1. **¿Qué línea imprime FALSE?**

```
Integer eye = new Integer (42);

Double d = new Double (42.0);

int i= 42

double dd = 42.0

System.out.println(i == eye); //1

System.out.println(eye.equals(d)); //2

System.out.println(i == eye); //3

System.out.println(i == 42); //4

System.out.println(i == dd); //5
```

- 3
- 1
- 4
- 2
- Sin respuesta

2. **¿Qué código colocado después del comentario //For loop podrá llenarle elementos al arreglo al con los valores de la variable i?**

```
public class Lini{

  public static void main(String args[]){

   Lini Lin = new Lini();

   Lamethod();

  }

  public void amethod() {

          int []a = new int[4];

          int initial=0;

    //Start For loop
```

```
System.out.println(a[i]);
}
```

- for (int i=0; i <a.length(); i++)
- for (int i=0; i<a.length()-1; i++)
- for (int i=1; i<4; i++)
- for (int i=0; i<a.length(i++);
- Sin respuesta

3. ¿Qué línea es la salida?

```
class test {
  test() {
        try{
     throw new RuntimeException();
        } finally {
    System.out.println("Damn it");
        }
  }
public static void main(String args[]) {
        try {
     new test();
        } catch (Throwable t) {
    System.out.println("Caught");
        }
  }
}
```

- Compila con error. incorrect syntax
- Ninguno de los anteriores

- Error Runtime
- Compila con error. Incompatible types
- Sin respuesta

4. **Selecciona una respuesta:**

```
int[] arr = {1, 2, 3, 4};

int [] arr2 = new int[4];

arr2=arr;

System.out.println(arr2[4]);
```

- Imprime 4
- Runtime Exception. ArrayOutofBounds
- Compila con warnings
- Compila con error}
- Sin respuesta

5. "Instrucción: Relaciona las columnas entre sí en tu hoja de respuestas."

The pairs are:

- **00. (C) Pruebas Unitarias** --> A) Validar que los componentes desarrollados se ensamblen de forma adecuada con la aplicación.
- **00. (A) Pruebas de Ensamblaje** --> B) Este tipo de pruebas se llevan a cabo en un ambiente o entorno previo; incluye pruebas Funcionales, de Integración, de Regresión y de Excepción.
- **00. (B) Pruebas de Sistema** --> C) Validar que los componentes que forman parte del sistema funcionan correctamente y cumplen con los requisitos de manera independiente.
- **00. (D) Pruebas Funcionales** --> D) Validar los requerimientos de negocio (lo que se supone que el sistema debe hacer); pretenden validar que el sistema construido hace lo que razonablemente se espera de él.

6. "Instrucción: Relaciona las columnas entre sí en tu hoja de respuestas."

The pairs are:

- **00. (A) Pruebas Negativas o de Excepción** --> A) Destinadas a mostrar que un componente o sistema no funciona.

- **00. (C) Pruebas Técnicas (Estructurales)** --> B) Pruebas finales ejecutadas por Socio de Negocio y/o Usuario para asegurar que el sistema satisfaga las necesidades de la organización y usuario final, contando con la aceptación formal de que el sistema construido es el solicitado.

- **00. (B) Pruebas de Aceptación** --> C) Incluyen un conjunto de categoría de prueba como: stress, volumen, seguridad, estándares; a fin de verificar que todas las partes del sistema funcionan en sincronía y que la tecnología y arquitectura están siendo usadas adecuadamente.

- **00. (D) Pruebas Unitarias** --> D) Validar que los componentes que forman parte del sistema funcionan correctamente y cumplen con los requisitos de manera independiente.

