1.- Given

```
public static void main(String[] args) {
}
int[][] array2D = {{0,1,2}, {3,4,5,6}}; System.out.print(array2D[0].length + "");
System.out.print(array2D [1].getClass().isArray() + ""); System.out.print(array2D[0][1]);
```

What is the result?

3false3
3false1
2false1
3true1
2true3

- array2D[0].length: La primera submatriz es {0, 1, 2}, que tiene 3 elementos.
- array2D[1].getClass().isArray(): array2D[1] es otra submatriz {3, 4, 5, 6}, y su clase es de tipo array, por lo tanto devuelve true.
- array2D[0][1]: El segundo elemento de la primera submatriz es 1.

Resultado correcto: 3true1

2.- Which two statments are true?

An interface CANNOT be extended by another interface.

An abstract class can be extended by a concrete class.

An abstract class CANNOT be extended by an abstract class.

An interface can be extended by an abstract class.

An abstract class can implement an interface,

An abstract class can be extended by an interface.

1. **Falso**: Una interfaz **puede** ser extendida por otra interfaz.
2. **Verdadero**: Una clase abstracta **puede** ser extendida por una clase concreta.
3. **Falso**: Una clase abstracta **puede** ser extendida por otra clase abstracta.
4. **Falso**: Una interfaz **no puede** ser extendida por una clase abstracta.
5. **Verdadero**: Una clase abstracta **puede** implementar una interfaz.
6. **Falso**: Una interfaz **no puede** extender una clase abstracta.

Respuestas correctas: An abstract class can be extended by a concrete class. y An abstract class can implement an interface.

3.- Given:

```
class Alpha{ String getType(){ return "alpha";}}

class Beta extends Alpha{String getType(){ return "beta";}}

public class Gamma extends Beta { String getType(){ return "gamma";} public static void
main(String[] args) {

Gamma g1 = (Gamma) new Alpha(); Gamma g2 = (Gamma) new Beta();
System.out.print(g1.getType()+ “ ”

+g2.getType());

}
```

What is the result?

Gamma gamma

Beta beta

Alpha beta

Compilation fails

Resultado: ClassCastException en tiempo de ejecución porque Alpha no puede ser convertido a Gamma.

Respuesta correcta: An exception is thrown at runtime.

4.- Which five methods, inserted independiently at line 5, will compile? (Choose five)

```
public class Blip{

protected int blipvert (int x) { return 0

}

class Vert extends Blip{

//insert code here

}

Private int blipvert(long x) { return 0; }

Protected int blipvert(long x) { return 0; }

Protected long blipvert(int x, int y) { return 0; }

Public int blipvert(int x) { return 0; }
```

Private int blipvert(int x) { return 0; }

Protected long blipvert(int x) { return 0; }

Protected long blipvert(long x) { return 0; }

Respuestas correctas: Private int blipvert(long x), Protected int blipvert(long x), Protected long blipvert(int x, int y), Public int blipvert(int x), Protected long blipvert(long x).

5.- Given:
1. class Super{
2. private int a;
3.
4. }
protected Super (int a){ this.a = a; }
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)
Change line 2 to: public int a;
Change line 13 to: public Sub(){ super(5);}
Change line 2 to: protected int a;
Change line 13 to: public Sub(){ this(5);}
Change line 13 to: public Sub(){ super(a);}

Respuestas correctas: Change line 13 to: public Sub() { super(5); } y Change line 2 to: protected int a;.

6.-

public abstract class Wow { private int wow;

public Wow(int wow) { this.wow = wow; }

public void wow() {} private void wowza(){}

}

It compiles without error.

It does not compile because an abstract class cannot have private methods

It does not compile because an abstract class cannot have instance variables.

It does not compile because an abstract class must have at least one abstract method.

It does not compile because an abstract class must have a constructor with no arguments.

Este código compila sin errores porque no hay ninguna regla que impida métodos privados o constructores con parámetros en clases abstractas.

Respuesta correcta: It compiles without error.

7.- What is the result?

```
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(); }
}
```

Compilation fails.

Atom granite.

Granite granite.

Atom granite granite.

An exception is thrown at runtime.

Atom granite atom granite.

Resultado: Error de compilación debido a que la clase Mountain no tiene un constructor sin argumentos para llamar al constructor de Rock.

Respuesta correcta: Compilation fails.

8.- What is printed out when the program is excuted?

```
public class MainMethod { void main() {
System.out.println("one");
}
}
static void main(String args) {
System.out.println("two");
}
public static final void main(String[] args) {
System.out.println("three");
}
void mina(Object[] args) {
System.out.println("four");
}
}
```

one

two

three

four

There is no output

El único método main válido es public static final void main(String[] args).

