Given:

```
class Atom{
Atom(){System.out.print(”atom”);}
}
class Rock extends Atom{
Rock(String type){System.out.print(type);}
}
public class Mountain extends Rock{
Mountain(){
super(“granite”);
new Rock(“granite”);
}
public static void main(String [] a){ new Mountain();}
}
```

What is the result?

A. Compilation fails
B. atom granite
C. granite granite
D. atom granite granite
E. An exception throw at runtime
F. atom granite atom granite

F

Which statements are true?(Choose three)

```
10. interface Foo{
11. int bar();
12. }
13.
14. public class Beta{
15.
16. class A implements Foo{
17. public int bar(){ return 1; }
18. }
19.
20. public int fubar(Foo foo){ return foo.bar(); }
21.
22. public void testFoo(){
23.
24. class A implements Foo{
25. public int bar(){ return 2; }
26. }
27.
28. System.out.println(fubar(new A()));
29. }
30.
31. public static void main(String [] argv){
32. new Beta().testFoo();
33. }
34. }
```

A. Compilation fails.
B. The code compiles and the output is 2.
C. If lines 16, 17 and 18 were removed, compilation would fail.
D. If lines 24, 25 and 26 were removed, compilation would fail.
E. If lines 16, 17 and 18 were removed, the code would compile and the output would be 2.
F. If lines 24, 25 and 26 were removed the code would compile and the output would be 1.

El código es un ejemplo de uso de una anonimous inner class. B, E, F

Given:

```
10. class Line{
11. public class Point{public int x, y;}
12. public Point getPoint(){return new Point();}
13. }
14. class Triangle{
15. public Triangle(){
16. //insert code here
17. }
18. }
```

Which code, inserted at line 16, correctly retrieves a local instance of a Point object?

A. Point p = Line.getPoint();
B. Line.Point p = Line.getPoint();
C. Point p = (new Line()).getPoint();
D. Line.Point p = (new Line()).getPoint();

Ejemplo de uso de una inner class.D

Given:

```
10. class Line{
11. public static class Point{}
12. }
13.
14. class Triangle {
15. // insert code here
16. }
```

Which code, inserted at line 15, creates an instance of the Point class defined in Line?

A. Point p = new Point();
B. Line Point p = new Line.Point();
C. The point class cannot be instantiated at line 15.
D. Line l = new Line(); Point p = new Point();

B

Given:

```
11. class Alpha{
12. public void foo(){System.out.print("Afoo");}
13. }
14. public class Beta extends Alpha{
15. public void foo(){System.out.print("Bfoo")}
16. public static void main(String [] args){
17. Alpha a = new Beta();
18. Beta b = (Beta)a;
19. a.foo();
20. b.foo();
21. }
22. }
```

What is the result?

A. Afoo Afoo
B. Afoo Bfoo
C. Bfoo Afoo
D. Bfoo Bfoo
E. Compilation fails
F. An exception is thrown at runtime

El casting castea también el objeto a.D

Which statement is true about the classes and interfaces in the exhibit?

```
1. public interface A{
```

```
2. public void doSomething(String thing)
3. }

1. public class Aimpl implements A{
2. public void doSomething(String msg){}
3. }

1. public class B{
2. public A doit(){
3. //more code here
4. }
5.
6. public String execute(){
7. // more code here
8. }
9. }

1. public class C extends B{
2. public Aimpl doit(){
3. //more code here
4. }
5.
6. public Object execute(){
7. // more code here
8. }
9. }
```

A. Compilation will succeed for all classes and interfaces.
B. Compilation of class C will fail because of an error in line 2.
C. Compilation of class C will fail because of an error in line 6.
D. Compilation of class Aimpl will fail because of an error in line 2.

En una sobreescritura el tipo de dato devuelto no debe cambiar. C

Given a class Repetition:

```
1. package util;
2.
3. public class Repetition{
4. public static String twice(String s){return s + s;}
5.}
```

and given another class Demo:

```
1 //insert code here
2.
3. public class Demo{
4. public static void main(String args[]){
5. System.out.println(twice("pizza"));
6. }
7. }
```

Which code should be inserted at line 1 of Demo.java to compile and run Demo to print "pizzapizza"?
A. import utils.*;
B. static import utils.*;
C. import utils.Repetition.*;
D. static import utils.Repetition.*;
E. import utils.Repetition.twice();
F. import static utils.Repetition.twice;
G. static import utils.Repetition.twice;

Es posible importar métodos estáticos de otras clases (aunque no es posible importar métodos no estáticos, para hacerlo necesitaríamos una instancia). Para importar un método estático, la importación deberá ir precedida del modificador static. F

Given classes defined in two differents file:

```
1. package util;
2. public class BitUtils{
3. private static void process(byte[] b){}
4. }
1. package class SomeApp{
2. public class SomeApp{
3. public static void main(String args[]){
4. byte[]bytes = new byte[256];
5. //insert code here
6. }
7. }
```

What is required at line 5 in class SomeApp to use the proccess method of BitUtils?