7. ¿Cuál será el resultado cuando se intenta compilar y ejecutar el siguiente código?

```
public class Conv{
 Conv c=new Conv();
 String s=new String("ello");
 c.amethod(s);
 public void amethod(String s1){
 char c='H';
 s1=c+s1;
 System.out.println(s1);
 }
 }
```

- La compilación y generación de la cadena "hello"
- La compilación y generación de la cadena 'H ello'
- La compilación y generación de la cadena 'helloH'
- Compila y genera error en tiempo de ejecución

8. (Repetición):

Selecciona una respuesta

```
 int [] iarr= new int[3];
 String [] sarr={"a","b","c"};
 for(String s: sarr)
 System.out.println(s);
```

- Imprime 1 2 3 4
- Error Runtime
- Ninguno de los anteriores
- Compila con error

9. **(Repetición):**

¿Qué línea imprime FALSE?

```
Integer eye = new Integer(42);
Double d = new Double(42.0);
int i = 42;
double dd = 42.0;
System.out.println(eye==eye); //1
System.out.println(eye.equals(d)); //2
System.out.println(eye == 42); //3
System.out.println(eye.intValue() == dd); //4
System.out.println(i == dd); //5
```

- 3
- 1
- 4
- 2

10. **(Repetición):**

¿Qué código colocado después del comentario //For loop podrá llenarse elementos el arreglo a[] con los valores de la variable i ?

```
public class Linl{
public static void main(String arg[]){
int i[] =new int[4];
for(int i=0;i < 5; i++ )
{
//for loop podrá llenarse elementos el arreglo al recorrer la variable i

}
}
```

- i=0
- i[i] = 2
- i = i+1
- i[i] = i;

11. **(Repetición):**

¿Qué línea es la salida?

```
class test {
test() {
try {
throw new RuntimeException();
}
finally {
```

```
System.out.println("Damn it");
}
public static void main(String arg[]){
try {
new test();
}
catch(Throwable t) {
System.out.println("Caught");
}
}
```

- Compila con error: incorrect syntax
- Ninguno de los anteriores
- Error Runtime
- Compila con error: incompatible types

12. (Repetición):

Selecciona una respuesta

```
int [] iarr= new int[]{1,2,3,4};
String [] sarr=Arrays.toString(iarr);
for( String s: sarr){
System.out.println(s);
}
```

- Imprime 1 2 3 4
- Error Runtime Exception, ArrayOutOfBounds
- Compila con warnings
- Compila con error

13. (Repetición):

¿Cuál será el resultado cuando se intenta compilar y ejecutar el siguiente código?

```
public class Conv{
Conv c=new Conv();
String s=new String("ello");
c.amethod(s);
public void amethod(String s1){
char c='H';
s1=c+s1;
System.out.println(s1);
}
}
```

- La compilación y generación de la cadena "hello"

- La compilación y generación de la cadena 'H ello'
- La compilación y generación de la cadena 'helloH'
- Compila y genera error en tiempo de ejecución

14. (Repetición):

Selecciona una respuesta

```
int [] iarr= new int[3];
String [] sarr={"a","b","c"};
for(String s: sarr)
System.out.println(s);
```

- Imprime 1 2 3 4
- Error Runtime
- Ninguno de los anteriores
- Compila con error

15. (Repetición):

¿Qué línea imprime FALSE?

```
Integer eye = new Integer(42);
Double d = new Double(42.0);
int i = 42;
double dd = 42.0;
System.out.println(eye==eye); //1
System.out.println(eye.equals(d)); //2
System.out.println(eye == 42); //3
System.out.println(eye.intValue() == dd); //4
System.out.println(i == dd); //5
```

- 3
- 1
- 4
- 2

16.

¿Qué código colocado después del comentario //For loop podrá llenarse elementos el arreglo a[] con los valores de la variable i ?

```
public class Linl{
public static void main(String arg[]){
int i[] =new int[4];
for(int i=0;i < 5; i++ )
{
```

```
//for loop podrá llenarse elementos el arreglo al recorrer la variable i}
}
```

- i=0
- i[i] = 2
- i = i+1
- i[i] = i;

17.