Resultado: three

Respuesta correcta: three

9.- What is the result?

```
class Feline {
public String type = "f";
public Feline() {
System.out.print("feline");
```

}

}

public class Cougar extends Feline (public Cougar() {

System.out.print("cougar ");

}

void go() {

type = "c":

System.out.print(this.type+ super.type);

}

}

public static void main(String[] args) { new Cougar().go():

}

Cougar c f.

Feline cougar c c.

Feline cougar c f.

Compilation fails.

Resultado: feline cougar c c porque super.type se refiere al mismo campo type modificado por this.type.

Respuesta correcta: feline cougar c c

10.- What is the result?

class Alpha { String getType() { return "alpha"; } }

class Beta extends Alpha { String getType() { return "beta";}}

public class Gamma extends Beta { String getType() { return "gamma"; } public static void main(String[] args) {

Gamma g1 = new Alpha();

Gamma g2= new Beta();

System.out.println(g1.getType()+*+g2.getType());

}

}

Alpha beta

Beta beta.

Gamma gamma.

Compilation fails.

Resultado: Error de compilación porque no se puede asignar Alpha a Gamma.

Respuesta correcta: Compilation fails.

11.- What is the result?

```
import java.util.*

public class MyScan {

public static void main(String[] args) { String in = "1 a 10. 100 1000"; Scanners = new Scanner(in);
int accum = 0;

for (int x=0; x <4; x++) { accum += s.nextInt();

}

System.out.println(accum);

}

}
```

11

111

1111

An exception is thrown at runtime.
Resultado: Excepción en tiempo de ejecución porque nextInt lanzará InputMismatchException al intentar leer a.

Respuesta correcta: An exception is thrown at runtime.

12.- What is the result?

```
public class Bees {

public static void main(String[] args) {

try{

new Bees().go();
```

System.out.println("thrown to main");

} catch (Exception e) {

}

}

synchronized void go() throws InterruptedException { Thread t1= new Thread();

t1.start();

System.out.print("1 "); t1.wait(5000);

System.out.print("2");

}

}

The program prints 1 then 2 after 5 seconds.

The program prints: 1 thrown to main.

The program prints: 1 2 thrown to main.

The program prints:1 then t1 waits for its notification

Resultado: El programa lanza IllegalMonitorStateException porque wait debe ser llamado desde un bloque sincronizado sobre el mismo monitor.

Respuesta correcta: The program prints: 1 thrown to main.

13.- Which statement is true?

class ClassA {

public int numberOfInstances;

protected ClassA(int numberOfInstances) {

this.numberOfInstances = numberOfInstances;

}

}

public class ExtendedA extends ClassA {

private ExtendedA(int numberOfInstances) {

super(numberOfInstances);

}

public static void main(String[] args) {

ExtendedA ext = new ExtendedA(420);

System.out.print(ext.numberOfInstances);

}

}

420 is the output.

An exception is thrown at runtime.

All constructors must be declared public.

Constructors CANNOT use the private modifier.

Constructors CANNOT use the protected modifier.

Resultado: El programa imprime 420 sin problemas.

Respuesta correcta: 420 is the output.

14.- The SINGLETON pattern allows:

Have a single instance of a class and this instance cannot be used by other classes

Having a single instance of a class, while allowing all classes have access to that instance.

Having a single instance of a class that can only be accessed by the first method that calls it

Respuesta correcta: Having a single instance of a class, while allowing all classes have access to that instance.

15.- What is the result?

import java.text.";

public class Align {

public static void main(String[] args) throws ParseException { String[] sa = {"111.234", "222.5678"};

NumberFormat nf = NumberFormat.getInstance(); nf.setMaximum FractionDigits(3);

for (String s: sa) { System.out.println(nf.parse(s)); }

}

}

111.234 222.567

111.234 222.568

111.234 222.5678

An exception is thrown at runtime.

Resultado: 111.234 y 222.568 debido a la configuración de MaximumFractionDigits.

Respuesta correcta: 111.234 222.568

16.- What is the result?

Given

```
public class SuperTest {
public static void main(String[] args) {
//statement1
//statement2
//statement3
}
}
class Shape {
}
public Shape() {
System.out.println("Shape: constructor");
public void foo(){
}
}
System.out.println("Shape: foo");
class Square extends Shape {
public Square() {
}
super();
public Square(String label) {
```

}

System.out.println("Square: constructor");

public void foo(){

}

super.foo();

public void foo(String label) {

System.out.println("Square: foo"):

}

}

What should statement1, statement2, and statement3, be respectively, in order to

produce the result?