A. process(bytes);
B. BitUtils.process(bytes);
C. app.BitUtils.process(bytes);
D. util.BitUtils.process(bytes);
E. import util.BitUtils.*; process(bytes);
F. SomeApp cannot use the process method in BitUtils

F

Given classes defined in two different files:

```
1. package util;
2. public class BitUtils{
3. public static void process (byte [] b){/* more code here */}
4. }
1. package app;
2. public class SomeApp{
3. public static void main(String [] args){
4. byte [] bytes = new byte[256];
5. // insert code here
6. }
7. }
```

A. process(bytes);
B. BitUtils.process(bytes);
C. util.BitUtils.process(bytes);
D. SomeApp cannot use methods in BitUtils
E. import util.BitUtils.*; process(bytes);

C

Given:

```
11. public class ItemTest{
12. private final int id;
13. public ItemTest(int id){this.id = id; }
14. public void updateId(int newId){id = newId;}
15.
16. public static void main(String [] args){
17. ItemTest fa = new ItemTest(42);
18. fa.updateId(69);
19. System.out.print(fa.id);
20. }
21. }
```

What is the result?

A. Compilation fails.

B. An exception is thrown at runtime.
C. The attribute id in the ItemTest object remains unchanged.
D. The attribute id in the ItemTest object is modified to the new value.
E. A new ItemTest object is created with the preferred value in the id attribute.

A

Given:
```
1. interface A{public void aMethod();}
2. interface B{public void bMethod();}
3. interface C extends A, B{public void cMethod();}
4. class D implements B{
5. public void bMethod(){}
6.}
7. class E extends D implements C{
8. public void aMethod(){}
9. public void bMethod(){}
10. public void cMethod(){}s
11. }
```
What is the result?
A. Compilation fails because of an error in line 3.
B. Compilation fails because of an error in line 7.
C. Compilation fails because of an error in line 9.
D. If you define D e = new E(), then e.bMethod() invokes the version of bMethod() defined in Line 5.
E. If you define D e = (D)(new E()), then e.bMethod() invokes the version of bMethod() defined in Line 5.
F. If you define D e = (D)(new E()), then e.bMethod() invokes the version of bMethod() defined in Line 9.

F

Given that: Gadget has-a Sprocket and has-a Spring and Gadget is-a Widget and Widget has-a Sprocket . Which two code fragments represent these relationships?(Choose two.)
A.
```
class Widget{Sprocket s;}
class Gadget extends Widget{Spring s;}
```
B.
```
class Widget{}
class Gadget extends Widget{Spring s1; Sprocket s2;}
```
C.
```
class Widget{Sprocket s1; Spring s2;}
class Gadget extends Widget{}
```
D.
```
class Gadget{Spring s;}
class Widget extends Gadget{Sprocket s;}
```
E.
```
class Gadget{}
class Widget extends Gadget{Sprocket s1; Spring s2;}
```
F.
```
class Gadget{Spring s1; Sprocket s2;}
class Widget extends Gadget{}
```

A, C

Which Man class properly represents the relationships “Man has a best friend who is a Dog”?
A. class Man extends Dog(){}
B. class Man implements Dog{}
C. class Man {private BestFriend dog;}
D. class Man {private Dog bestFriend; }
E. class Man {private Dog <bestFriend>;}
F. class Man {private BestFriend<dog>}

D

Given:

```
31. class Foo{
32. public int a = 3;
33. public void addFive(){a +=5; System.out.print(“f”);}
34. }
35. class Bar extends Foo{
36. public int a = 8;
37. public void addFive(){this.a +=5; System.out.print(“b ”);}
38. }
```

Invoked with: Foo f = new Bar(); f.addFive(); System.out.println(f.a);

What is the result?

A. b 3
B. b 8
C. b 13
D. f 3
E. f 8
F. f13
G. Compilation fails
H. An exception is thrown at runtime

Al estar declarando un objeto de la clase Foo, sólo tendré disponibles las propiedades y métodos de la clase Foo (si hay sobreescritura, el código de los métodos es sobreescrito).A

Given:

```
11. class Animal {public String noise(){return “peep”;}}
12. class Dog extends Animal{
13. public String noise(){return “bark”;}
14. }
15. class Cat extends Animal {
16. public String noise(){return “meow”; }
17. }...
30. Animal animal = new Dog();
31. Cat cat = (Cat)animal;
32. System.out.println(cat.noise());
```

What is the result?

A. peep
B. bark
C. meow
D. Compilation fails.
E. An exception is thrown at runtime.

Animal es un Dog, y no puedo convertir un Dog en un Cat. Estos errores de conversión son arrojados en tiempo de ejecución. E

Given:

```
1. class Super{
2. private int a;
3. protected Super(int a){this.a = a;}
4. }...
11. class Sub extends Super {
12. public Sub(int a){super(a);}
13. public Sub(){this.a = 5;}
14. }
```

Which two, independently, will allow Sub to compile? (Choose two.)

A.
Change line 2 to:
public int a;

B.