¿Qué línea es la salida?

```
class test {
tes() {
try {
throw new RuntimeException();
}
finally {
System.out.println("Damn it");
}
public static void main(String arg[]){
try {
new test();
}
catch(Throwable t) {
System.out.println("Caught");
}
}
```

- Compila con error: incorrect syntax
- Ninguno de los anteriores
- Error Runtime
- Compila con error: incompatible types

18.

Selecciona una respuesta

```
int [] iarr= new int[]{1,2,3,4};
String [] sarr=Arrays.toString(iarr);
for( String s: sarr){
System.out.println(s);
}
```

- Imprime 1 2 3 4
- Error Runtime
- Ninguno de los anteriores

- Compila con error

19. (Repetición):

¿Cuál es el valor de funcionRetornoControlador?

```
var funcionRetornoControlador;
traerArchivo();
function traerArchivo(){
leerArchivoServidor('PruebaParamsAJAXUltraAvanzado.jsp', recibeArchivo());
}
function recibirArchivoTexto(texto){
document.getElementById('divContenido').innerHTML = texto;
}
function leerArchivoServidor(archivo, funcionRetorno){
funcionRetornoControlador = funcionRetorno;
funcionRetorno(leerArchivoServidor(archivo));
}
```

- funcionRetorno
- leerArchivoServidor
- traerArchivo
- recibirArchivo

20. (Repetición):

¿Qué va a ser impreso si se intenta compilar y ejecutar el siguiente código?

```
public class MyClass{
static int i;
public static void main(String arg[]){
System.out.println(i);
}
}
```

- Al null
- 1
- 0
- Error: Variable i may not have been initialized

21. (Repetición):

¿Cuál será el resultado cuando se intenta compilar y ejecutar el siguiente código?

```
public class Conv{
Conv c=new Conv();
String s=new String("ello");
c.amethod(s);
public void amethod(String s1){
char c='H';
s1=c+s1;
System.out.println(s1);
}
}
```

- La compilación y generación de la cadena "hello"
- La compilación y generación de la cadena 'H ello'
- La compilación y generación de la cadena 'helloH'
- Compila y genera error en tiempo de ejecución

22. (Repetición):

Selecciona una respuesta

```
int [] iarr= new int[3];
String [] sarr={"a","b","c"};
for(String s: sarr)
System.out.println(s);
```

- Imprime 1 2 3 4
- Error Runtime
- Ninguno de los anteriores
- Compila con error

23. :

¿Qué línea imprime FALSE?

```
Integer eye = new Integer(42);
Double d = new Double(42.0);
int i = 42;
double dd = 42.0;
System.out.println(eye==eye); //1
System.out.println(eye.equals(d)); //2
System.out.println(eye == 42); //3
System.out.println(eye.intValue() == dd); //4
System.out.println(i == dd); //5
```

- 3
- 1
- 4
- 2

24.

¿Qué código colocado después del comentario //For loop podrá llenarse elementos el arreglo al recorrer la variable i ?

```
 public class Linl{
 public static void main(String arg[]){
 int i[] =new int[4];
 for(int i=0;i < 5; i++ )
 {
 //for loop podrá llenarse elementos el arreglo al recorrer la variable i

}
 }
```

- i=0
- i[i] = 2
- i = i+1
- i[i] = i;

25. :

¿Qué línea imprime FALSE?

```
 Integer eye = new Integer(42);
 Double d = new Double(42.0);
 int i = 42;
 double dd = 42.0;
 System.out.println(eye==eye); //1
 System.out.println(eye.equals(d)); //2
 System.out.println(eye == 42); //3
 System.out.println(eye.intValue() == dd); //4
 System.out.println(i == dd); //5
```

- 3
- 1
- 4
- 2

26. (Repetición):

Es el nombre del hilo, en el que corre el método main, de un programa Java.

- main
- central
- head
- prime

27. (Repetición):

En la práctica, son los modificadores de acceso que se emplean en las variables de instancia de una clase, para controlar los datos asignables a las mismas.