Shape: constructor

Shape: foo

Square: foo

Square square = new Square ("bar"); square.foo ("bar"); square.foo();

Square square = new Square ("bar"); square.foo ("bar"); square.foo ("bar");

Square square = new Square (); square.foo (); square.foo(bar);

Square square = new Square (); square.foo (); square.foo("bar");

Square square = new Square (); square.foo (); square.foo ();

Respuesta correcta: Square square = new Square("bar"); square.foo("bar"); square.foo();

17.- Which three implementations are valid?

interface SampleCloseable {
public void close() throws java.io.IOException;

}

class Test implements SampleCloseable { public void close() throws java.io.IOException { // do

something } }

class Test implements SampleCloseable { public void close() throws Exception { // do

something } }

class Test implements SampleCloseable { public void close() throws FileNotFoundException {

// do something } }

class Test extends SampleCloseable { public void close() throws java.io.IOException { // do

something } }

class Test implements SampleCloseable { public void close() { // do something }}

Respuestas correctas: class Test implements SampleCloseable { public void close() throws java.io.IOException { // do something } } y class Test implements SampleCloseable { public void close() { // do something } }

18.- What is the result?

class MyKeys {

Integer key;

MyKeys(Integer k) { key = k; } public boolean equals(Object o) {

return ((MyKeys) o).key == this.key;

}

}

And this code snippet:

Map m = new HashMap(); MyKeys m1 = new MyKeys(1);

MyKeys m2 = new MyKeys(2);

MyKeys m3 = new MyKeys(1);

MyKeys m4 = new MyKeys (new Integer(2));

m.put(m1, "car");

m.put(m2, "boat");

m.put(m3, "plane");

m.put(m4, "bus"); System.out.print(m.size());

2

3

4

Compilation fails

Resultado: 4 ya que equals usa == en lugar de equals, así que las claves no son iguales.

Respuesta correcta: 4

19.- What value of x, y, z will produce the following result?

```
public static void main(String[] args) { // insert code here int j = 0, k = 0;
for (int i = 0; i < x; i++){ do {
k = 0;
while (k <z) {
}
}
k++;
System.out.print(k + " ");
System.out.println(" ");
}
j++;
} while (j <y);
System.out.println("---");
}}
```

int x = 4, y = 3, z = 2;

int x = 3, y = 2, z = 3;

int x = 2, y = 3, z = 3;

int x = 2, y = 3, z = 4;

int x = 4, y = 2, z= 3;

Respuesta correcta: `int x = 2, y = 3, z = 3;`

20.- Which three lines will compile and output "Right on!"?

```
13. public class Speak {
14. public static void main(String[] args) {
15. Speak speakIT = new Tell();
16. Tell telllt = new Tell();
```

```
17. speakIT.tellltLikeltls();
18. (Truth) speakIT.tellltLikeltis();
19. ((Truth) speakIT).tellltLikeltis();
20. telllt.tellltLikeltls();
21. (Truth) telllt.tellltLikeltls();
22. ((Truth) telllt).tellitLikeltis();
23.}
24. }

class Tell extends Speak implements Truth {
@Override
public void tellltLikeltis() {
System.out.println("Right on!");
}
}
interface Truth {
public void tellltLikeltis();
}
```

Line 17

Line 18

Line 19

Line 20

Line 21

Line 22

Respuestas correctas: Line 17, Line 19, Line 22.

21.- What is the result?

```
class Feline {
public String type
public Feline() {
```

=

"f";

System.out.print(s: "feline ");

}

public class Cougar extends Feline{

public Cougar() {

System.out.print(s: "cougar ");

}

void go(){

String type

=

"c";

System.out.print(this.type + super.type);

}

Run | Debug

public static void main(String[] args) {

new Cougar().go();

}}

Feline cougar c f

Feline cougar c c

Feline cougar f f

No compila

Respuesta correcta: `Feline cougar c f`

22.- ¿Cuál es el resultado?

import java.util.*;

public class App {

public static void main(String[] args) { List p = new ArrayList();

p.add(7);

```
p.add(1);

p.add(5); p.add(1);

p.remove(1);

System.out.println(p);

}

}
```

[7, 5]

[7, 1]

[7, 5, 1]

[7, 1, 5, 1]

Resultado: [7, 5, 1]. La llamada a p.remove(1) elimina el elemento en el índice 1 (el segundo elemento), no el valor 1.