Change line 2 to:
protected int a;
C.
Change line 13 to:
public Sub(){this(5);}
D.
Change line 13 to:
public Sub(){super(5);}
E.
Change line 13 to:
public Sub(){super(a);}

C,D

Given:
1. public class Base{
2. public static final String FOO = "foo";
3. public static void main(String [] args){
4. Base b = new Base();
5. Sub s = new Sub();
6. System.out.print(Base.FOO);
7. System.out.print(Sub.FOO);
8. System.out.print(b.FOO);
9. System.out.print(s.FOO);
10. System.out.print(((Base)s).FOO);
11. }}
12. class Sub extends Base{public static final String FOO ="bar";}
What is the result?
A. foofoofoofoofoo
B. foobarfoobarbar
C. foobarfoofoofoo
D. foobarfoobarfoo
E. barbarbarbarbar
F. foofoofoobarbar
G. foofoofoobarfoo

D

Given:
1. package geometry;
2. public class Hypotenuse{
3. public InnerTriangle it = new InnerTriangle();
4. class InnerTriangle{
5. public int base;
6. public int height;
7. }
8. }
Which statement is true about the class of an object that can reference the variable base?
A. It can be any class.
B. No class has access to base.
C. The class must belong to the geometry package.
D. The class must be a subclass of the class Hypotenuse.

C

Given:
2. public class Hi{
3. void m1(){}
4. protected void() m2 {}
5. }

```
6. class Lois extends Hi{
7. // insert code here
8. }
```

Which four code fragments, inserted independently at line 7, will compile? (Choose four.)

A. public void m1(){}
B. protected void m1(){}
C. private void m1(){}
D. void m2(){}
E. public void m2(){}
F. protected void m2(){}
G. private void m2(){}

En un método sobreescrito la accesibilidad no debe ser más restrictiva que en el método original, luego la respuesta C y D son incorrectas. A, B, E, F

Given:

```
11. class A{
12. public void process(){System.out.println("A,");}
13. class B extends A{
14. public void process() throws IOException{
15. super.process();
16. System.out.print("B");
17. throw new IOException();
18. }
19. public static void main(String [] args){
20. try{new B().process();}
21. catch(IOException e){System.out.println("Exception");}
22. }
```

What is the result?

A. Exception
B. A, B, Exception
C. Compilation fails because of an error in line 20.
D. Compilation fails because of an error in line 14.
E. A NullPointerException is thrown at runtime

When you override a method, you can't start throwing any new checked exceptions that the method in the parent class didn't already throw.D

Given:

```
11. static class A{
12. void process() throws Exception{ thrown new Exception(); }
13. }
14. static class B extends A {
15. void process(){ System.out.println("B"); }
16. }
17. public static void main(String [] args){
18. new B().process();
19. }
```

What is the result?

A. B
B. The code runs with no output.
C. Compilation fails because of an error in line 12.
D. Compilation fails because of an error in line 15.
E. Compilation fails because of an error in line 18.

Llamar a un método estático a través de un objeto da un warning, pero el çodigo compila. A

Given:

```
10. interface A{void x();}
11. class B implements A{public void x() {}public void y(){}}
```

```
12. class C extends B{public void x(){}}
```
And:
```
20. java.util.List<A> list = new java.util.ArrayList<A>();
21. list.add(new B());
22. list.add(new C());
23. for(A a: list){
24. a.x();
25. a.y();
26. }
```
What is the result?
A. The code runs with no output.
B. An exception is thrown at runtime.
C. Compilation fails because of an error in line 20.
D. Compilation fails because of an error in line 21.
E. Compilation fails because of an error in line 23.
F. Compilation fails because of an error in line 25.

La lista sólo puede tener interfaces de A, y la interfaz A no tiene el método y().F

Given:
```
10. public class Hello{
11. String title;
12. int value;
13. public Hello(){
14. title += "World";
15. }
16. public Hello(int value){
17. this.value = value;
18. title = "Hello";
19. Hello();
20. }
21. }
```
and:
```
30. Hello c = new Hello(5);
31. System.out.println(c.title);
```
What is the result?
A. Hello
B. Hello World
C. Compilation fails.
D. Hello World 5
E. The code runs with no output.
F. An exception is thrown at runtime.

Hello() es un constructor, no un método, por tanto no lo puedo llamar explícitamente. C

Given:
```
1. public class Target{
2. private int i = 0;
3. public int addOne{
4. return ++i;
5. }
6. }
```
And:
```
1. public class Client{
2. public static void main (String [] args){
3. System.out.println(new Target().addOne());
4. }
5. }
```

Which change can you make to Target without affecting Client?
A. Line 4 of class Target can be changed to return i++.
B. Line 2 of class Target can be changed to private int i = 1;
C. Line 3 of class Target can be changed to private int addOne(){
D. Line 2 of class Target can be changed to private Integer i = 0;

D

Given:
1. public class Blip{
2. protected int blipvert(int x){return 0; }
3. }
4. class Vert extends Blip{
5. // insert code here
6. }
Which five methods, inserted independently at line 5, will compile?(Choose five.)
A. public int blipvert(int x){return 0;}
B. private int blipbert(int x){return 0;}
C. private int blipvert(long x){return 0;}
D. protected long blipvert(int x){return 0;}
E. protected int blipvert(long x){return 0;}
F. protected long blipvert(long x){return 0;}
G. protexted long blipvert(int x, int y){return 0;}

El método sobreescritor no puede tener un modificador más restrictivo que el método sobreescrito. En una sobreescritura el tipo de dato devuelto no puede cambiar.A, C, E, F, G

Given:
1. class Pizza{
2. java.util.ArrayList toppings;
3. public final void addTopping(String topping){
4. toppings.add(topping);
5. }
6. }
7. public class PepperoniPizza extends Pizza{
8. public void addTopping(String topping){
9. System.out.println("Cannot add Toppings");
10. }
11. public static void main(String [] args){
12. Pizza pizza = new PepperoniPizza();
13. pizza.addTopping("Mushrooms");
14. }
15. }
What is the result?
A. Compilation fails.
B. Cannot add Toppings.
C. The code runs with no output.
D. A NullPointerException is thrown in Line 4.