- final, private
- static, transient
- public, private
- private, protected

28. (Repetición):

Una excepción en Java es un error:

- En tiempo de ejecución
- En tiempo de compilación
- En la sintaxis del código
- Ninguna de las anteriores

29. (Repetición):

Son clases en Java, que a diferencia de los arreglos, pueden expandirse o contraerse dinámicamente, conforme se les agregan o restan elementos.

- Expansores
- Reductores
- Cadenas
- Colecciones

30. (Repetición):

Un objeto de tipo String se caracteriza por ser:

- Inmutable
- Un tipo primitivo
- Mutable
- Polimórfico

31. (Repetición):

Se refiere al mecanismo que permite al programador Java, probar suposiciones durante la fase de desarrollo, sin tener que declarar el código a probar dentro de un bloque try.

- Bloque
- Comentario
- Contención (contention)
- Asención (assertion)

32. (Repetición):

Es el estado en el que se encuentra un hilo, cuando se encuentra “esperando” la disponibilidad de un recurso.

- Durmiendo (sleep)
- Bloqueado (blocked)
- Muerto (dead)
- Ninguno de los anteriores

33. (Repetición):

Los operadores lógicos de corto circuito (short-circuit) evalúan:

- Siempre ambos lados de la expresión lógica
- Condicionalmente el lado derecho de la expresión lógica
- Únicamente el lado izquierdo de la expresión lógica
- Únicamente el lado derecho de la expresión lógica

34. (Repetición):

Es el componente de la máquina virtual de Java, que coordina la ejecución de varios hilos.

- Controlador (controller)
- Pila (stack)
- Programador de hilos (thread scheduler)
- Memoria

35. (Repetición):

Se refiere a los diseños orientados a objetos, los cuales ya han sido probados y garantizan la reducción de errores potenciales en el código.

- Patrones de diseño
- Hojas de estilo
- Interfase
- Metadatos

36. (Repetición):

Al comparar caracteres, Java emplea el valor:

- Unicode de los caracteres
- ANSI de los caracteres
- UTF-8 de los caracteres
- Ninguno de los anteriores

37. (Repetición):

Al terminar de escribir datos a un flujo de salida (output stream), se emplea este método para garantizar que todos los datos en el flujo, sean escritos al archivo asociado.

- Unload
- Discharge
- Empty
- Flush

38. (Repetición):

Es el mecanismo nativo de Java a través del cual, el estado de un objeto puede ser guardado y posteriormente recuperado.

- Contención
- Serialización
- Almacenamiento
- Registro

39. (Repetición):

En el API I/O de Java, se encuentra definida como una clase, la cual es una representación abstracta de la ruta de un archivo o directorio.

- Directory
- Pathname
- File
- FilePath

40. (Repetición):

Es un tipo de colección, la cual, no acepta elementos duplicados, para lo cual, emplea el método equals

- Mapa (Map)
- Árbol (Tree)
- Lista (List)
- Conjunto (Set)

41. (Repetición):

Selecciona una respuesta

```
class test {
public static void main (String [] blah )
{
System.out.printf("%s", new test());
}
public String toString()
{
return "testing something";
}
}
```

- Da un runtime exception
- Imprime testing1234 o algo como eso
- Compila con error
- Imprime testing something

42. (Repetición):

¿Qué va a ser impreso si se intenta compilar y ejecutar el siguiente código?

```
int i=0;
switch (i) {
default:
System.out.println(“default”);
case 0:
System.out.println(“cero”);
break;
}
```

- default
- cero
- da error de compilación
- nada

43.

¿Cuál es el valor de funcionRetornoControlador?

```
var funcionRetornoControlador;
traerArchivo0;
function traerArchivo0{
leerArchivoServidor(‘PruebaParamsAJAXUltraAvanzado.jsp’, recibeArchiv0);
}
function recibirArchivoTexto(texto){
document.getElementById('divContenido').innerHTML = texto;
}
function leerArchivoServidor(archivo, funcionRetorno){
funcionRetornoControlador = funcionRetorno;
funcionRetorno(leerArchivoServidor(archivo));
}
```

- funcionRetorno
- leerArchivoServidor
- traerArchivo
- recibirArchivo

44.