Respuesta correcta: [7, 5, 1]

23.- Which five methods, inserted independiently at line 5, will compile? (Choose five)

```
public class Blip{

protected int blipvert (int x) { return 0}

}

class Vert extends Blip{

}

//insert code here

Public int blipvert(int x) { return 0; }

Protected long blipvert(int x) { return 0; }

Protected int blipvert(long x) { return 0; }

Private int blipvert(long x) { return 0; }

Protected long blipvert(int x, int y) { return 0; }

Private int blipvert(int x) { return 0; }

Protected long blipvert(long x) { return 0; }
```

Respuestas correctas:

1. Public int blipvert(int x) { return 0; }

2. Protected long blipvert(int x) { return 0; }
3. Protected int blipvert(long x) { return 0; }
4. Private int blipvert(long x) { return 0; }
5. Protected long blipvert(long x) { return 0; }

24.- 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)

Change line 2 to: public int a;

Change line 13 to: public Sub(){ super(5);}

Change line 2 to: protected int a;

Change line 13 to: public Sub(){ this(5);}

Change line 13 to: public Sub(){ super(a);

Respuestas correctas:

1. Cambiar la línea 13 a: public Sub() { super(5); }
2. Cambiar la línea 2 a: protected int a;

25.- Given

public static void main(String[] args){

int[][] array2D = {{0,1,2}, {3,4,5,6}}; System.out.print(array2D[0].length + "");
System.out.print(array2D [1].getClass().isArray() System.out.print(array2D[0][1]);

}

+"");

¿Cuál es el resultado?

3false3

3false1

2false1

3true1

2true3

Respuesta correcta: 3true1

26.- Which two statments are true?

An interface CANNOT be extended by another interface.

An abstract class can be extended by a concrete class.

An abstract class CANNOT be extended by an abstract class.

An interface can be extended by an abstract class.

An abstract class can implement an interface.

An abstract class can be extended by an interface.

Respuestas correctas: An abstract class can be extended by a concrete class. y An abstract class can implement an interface.

27.- Given:

class Alpha{ String getType(){ return "alpha";}}

class Beta extends Alpha{String getType(){ return "beta";}} public class Gamma extends Beta { String getType(){ return "gamma";} void main(String[] args) {

public static

Gamma g1 =

(Gamma) new Alpha();

Gamma g2 = (Gamma) new Beta();

System.out.print(g1.getType()+" "+g2.getType());

}

}

What is the result?

Gamma gamma

Beta beta

Alpha beta

Compilation fails

Respuesta correcta: An exception is thrown at runtime.

28.- Which three are bad practices?

Checking for an IOException and ensuring that the program can recover if one occurs.

Checking for ArrayIndexOutOfBoundsException and ensuring that the program can recover if one occurs.

Checking for FileNotFoundException to inform a user that a filename entered is not valid.

Checking for Error and, if necessary, restarting the program to ensure that users are unaware problems.

Checking for ArrayIndexOutOfBoundsException when iterating through an array to determine when all elements.-have been visited

Respuestas correctas:

Checking for ArrayIndexOutOfBoundsException and ensuring that the program can recover if one occurs.

Checking for Error and, if necessary, restarting the program to ensure that users are unaware problems.

Checking for ArrayIndexOutOfBoundsException when iterating through an array to determine when all elements have been visited.

29.- What is the result?

```
public class Test {

public static void main(String[] args) {

int b = 4;

b--;

System.out.print(--b); System.out.println(b);
```

}

}

22

12

32

33

Respuesta correcta: 22

30.- In Java the difference between throws and throw Is

Throws throws an exception and throw indicates the type of exception that the method.

Throws is used in methods and throw in constructors.

Throws indicates the type of exception that the method does not handle and throw an

exception.

Respuesta correcta: Throws indicates the type of exception that the method does not handle and throw an exception.

31.- What is the result?

class Feline {

public String type = "f"; public Feline() {

}

}

System.out.print("feline ");

public class Cougar extends Feline (public Cougar() {

System.out.print("cougar ");

}

void go(){

type="c":

}

System.out.print(this.type + super.type):

public static void main(String[] args) { new Cougar().go():

}

Cougar c f.

Feline cougar c c.

Feline cougar c f.

Compilation fails.

Respuesta correcta: feline cougar c f

32.- Which statement, when inserted into line '' // TODO code application logic here", is valid in compilation time change?

public class SampleClass{

public static void main(String[] args) {

Another SampleClass asc = new AnotherSampleClass(); SampleClass sc = new SampleClass();

// TODO code application logic here

}

}

class Another SampleClass extends SampleClass {}

asc = sc;

sc = asc;

asc = (Object) sc;

asc= sc.clone();

Respuestas correctas:

1. asc = sc;
2. sc = asc;
3. asc = (Object) sc;

33.- What is the result?

public class Test {

public static void main(String[] args) {

int array = {{0}, {0,1}, (0,2,4), (0,3,6,9), (0,4,8,12,16} };

System.out.println(array[4][1]); System.out.println(array[1][4]);

}

}

4 Null.

Null 4.