No es posible sobreescribir un método que tenga el modificador final.A

Given:
11. class ClassA{}
12. class ClassB extends ClassA{}
13. class ClassC extends ClassA {}
and:
21. ClassA p0 = new ClassA();
22. ClassB p1 = new ClassB();
23. ClassC p2 = new ClassC();
24. ClassA p3 = new ClassB();

```
25. ClassA p4 = new ClassC();
```

Which three are valid? (Choose three.)
A. p0 = p1;
B. p1 = p2;
C. p2 = p4;
D. p2 = (ClassC)p1;
E. p1 = (ClassB)p3;
F. p2 = (ClassC)p4;

A, E, F

Given two files, GrizzlyBear.java and Salmon.java:

```
1. package animals.mammals;
2.
3. public class GrizzlyBear extends Bear{
4. void hunt(){
5. Salmon s = findSalmon();
6. s.consume();
7. }
8. }
```

```
1. package animals.fish;
2.
3. public class Salmon extends Fish{
4. public void consume(){ /* do stuff */ }
5. }
```

If both classes are in the correct directories for their packages, and the Mammal class correctly defines the findSalmon() method, which change allows this code to compile?
A. add import animals.mammals.*; at line 2 in Salmon.java
B. add import animals.fish.*; at line 2 in GrizzlyBear.java
C. add import animals.fish.Salmon.*; at line 2 in Salmon.java
D. add import animals mammals.GrizzlyBear.*; at line 2 in Salmon.java

B

Which three code fragments, added individually at line 29, produce the output 100?(Choose three.)

```
10. class Inner{
11. private int x;
12. public void setX(int x){this.x = x;}
13. public int getX(){return x;}
14. }
15.
16. class Outer{
17. private Inner y;
18. public void setY(Inner y){this.y = y;}
19. public Inner getY(){return y; }
20. }
21.
22. public class Gamma{
23. public static void main(String [] args){
24. Outer o = new Outer();
25. Inner i = Inner();
26. int n = 10;
27. i.setX(n);
28. o.setY(i);
29. // insert code here
30. System.out.println(o.getY().getX());
31. }
32. }
```

A. n = 100;
B. i.setX(100);
C. o.getY().setX(100);
D. i = new Inner(); i.setX(100);
E. o.setY(i); i = new Inner(); i.setX(100);
F. i = new Inner(); i.setX(100); o.setY(i);

o has-a i.B, C, F

Given:

```
11. interface DeclareStuff{
12. public static final int EASY = 3;
13. void doStuff(int i); }
14. public class TestDeclare implements DeclareStuff{
15. public static void main(String [] args){
16. int x = 5;
17. new TestDeclare().doStuff(++x);
18. }
19. void doStuff(int s){
20. s += EASY +++ ;
21. System.out.println("s " + s);
22. }
23. }
```

What is the result?
A. s14
B. s16
C. s10
D. Compilation fails.
E. An exception is thrown at runtime

No es posible modificar una variable con el modificador de acceso final. D

Given:

```
1. public class Plant{
2. private String name;
3. public Plant(String name){this.name = name;}
4. public String getName(){return name;}
5. }
1. public class Tree extends Plant {
2. public void growFruit(){}
3. public void dropLeaves(){}
4. }
```

Which statement is true?
A. The code will compile without changes.
B, The code will compile if public Tree(){Plant(); } is added to the Tree class.
C. The code will compile if public Plant(){Tree();} is added to the Plant class.
D. The code will compile if public Plant(){this("fem");} is added to the Plant class.
E. The code will compile if public Plant(){Plant("fem"); } is added to the Plant class.

La clase Tree tratará de llamar al constructor vacío de la clase Plant. Por tanto para que el código funcione, la clase Plant deberá tener un constructor vacío. Desde este contructor llamaremos al constructor con argumento String de la propia clase Plant usando this("fem"). D

Given:

```
11. public interface A111{
12. String s = "yo";
13. public void method1();
14. }
17. interface B{}
20. interface C extends A111, B{
```

21. public void method1();
22. public void method1(int x);
23. }
What is the result?
A. Compilation succeds.
B. Compilation fails due to multiple errors.
C. Compilation fails due to an error only on line 20.
D. Compilation fails due to an error only on line 21.
E. Compilation fails due to an error only on line 22.
F. Compilation fails due to an error only on line 12.

A

Given:
1. interface TestA{String toString(); }
2. public class Test{
3. public static void main(String [] args){
4. System.out.println(new TestA(){
5. public String toString(){return “test”;}
6. });
7. }
8. }
What is the result?
A. test
B. null
C. An exception is thrown at runtime.
D. Compilation fails because of an error in line 1.
E. Compilation fails because of an error in line 4.
F. Compilation fails because of an error in line 5.