¿Qué sucederá cuando compiles y ejecutes el siguiente código?

```
public class MyClass{
 static int i;
 public static void main(String arg[]){
 System.out.println(i);
 }
 }
```

- Al null
- 1
- 0
- Error: Variable i may not have been initialized

45.

¿Cuál será el resultado cuando se intenta compilar y ejecutar el siguiente código?

```
 public class Conv{
 Conv c=new Conv();
 String s=new String("ello");
 c.amethod(s);
 public void amethod(String s1){
 char c='H';
 s1=c+s1;
 System.out.println(s1);
 }
 }
```

- La compilación y generación de la cadena "hello"
- La compilación y generación de la cadena 'H ello'
- La compilación y generación de la cadena 'helloH'
- Compila y genera error en tiempo de ejecución

46.

Selecciona una respuesta

```
int [] iarr= new int[3];
String [] sarr={"a","b","c"};
for(String s: sarr)
System.out.println(s);
```

- Imprime las variables
- Genera un error de excepción de tipo NULL
- Genera un error de sintaxis.
- Genera una excepción

47. (Repetición):

Es el nombre del hilo, en el que corre el método main, de un programa Java

- main
- central
- head
- prime

48. (Repetición):

En la práctica, son los modificadores de acceso que se emplean en las variables de instancia de una clase, para controlar los datos asignables a las mismas.

- final, private
- static, transient
- public, private
- private, protected

49. (Repetición):

Una excepción en Java es un error:

- En tiempo de ejecución
- En tiempo de compilación
- En la sintaxis del código
- Ninguna de las anteriores

50. (Repetición):

Son clases en Java, que a diferencia de los arreglos, pueden expandirse o contraerse dinámicamente, conforme se les agregan o restan elementos.

- Expansores
- Reductores
- Cadenas
- Colecciones

51. (Repetición):

Un objeto de tipo String se caracteriza por ser:

- Inmutable
- Un tipo primitivo
- Mutable
- Polimórfico

52. (Repetición):

Se refiere al mecanismo que permite al programador Java, probar suposiciones durante la fase de desarrollo, sin tener que declarar el código a probar dentro de un bloque try.

- Bloque
- Comentario
- Contención (contention)
- Asención (assertion)

53. (Repetición):

Es el estado en el que se encuentra un hilo, cuando se encuentra “esperando” la disponibilidad de un recurso.

- Durmiendo (sleep)
- Bloqueado (blocked)
- Muerto (dead)
- Ninguno de los anteriores

54. (Repetición):

Los operadores lógicos de corto circuito (short-circuit) evalúan:

- Siempre ambos lados de la expresión lógica
- Condicionalmente el lado derecho de la expresión lógica
- Únicamente el lado izquierdo de la expresión lógica
- Únicamente el lado derecho de la expresión lógica

55. (Repetición):

Es el componente de la máquina virtual de Java, que coordina la ejecución de varios hilos.

- Controlador (controller)
- Pila (stack)
- Programador de hilos (thread scheduler)
- Memoria

56. (Repetición):

Se refiere a los diseños orientados a objetos, los cuales ya han sido probados y garantizan la reducción de errores potenciales en el código.

- Patrones de diseño
- Hojas de estilo
- Interfase
- Metadatos

57.

Al comparar caracteres, Java emplea el valor:

- Unicode de los caracteres
- ANSI de los caracteres
- UTF-8 de los caracteres
- Ninguno de los anteriores

58.

Al terminar de escribir datos a un flujo de salida (output stream), se emplea este método para garantizar que todos los datos en el flujo, sean escritos al archivo asociado

- Unload
- Discharge
- Empty
- Flush

59.

