor { }
interface Vonk extends Protefor { int x = 7; }
class Jooker { Bostron b; }
```

Which are true? (Choose all that apply.)

A. Glanks have a Bostron.
B. Jookers implement Protefors.
C. Glanks implement Bostrons.
D. Jookers have a String.
E. Bostrons implement Vonks.
F. Bostrons have a Bostron.

12 Given:

```
1. interface Syrupable {
2. void getSugary();
3. }
4. abstract class Pancake implements Syrupable { }
5.
6. class BlueBerryPancake implements Pancake {
7. public void getSugary() { ; }
8. }
9. class SourdoughBlueBerryPancake extends BlueBerryPancake {
10. void getSugary(int s) { ; }
11. }
```

Which are true? (Choose all that apply.)

A. Compilation succeeds.
B. Compilation fails due to an error on line 2.
C. Compilation fails due to an error on line 4.
D. Compilation fails due to an error on line 6.
E. Compilation fails due to an error on line 7.
F. Compilation fails due to an error on line 9.
G. Compilation fails due to an error on line 10.

13. Given:

```
class c1 { }
class c2 { }
interface i1 { }
interface i2 { }
class A extends c2 implements i1 { }
class B implements i1 implements i2 { }
class C implements c1 { }
class D extends c1, implements i2 { }
class E extends i1, i2 { }
class F implements i1, i2 { }
```

What is the result? (Choose all that apply.)

A. Class A does not compile.
B. Class B does not compile.
C. Class C does not compile.
D. Class D does not compile.
E. Class E does not compile.
F. Class F does not compile.
G. Compilation succeeds for all of the classes.

14. Given:

```
abstract interface Pixie {
abstract void sprinkle();
static int dust = 3;
}
abstract class TinkerBell implements Pixie {
String fly() { return "flying "; }
}
public class ForReal extends TinkerBell {
public static void main(String[] args) {
new ForReal().sprinkle();
}
public void sprinkle() { System.out.println(fly() + " " + dust); }
}
```

What is the result? (Choose all that apply.)

A. flying 3
B. Compilation fails because TinkerBell doesn’t properly implement Pixie.
C. Compilation fails because ForReal doesn’t properly extend TinkerBell.
D. Compilation fails because Pixie is not a legal interface.
E. Compilation fails because ForReal doesn’t properly implement Pixie.
F. Compilation fails because TinkerBell is not a legal abstract class.

15. Given:

```
interface Machine { }
interface Engine { }
abstract interface Tractor extends Machine, Engine {
void pullStuff();
}
class Deere implements Tractor {
public void pullStuff() { System.out.print("pulling "); }
}
class LT255 implements Tractor extends Deere {
public void pullStuff() { System.out.print("pulling harder "); }
}
public class LT155 extends Deere implements Tractor, Engine { }
```

What is the result? (Choose all that apply.)

A. Compilation succeeds.
B. Compilation fails because of error(s) in Tractor.
C. Compilation fails because of error(s) in Deere.
D. Compilation fails because of error(s) in LT255.
E. Compilation fails because of error(s) in LT155.