No estoy instanciando una interfaz (eso no es posible). Estoy instanciando una anonimous inner class. A

Given:
11. class Alpha{
12. public void foo(){System.out.print(“Afoo”);}
13. }
14. public class Beta extends Alpha{
15. public void foo(){System.out. print(“Bfoo”);}
16. public static void main(String [] args){
17. Alpha a = new Beta();
18. Beta b = (Beta)a;
19. a.foo();
20. b.foo();
21. }
22. }
What is the result?
A. Afoo Afoo
B. Afoo Bfoo
C. Bfoo Afoo
D. Bfoo Bfoo
E. Compilation fails.
F. An exception is thrown at runtime.

D

Given:
10. abstract public class Employee{
11. protected abstract double getSalesAmount();
12. public double getCommision(){
13. return getSalesAmoun()*0.15;

```
14. }
15. }
16. class Sales extends Employee{
17. // insert method here
18. }
```

Which two methods, inserted independently at line 17, correctly complete the Sales class?(Choose two.)

A. double getSalesAmount(){return 1230.45}
B. public double getSalesAmount(){return 1230.45}
C. private double getSalesAmount(){return 1230.45}
D. protected double getSalesAmount(){return 1230.45}

En una sobreescritura el tipo de dato devuelto debe coincidir con el tipo de dato devuelto por el método que estoy sobreescribiendo y el método sobreescritor no puede ser más reestrictivo que el método sobreescrito. B, D

What is the result?

```
11. class Person{
12. String name = “No name”;
13. public Person(String nm){name = nm;}
14. }
15.
16. class Employee extends Person {
17. String empID = “0000”;
18. public Employee(String id){empID = id;}
19. }
20.
21. public class EmployeeTest{
22. public static void main(String [] args){
23. Employee e = new Employee(“4321”);
24. System.out.println(e.empID);
25. }
26. }
```

A. 4321
B. 0000
C. An excetpion is thrown at runtime.
D. Compilation fails because of an error in line 18,

La clase Employee llamará al constructor vacío de la clase Person, pero la clase Person no tiene constructor vacío. D

Given:

```
11. class X{public void foo(); {System.out.print(“X ”); }}
12.
13. public class SubB extends X{
14. public void foo() throws RuntimeException{
15. super.foo();
16. if(true)throw new RuntimeException();
17. System.out.print(“B”);
18. }
19. public static void main(String args[]){
20. new SubB().foo();
21. }
22. }
```

What is the result?

A. X, followed by an Exception.
B. No output, and an Exception is thrown.
C. Compilation fails due to an error on line 14.
D. Compilation fails due to an error on line 16.
E. Compilation fails due to an error on line 17.
F. X, followed by an Exception, followed by B.

A

Given:

```
2. public class Hi{
3. void m1();
4. protected void() m2{}
5. }
6. class Lois extends Hi{
7. //insert code here
8. }
```

Which four code fragments, inserted independently at line 7, will compile?(Choose four.)

A. public void m1(){}
B. protected void m1(){}
C. private void m1(){}
D. void m2(){}
E. public void m2(){}
F. protected void m2(){}
G. private void m2(){}

El modificador del método sobreescritor no puede ser menos genérico que el modificador del método sobreescrito. A, B, E, F

Given:

```
10. public class Hello{
11. String title;
12. int value;
13. public Hello(){
14. title += “World”;
15. }
16. public Hello(int value){
17. this.value = value;
18. title = “Hello”;
19. Hello();
20. }
21. }
```

and:

```
30. Hello c = new Hello(5);
31. System.out.println(c.title);
```

What is the result?

A. Hello
B. Hello World
C. Compilation fails.
D. Hello World 5.
E. The code runs with no output.
F. An exception is thrown at runtime.

title es una variable miembro inicializada a null por defecto. C

Given:

```
3. class Employee{
4. String name; double baseSalary;
5. Employee(String name, double baseSalary){
6. this.name = name;
7. this.baseSalary = baseSalary;
8. }
9. }
10. public class SalesPerson extends Employee{
11. double comission;
12. public SalesPerson(String name, double baseSalary, double commission){
```

```
13. // insert code here
14. }
15. }
```

Which two code fragments, inserted independently at line 13, will compile?(Choose two.)

A. super(name, baseSalary);

B. this.commision = commision;

C.
```
super();
this.commission = commission();
```
D.
```
this.commission = commission;
super();
```
E.
```
super(name, baseSalary);
this.commission = commission;
```
F.
```
this.commission  = commission;
super(name, baseSalary);
```
G.
```
super(name, baseSalary, commission);
```

Desde la clase hija se llamará al constructor de la clase padre, salvo que la clase padre tenga un constructor con argumentos, en cuyo caso habrá que llamar a dicho constructor explícitamente. A, E

Given:
```
1.class ClassA{
2. public int numberOfInstances;
3. protected ClassA(int numberOfInstances){
4. this.numberOfInstances = numberOfInstances;
5. }
6. }
7. public class ExtendedA extends ClassA{
8. private Extended(int numberOfInstances){
9. super(numberOfInstances);
10. }
11. public static void main(String [] args){
12. ExtendedA ext = new ExtendedA(420);
13. System.out.print(ext.numberOfInstances);
14. }
15. }
```

Which statament is true?