Es el mecanismo nativo de Java a través del cual, el estado de un objeto puede ser guardado y posteriormente recuperado

- Contención
- Serialización
- Almacenamiento
- Registro

60.

En el API I/O de Java, se encuentra definida como una clase, la cual es una representación abstracta de la ruta de un archivo o directorio

- Directory
- Pathname
- File
- FilePath

61.

Es un tipo de colección, la cual, no acepta elementos duplicados, para lo cual, emplea el método equals

- Mapa (Map)
- Árbol (Tree)
- Lista (List)
- Conjunto (Set)

62.

Selecciona una respuesta

```
class test {
public static void main (String [] blah )
{
System.out.printf("%s", new test());
}
public String toString()
{
return "testing something";
}
}
```

- Da un runtime exception
- Imprime testing1234 o algo como eso
- Compila con error
- Imprime testing something

63.

¿Qué va a ser impreso si se intenta compilar y ejecutar el siguiente código?

```
int i=0;
switch (i) {
default:
System.out.println("default");
case 0:
System.out.println("cero");
break;
}
```

- default
- cero
- da error de compilación
- nada

64.

Es el nombre del hilo, en el que corre el método main, de un programa Java

- main
- central
- head
- prime

65.

En la práctica, son los modificadores de acceso que se emplean en las variables de instancia de una clase, para controlar los datos asignables a las mismas.

- final, private
- static, transient
- public, private
- private, protected

66.

Una excepción en Java es un error:

- En tiempo de ejecución
- En tiempo de compilación
- En la sintaxis del código
- Ninguna de las anteriores

67. (Repetición):

Son clases en Java, que a diferencia de los arreglos, pueden expandirse o contraerse dinámicamente, conforme se les agregan o restan elementos.

- Expansores
- Reductores
- Cadenas
- Colecciones

68.

Un objeto de tipo String se caracteriza por ser:

- Inmutable
- Un tipo primitivo
- Mutable
- Polimórfico

69.

Se refiere al mecanismo que permite al programador Java, probar suposiciones durante la fase de desarrollo, sin tener que declarar el código a probar dentro de un bloque try.

- Bloque
- Comentario
- Contención (contention)
- Asención (assertion)

70.

Es el estado en el que se encuentra un hilo, cuando se encuentra “esperando” la disponibilidad de un recurso.

- Durmiendo (sleep)
- Bloqueado (blocked)
- Muerto (dead)
- Ninguno de los anteriores

71.

Los operadores lógicos de corto circuito (short-circuit) evalúan:

- Siempre ambos lados de la expresión lógica
- Condicionalmente el lado derecho de la expresión lógica
- Únicamente el lado izquierdo de la expresión lógica
- Únicamente el lado derecho de la expresión lógica

72.

Es el componente de la máquina virtual de Java, que coordina la ejecución de varios hilos.

- Controlador (controller)
- Pila (stack)
- Programador de hilos (thread scheduler)
- Memoria

73.

Se refiere a los diseños orientados a objetos, los cuales ya han sido probados y garantizan la reducción de potenciales fallas en el código.

- Patrones de diseño
- Patrones de arquitectura
- Patrones de comportamiento
- Ninguno de los anteriores

74.

Es la sintaxis empleada en Java, para declarar una colección que solo acepta objetos de un tipo en particular.

- Genérico
- Especificador
- Modificador
- Limitante

75.

El tipo parametrizado, en la sintaxis de un genérico se escribe:

- Entre paréntesis
- Entre paréntesis angulares < & >
- Entre corchetes
- Entre llaves

76.

Es el modificador del lenguaje Java, con el cual se marca una variable de instancia, la cual no se desea incluir en la serialización de una clase.

- static
- volatile
- remote
- transient

77. (Repetición):

Al emplear el operador de incremento (++) en una variable declarada con el modificador final.

- La variable conserva su valor original.
- La variable es preincrementada.
- La variable es postincrementada.
- Se produce un error en tiempo de compilación.