An IllegalArgumentException is thrown at run time.

4 An ArrayIndexOutOfBoundsException is thrown at run time.

Resultado: 4 seguido de ArrayIndexOutOfBoundsException.

Respuesta correcta: 4 An ArrayIndexOutOfBoundsException is thrown at run time.

34.- What is the result?

import java.util.*;

public class App {

public static void main(String[] args) {

List p = new ArrayList();

p.add(7);

p.add(1);

p.add(5);

p.add(1);

p.remove(1);

System.out.println(p);

}

}

[7, 1, 5, 1]

[7, 5, 1]

[7, 5]

[7, 1]

Respuesta correcta: [7, 5, 1]

35.- What is the result?

```
public class Bees {
public static void main(String[] args) {
try {
new Bees().go();
} catch (Exception e) {
}
}
System.out.println("thrown to main");
synchronized void go() throws InterruptedException { Thread t1 = new Thread();
t1.start();
System.out.print("1 "); t1.wait(5000);
System.out.print("2");
}
}
```

The program prints 1 then 2 after 5 seconds.

The program prints: 1 thrown to main.

The program prints: 1 2 thrown to main.

The program prints:1 then t1 waits for its notification.

Respuesta correcta: The program prints: 1 thrown to main.

36.- Which three are valid? (Choose three)

```
class ClassA {}
class ClassB extends ClassA {} class ClassC extends ClassA And:
ClassA p0= new ClassA();
ClassB p1 = new ClassB();
ClassC p2 = new ClassC();
ClassA p3= new ClassB();
```

ClassA p4 = new ClassC();

p0 = p1;

p1 = p2;

p2 = p4;

p2 = (ClassC)p1;

p1 = (ClassB)p3;

p2 = (ClassC)p4;

Respuestas correctas:

1. p0 = p1;
2. p1 = (ClassB)p3;
3. p2 = (ClassC)p4;

37.- Which three options correctly describe the relationship between the classes?

class Class1 { String v1; }

class Class2 {

Class1 c1;

String v2;

}

class Class3 {Class2 c1; String v3;}

Class2 has-a v3.

Class1 has-a v2.

Class2 has-a v2.

Class3 has-a v1.

Class2 has-a Class3.

Class2 has-a Class1.

Respuestas correctas:

1. Class2 has-a Class1.
2. Class3 has-a Class2.
3. Class2 has-a v2.

38.- Which three implementations are valid?

interface SampleCloseable {

public void close() throws java.io.IOException;

}

class Test implements SampleCloseable { public void close() throws java.io.IOException { // do

something } }

class Test implements SampleCloseable { public void close() throws Exception { // do

something } }

class Test implements SampleCloseable { public void close() throws FileNotFoundException {

// do something } }

class Test extends SampleCloseable { public void close() throws java.io.IOException { // do

something } }

class Test implements SampleCloseable { public void close() { // do something }}

Respuestas correctas:

1. class Test implements SampleCloseable { public void close() throws java.io.IOException { // do something } }
2. class Test implements SampleCloseable { public void close() { // do something } }

39.- What is the result?

class MySort implements Comparator<Integer> { public int compare(Integer x, Integer y) { return y.compareTo(x);

}

}

And the code fragment:

Integer[] primes = {2, 7, 5, 3}; MySort ms = new MySort(): Arrays.sort(primes, ms);

for (Integer p2: primes) { System.out.print(p2 + " "); }

2 3 5 7

2 7 5 3

7 5 3 2

Compilation fails.

Respuesta correcta: 7 5 3 2

40.- Which two possible outputs?

public class Main {

public static void main(String[] args) throws Exception { doSomething():

}

private static void doSomething() throws Exception {

}

System.out.println("Before if clause");

if (Math.random() > 0.5) { throw new Exception();} System.out.println("After if clause");

}}

Before if clause Exception in thread "main" java.lang.Exception at Main.doSomething

(Main.java:21) at Main.main (Main.java:15).

Before if clause Exception in thread "main" java.lang.Exception at Main.doSomething

(Main.java:21) at Main.main (Main.java:15) After if clause.

Exception in thread "main" java.lang.Exception at Main.doSomething (Main.java:21) at

Main.main (Main.java:15).

Before if clause After if clause.

Respuestas correctas:

1. Before if clause Exception in thread "main" java.lang.Exception at Main.doSomething (Main.java:21) at Main.main (Main.java:15).
2. Before if clause After if clause.