A. 420 is the output.

B. An exception is thrown at runtime.

C. All constructors must be declared public.

D. Constructors CANNOT use the private modifier.

E. Constructors CANNOT use the protected modifer.

A

Given:
```
5. class Building{}
6. public class Barn extends Building{
7. public static void main(String [] args){
8. Building build1 = new Building();
9. Barn barn1 = new Barn();
10. Barn barn2 = new Barn();
11. Object obj1 = (Object) build1;
12. String str1 = (String) build1;
13. Building build2 = (Building) barn1;
14. }
15. }
```

Which is true?

A. If line 10 is removed, the compilation succeeds.
B. If line 11 is removed, the compilation succeeds.
C. If line 12 is removed, the compilation succeeds.
D. If line 13 is removed, the compilation succeeds.
E. More than one line must be removed for compilation to succeed.

No es posible convertir un tipo de dato más genérico a uno más específico. C

Given:

```
3. interface Fish{}
4. class Perch implements Fish{}
5. class Walleye extends Perch{}
6. class Bluegill{}
7. public class Fisherman{
8. public static void main(String [] args){
9. Fish f = new Walleye();
10. Walleye w = new Walleye();
11. Bluegill b = new Bluegill();
12. if( f instanceof Perch) System.out.println("f-p");
13. if( w instanceof Fish) System.out.println("w-f");
14. if( b instanceof Fish) System.out.println("b-f");
15. }
16. }
```

What is the result?

A. w-f
B. f-p w-f
C. w-f b-f
D. f-p w-f b-f
E. Compilation fails.
F. An exception is thrown at runtime.

B

Given:

```
10. interface Foo{}
11. class Alpha implements Foo{}
12. class Beta extends Alpha{}
13. class Delta extends Beta{
14. public static void main(String [] args){
15. Beta x = new Beta();
16. // insert code here
17. }
18. }
```

Which code, inserted at line 16, will cause a java.lang.ClassCastException?

A. Alpha a = x;
B. Foo f = (Delta)x;
C. Foo f = (Alpha)x;
D. Beta b = (Beta)(Alpha)x;

No es posible convertir un tipo de dato más genérico en un tipo de dato más específico. B

Given:

```
10.class One{
11. void foo() {}
12. {
13. class Two extends One{
```

```
14. // insert method here
15. }
```

Which three methods, inserted individually at line 14, will correctly complete class Two? (Choose three)

A. int foo (){/* more code here*/}

B. void foo(){/* more code here*/}

C. public void foo(){/* more code here*/}

D. private void foo(){/*more code here*/}

E. protected void foo(){/* more code here*/}

El modificador de acceso del método sobreescritor no puede ser más reestrictivo que el modificador de acceso del método sobreescrito. Además, el tipo de dato devuelto debe ser el mismo para ambos. B, C, E

Which statement is true about the classes and interfaces in the exhibit?

```
1. public interface A{
2. public void doSomething(String thing);
3. }

1. public class Aimpl implements A{
2. public doSomething(String msg){}
3. }

1. public class B{
2. public A doit(){
3. //insert code here
4. }
5.
6. public String execute(){
7. // more code here
8. }
9. }

1. public class C extends B{
2. public Aimpl doit(){
3. // more code here
4. }
5.
6. public Object execute(){
7. // more code here
8. }
9. }
```

A. Compilation will succeed for all classes and interfaces.

B. Compilation of class C will fail because of an error in line 2.

C. Compilation of class C will fail because of an error in line 6.

D. Compilation of class Aimpl will fail because of an error in line 2.

El tipo de dato devuelto por un método sobreescrito debe coincidir con el tipo de dato devuelto por el método sobreescribidor. C

Given:

```
11. public interface A{public void m1();}
12.
13.class B implements A{}
14. class C implements A{public void m1(){}}
15. class D implements A{public void m1(int x){}}
16. abstract class E implements A{}
17. abstract class F implements A{public void m1(){}}
18. abstract class G implements A{public void m1(int x){}}
```

What is the result?

A. Compilation succeds.
B, Exactly one class does NOT compile.
C, Exactly two classes do NOT compile.
D. Exactly four classes do NOT compile.

C

Given:
21. class Money{
22. private String country = “Canada”;
23. public String getC(){ return country;}
24. }
25. class Yen extends Money{
26. public String getC();{return super.country;}
27. }
28. public class Euro extends Money{
29. public String getC(int x){return super.getC(); }
30. public static void main(String [] args){
31. System.out.print(new Yen().getC() + “” + new Euro().getC());
32. }
33. }
What is the result?
A. Canada
B. null Canada
C. Canada null
D. Canada Canada
E. Compilation fails due to an error on line 26.
F. Compilation fails due to an error on line 29.

E

Which three statements are true? (Choose three.)
A. A final method in class X can be abstract if and only if X is abstract.
B. A protected method in class X can be overriden by any subclass of X.
C. A private static method can be called only within other static methods in class X.
D. A non-static public final method in class X can be overriden in any subclass of X.
E. A public static method in class X can be called by a sublcass of X without explicity referencing the class X.
F. A method with the same signature as a private final method in class X can be implemented in a subclass of X.
G. A protected method in class X can be overriden by a subclass of X only if the sublclass is in the same package as X.