78.

Una clase Java soporta:

- Herencia simple
- Herencia multiple
- Herencia compuesta
- Ninguna de las anteriores

79.

Es el nombre del método que es invocado de manera implícita en un objeto, cuando se le pasa una referencia al mismo al método System.out.println:

- print
- toChar
- toString
- hashCode

80.

La invocación de métodos de manera polimórfica aplica solo para:

- Métodos estáticos
- Métodos de instancia
- Variables de instancia
- Métodos marcados con el modificador native

81.

Es el nombre del hilo, en el que corre el método main, de un programa Java.

- main
- central
- head
- prime

82.

En la práctica, son los modificadores de acceso que se emplean en las variables de instancia de una clase, para controlar los datos asignables a las mismas.

- final, private
- static, transient
- public, private
- private, protected

83.

Una excepción en Java es un error:

- En tiempo de ejecución
- En tiempo de compilación
- En la sintaxis del código
- Ninguna de las anteriores

84.

Son clases en Java, que a diferencia de los arreglos, pueden expandirse o contraerse dinámicamente, conforme se les agregan o restan elementos.

- Expansores
- Reductores
- Cadenas
- Colecciones

85.

Es el nombre del estado en el que se encuentra un hilo, antes de invocar el método Thread.start()

- Nuevo (new)
- Ejecutable (runnable)
- Muerto (dead)
- En ejecución (running)

86.

Son los tipos de clases internas en Java.

- Abstracta, estática, heredada, anónima
- Anónima, pública estándar estática
- Pública, nativa, heredada, anónima
- Regular, estática, local a un método, anónima

87.

Son consideradas clases internas, las cuales no tienen ninguna relación especial con su clase externa.

- Abstractas
- Anónimas
- Estáticas
- Regulares

88.

Se refiere a la declaración de clases dentro de otra.

- Clases abstractas
- Clases heredadas
- Clases Internas
- Ninguna de las anteriores

89.

Es la unidad de tiempo empleada en el argumento al método estático Thread.sleep().

- Hora
- Minuto
- Segundo
- Milisegundo

90.

Es un tipo de colección, en la cual, existe un mapeo entre una llave única (id) a un valor específico. Ambos, valor y llave son objetos.

- Mapa (Map)
- Árbol (Tree)
- Lista (List)
- Conjunto (Set)

91.

Una clase interna no-estática.

- Tiene acceso a todos los miembros de su clase externa.
- No tiene acceso a los miembros de su clase externa.
- Tiene acceso solo a los miembros públicos de su clase externa.
- Tiene acceso a todos los miembros de su clase externa, con excepción de los privados.

92.

El método String.split regresa.

- Un arreglo de cadenas
- Una lista de cadenas
- Un mapa de cadenas
- Ninguno de los anteriores

93.

Es la única manera de acceder a una clase interna regular.

- A través de la línea de comandos, empleando el comando java, seguido del nombre de la clase interna.
- Directamente en tiempo de ejecución, mediante una referencia a un objeto de su tipo.
- Indirectamente en tiempo de ejecución, mediante una referencia a un objeto del tipo de clase que la contiene.
- Ninguna de las anteriores.

94.

Es la sintaxis empleada en Java, para declarar una colección que solo acepta objetos de un tipo en particular.

- Genérico
- Especificador
- Modificador
- Limitante

95.

El tipo parametrizado, en la sintaxis de un genérico se escribe:

- Entre paréntesis
- Entre paréntesis angulares < & >
- Entre corchetes
- Entre llaves

96.

Es el modificador del lenguaje Java, con el cual se marca una variable de instancia, la cual no se desea incluir en la serialización de una clase.

- static
- volatile
- remote
- transient

97.

Al emplear el operador de incremento (++) en una variable declarada con el modificador final.

- La variable conserva su valor original.
- La variable es preincrementada.
- La variable es postincrementada.
- Se produce un error en tiempo de compilación.