1. What is the result?

public class Test {

public static void main(String[] args) {

int b = 4;

b--;

System.***out***.println(--b);

System.***out***.println(b);

}

}

22 12 32 33

el operador de predecremento actua antes de que imprima la variable b

2. 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.

3. What is the result?

class Feline {

public String type = "f ";

public Feline() {

System.***out***.println("feline ");

}

}

public class Cougar extends Feline {

public Cougar() {

System.***out***.println("cougar ");

}

void go() {

type = "c";

System.***out***.println(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.

Explicacion:

**Instanciación de Cougar**:

* Al crear una instancia de Cougar con new Cougar(), primero se ejecuta el constructor de la superclase Feline, que imprime "feline ".
* Luego se ejecuta el constructor de Cougar, que imprime "cougar "

**Método go**:

* Dentro del método go, type es asignado a "c". Esto afecta al atributo type de la instancia de Cougar porque type es una variable de instancia que se hereda de Feline.
* this.type ahora es "c", ya que type se ha modificado en la instancia de Cougar.
* super.type también es "c" porque super.type y this.type se refieren al mismo atributo en la instancia de Cougar. La asignación a this.type también modifica super.type ya que ambos apuntan al mismo atributo.

**Impresión en go**:

* El método go imprime this.type + super.type. Dado que ambos son "c", el resultado es "c c".

4. 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) {

AnotherSampleClass asc = new AnotherSampleClass();

SampleClass sc = new SampleClass();

//**TODO** code application logic here

}

}

class AnotherSampleClass extends SampleClass{}

asc = sc; sc = asc asc = (Object) sc; asc= sc.clone();Debido a que la variable de referencia sc es padre de la variable asc.

5. 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.

En la Segunda impression en consola, se sale del tamaño de ese array, solo tiene 2 elementos y lo busca en el index 4

6. 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]

Elimina el primer elemento que encuentra en la lista, de izquierda a derecha.

7. 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 tellIt = new Tell();

17. speakIT.tellItLikeItIs();

18. (Truth)speakIt.tellItLikeItIs();

19. ((Truth)speakIt).tellItLikeItIs();

20. tellIt.tellItLikeItIs();

21. (Truth)tellIt.tellItLikeItIs();

22. ((Truth)tellIt).tellItLikeItIs();

23. }

24. }

class Tell extends Speak implements Truth {

@Override

public void tellItLikeItIs(){

System.out.println(“Right on!”);

}

}

Interface Truth {

public void tellItLikeItIs();

Line 17 Line 18 Line 19 Line 20 Line 21 Line 22

* Línea 19 y 22: Las conversiones explícitas ((Truth)) se aplican a los objetos antes de llamar al método tellItLikeItIs().
* Línea 20: tellIt es un objeto de Tell, que tiene el método sobrescrito tellItLikeItIs().
* **Línea 18 y 21** no compilan porque la conversión (Truth) se aplica al resultado del método en lugar del objeto.
* **Línea 17** no compila porque speakIt es de tipo Speak, que no tiene el método tellItLikeItIs().

8. 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.

Esto se debe a que t1.wait(5000); lanza una excepción IllegalMonitorStateException al no estar en un bloque sincronizado, y dicha excepción es capturada e imprime el mensaje "thrown to main".

9. Which three are valid?

class ClassA{}

class ClassB extends ClassA{}

class ClassC extends ClassA{}

And:

ClassA p0 = new ClassA();

ClassA p1 = new ClassB();

ClassA p2 = new ClassC();

ClassA p3 = new ClassB();

ClassA p4 = new ClassC();

p0 = p1; la variable p0 es clase A, padre de la clase B por lo que puede contener sus objetos.

p1 = p2;

p2 = p4;

p2 = (ClassC)p1;

p1 = (ClassB)p3;

p2 = (ClassC)p4; estas dos respuestas estan casteadas correctamente

10. 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. --- Class2 tiene un atributo de tipo String llamado v2.

Class3 has-a v1. --- Class3 tiene un atributo de tipo Class2 llamado c1, y Class2 tiene un atributo de tipo Class1 llamado c1, que a su vez tiene un atributo String v1.

Class2 has-a Class3.

Class2 has-a Class1. --- Class2 tiene un atributo de tipo Class1 llamado c1.

11. Which three implementations are valid?