B, E, F

What two must the programmer do to correct the compilation error? (Choose two.)
1. public class Car{
2. private int wheelCount;
3. private String vin;
4. public Car(String vin){
5. this.vin = vin;
6. this.wheelCount = 4;
7. }
8. public String drive(){
9. return “zoom-zoom”;
10. }
11. public String getInfo(){
12. return “VIN : ” + vin + “wheels: ” + wheelCount;
13. }
14. }
And:
1. public class MeGo extends Car{
2. public MeGo(String vin){

```
3. this.wheelCount = 3;
4. }
5. }
```
A. insert a call to this() in the Car constructor
B. insert a call to this() int the MeGo constructor
C. insert a call super() in the MeGo constructor
D. insert a call to super(vin) in the MeGo constructor
E. change the wheelCount variable in Car to protected
F. change line 3 in the MeGo class to super.wheelCount = 3;

D, E

Given:
```
5. class Thingy{Meter m = new Meter();}
6. class Component {void go(){System.out.print("c");}}
7. class Meter extends Component{void go(){System.out.print("m");}}
8.
9. class Deluxe Thingy extends Thingy{
10. public static void main(String [] args){
11. DeluxeThingy dt = new DeluxeThingy();
12. dt.m.go();
13. Thingy t = new DeluxeThingy();
14. t.m.go();
15. }
16. }
```
Which two are true? (Choose two.)
A. The output is mm.
B. The output is mc.
C. Component is-a Meter.
D. Component has-a Meter.
E. DeluxeThingy is-a Component.
F. DeluxeThingy has-a Component.

A, F

Given:
```
10. interface Jumper{public void jump(); }
20. class Animal{}...
30. class Dog extends Animal{
31. Tail tail; 32 }...
40. class Beagle extends Dog implements Jumper{
41. public void jump(){}
42. }...
50. class Cat implements Jumper{
51. public void jump(){}
52. }
```
Which three are true?(Choose three.)
A. Cat is-a Animal
B. Cat is-a Jumper
C. Dog is-a Animal
D. Dog is-a Jumper
E. Cat has-a Animal
F. Beagle has-a Tail
G. Beagle has-a Jumper

B, C, F

What is the result?
```
1. public class SimpleCal{
2. public int value;
```

```
3. public void calculate(){value +=7; }
```
And:
```
1. public class MultiCalc extends SimpleCalc{
2. public void calculate(){value -= 3; }
3. public void calculate(int multiplier);
4. calculate();
5. super.calculate();
6. value *= multiplier;
7. }
8. public static void main(String [] args){
9. MultiCalc calculator = new MultiCalc();
10. calculator.calculate(2);
11. System.out.println("Value is" + calculator.value);
12. }
13. }
```
A. Value is: 8
B. Compilation fails.
C. Value is 12
D. Value is -12
E. The code runs with no output.
F. An exception is thrown at runtime.

Los valores que toma value son -3, 4 y 8. A

Given:
```
1. interface DoStuff2{
2. float getRange(int low, int high); }
3.
4. interface DoMore{
5. float getAvg(int a, int b, int c); }
6.
7. abstract class DoAbstract implements DoStuff2, DoMore{}
8.
9. class DoStuff implements DoStuff2{
10. public float getRange(int x, int y ){return 3.14f;}}
11.
12. interface DoAll extends DoMore{
13. float getAvg(int a, int b, int c, int d); }
```
What is the result?
A. The file will compile withou error.
B. Compilation fails. Only line 7 contains error.
C. Compilation fails. Only line 12 contains error.
D. Compilation fails. Only line 13 contains an error.
E. Compilation fails. Only line 7 and 12 contain errors.
F. Compilation fails. Only line 7 and 13 contain errors.
G. Compilation fails. Lines 7, 12, and 13 contain errors.

A

Given:
```
11. public class Person{
12. private String name;
13. public Person(String name){
14. this.name = name;
15. }
16. public boolean equals(Object o){
17. if(!(o instanceof Person))retur n false;
18. Person p = (Person) o;
19. return p.name.equals(this.name);
20. }
21. }
```

Which statement is true?

A. Compilation fails because the hashCode method is not overriding.

B. A HashSet could contain multiple Person objects with the same name.

C. All Person objects will have the same hash code because the hashCode method is not overriden.

D. If a HashSet contains more than one Person object with name=”Fred”, then removing another Person, also with name=”Fred”, will remove them all.

B

Given:

```
10. interface Data { public void load(); }
11. abstract class Info{public abstract void load(); }
```

Which class correctly uses the Data interface and Info class?

A.

```
public class Employee extends Info implements Data{
public void load(){/* do something */}
```

B.

```
public class Employee implements Info extends Data{
public void load(){ /* do something */}
}
```

C.

```
public class Employee extends Info implements Data{
public void load(){/*do something*/}
public void Info.load(){/*do something*/}
}
```

D.

```
public class Employee implements Info extends Data{
public void Data load(){/*do something*/}
public void load(){/*do something*/}
}
```

E.

```
public class Employee implements Info extends Data{
public void load(){/*do something*/}
public void Info.load(){/*do something*/}
}
```

F.

```
public class Employee extends Info implements Data{
public void Data.load(){/*do something*/}
public void Info.load(){/*do something*/}
}
```

A

Given:

```
21. abstract class C1{
22. public C1(){System.out.print(1);}
23. }
24. class C2 extends C1{
25. public C2(){System.out.print(2); }
26. }
27. class C3 extends C2 {
28. public C3(){System.out.println(3); }
29. }
30. public class Ctest{
31. public static void main(String [] args){new C3(); }
32. }
```
What is the result?
A. 3
B. 23
C. 32
D. 123
E. 321
F. Compilation fails.
G. An exception is thrown at runtime.

D

Given:
```
10. class One{
11. public One foo(){return this;}
12. }
13. class Two extends One{
14. public One foo(){return this;}
15. }
16. class Three extends Two{
17. // insert method here
18. }
```
Which two methods, inserted individually, correctly complete the Three class?(Choose two.)
A. public void foo(){}
B. public int foo(){return 3;}
C. public Two foo(){return this;}
D. public One foo(){ return this; }
E. public Object foo(){return this;}

C, D

Given:
```
5. class Payload{
6. private int weight;
7. public Payload(int w){weight = w; }
8. public void setWeight(int w){weight = w;}
9. public String toString(){return Integer.toString(weight);}
10. }
11. public class TestPayload{
12. static void changePayload(Payload p){/* insert code*/}
13. public static void main(String [] args){
14. Pauload p = new Payload(200);
15. p.setWeight(1024);
16. changePayload(p);
17. System.out.println(“p is ” + p);
18. }}
```
Which code fragment, inserted at the end of line 12, produces the output p is 420?
A. p.setWeight(420);
B. p.changePayload(420);
C. p = new Payload(420);
D. Payload.setWeight(420);
E. p = Payload.setWeight(420);

A

Given:
```
1. interface DoStuff2{
2. float getRange(int low, int high); }
3.
4. interface DoMore{
```

```
5. float getAvg(int a, int b, int c); }
6.
7. abstract class DoAbstract implements DoStuff2, DoMore{}
8.
9. class DoStuff implements DoStuff2{
10. public float getRange(int x, int y){return 3.14f}
11.
12. interface DoAll extends DoMore{
13. float getAvg(int a, int b, int c, int d); }
```

What is the result?
A. The file will compile without error.
B. Compilation fails. Only line 7 contains an error.
C. Compilation fails. Only line 12 contains an error.
D. Compilation fails. Only line 13 contains an error.
E. Compilation fails. Only lines 7 and 12 contain errors.
F. Compilation fails. Only lines 7 and 13 contain errors.
G. Compilation fails. Lines 7, 12 and 13 contain errors.

A

Given:

```
1. class TestA{
2. public void start(){System.out.println("TestA"); }
3. }
4. public class TestB extends TestA{
5. public void start(){System.out.println("TestB"); }
6. public static void main(String [] args){
7. ((TestA)new TestB()).start();
8. }
9. }
```

What is the result?
A. TestA
B. TestB
C. Compilation fails.
D. An exception is thrown at runtime.

B

Given:

```
11. public abstract class Shape{
12. private int x;
13. private int y;
14. public abstract void draw();
15. public void setAnchor(int x, int y){
16. this.x = x;
17. this.y = y;
18. }
19. }
```

Which two classes use the Shape class correctly?(Choose two.)
A.

```
public class Circle implements Shape{
private int radius;
}
```

B.

```
public abstract class Circle extends Shape{
private int radius;
}
```

C.

public class Circle extends Shape{
private int radius;
public void draw();
}
D.
public abstract class Circle implements Shape{
private int radius;
public void draw();
}
E.
public class Circle extends Shape{
private int radius;
public void draw(){ /* code here*/}
F.
public abstract class Circle implements Shape{
private int radius;
public void draw(){ /* code here*/}

Una clase, aunque sea abstracta, no se implementa, se hereda. B, E

Given:
11. abstract class Vehicle {public int speedIO{return 0;}}
12. class Car extends Vehicle{public int speed(){return 60;}}
13. class RaceCar extends Car{public int speed(){return 150; }...
21. RaceCar racer = new RaceCar();
22. Car car = new RaceCart();
23. Vehicle vehicle = new RaceCar();
24. System.out.println(racer.speed() + "," + car.speed())
25. + "," + vehicle.speed());
What is the result?
A. 0, 0, 0
B. 150, 60, 0
C. Compilation fails.
D. 150. 150, 150, 150
E. An exception is thrown at runtime.

D

Given:
11. class Mammal{}
12.
13. class Raccoon extends Mammal{
14. Mammal m = new Mammal();
15. }
16.
17. class BabyRaccoon extends Mammal{}
Which four statements are true? (Choose four.)
A. Raccoon is-a Mammal.
B. Raccoon has-a Mammal.
C. BabyRaccoon is-a Mammal.
D. BabyRaccoon is-a Raccoon.
E. BabyRaccoon has a Mammal.
F. BabyRaccoon is-a BabyRaccoon.

A, B, C, F

Given:
10. public class SuperCalc{
11. protected static int multiply(int a, int b){return a*b;}
12. }

and:

```
20. public class SubCalc extends SuperCalc{
21. public static int multiply(int a, int b){
22. int c = super.multiply(a, b);
23. return c;
24. }
25. }
```

and:

```
30. SubCalc sc = new SubCalc();
31. System.out.println(sc.multiply(3, 4));
32. System.out.println(SubCalc.multiply(2, 2));
```

What is the result?

A.
12
4

B. The code runs with no output.

C. An exception is thrown at runtime.

D. Compilation fails because of an error in line 21.

E. Compilation fails because of an error in line 22.

F. Compilation fails because of an error in line 31.

No puedo usar super en un contexto estático. E