

Oracle_Actual_test_1Z0-851_v_2012_08_23

Number: 1Z0-851
Passing Score: 800
Time Limit: 120 min
File Version: 1.0

Oracle 1z0-851



Java Standard Edition 6 Programmer Certified

Professional Exam

Practice Test

Version: 4.2

Oracle 1z0-851: Practice Exam

Buena suerte Amigos desde Colombia!!!!

Andravel

Exam A

QUESTION 1

Given a pre-generics implementation of a method:

```
11. public static int sum(List list) {  
12.     int sum = 0;  
13.     for ( Iterator iter = list.iterator(); iter.hasNext(); ) {  
14.         int i = ((Integer)iter.next()).intValue();  
15.         sum += i;  
16.     }  
17. return sum;  
18. }
```

What three changes allow the class to be used with generics and avoid an unchecked warning? (Choose three.)

- A. Remove line 14.
- B. Replace line 14 with "int i = iter.next();".
- C. Replace line 13 with "for (int i : intList) {".
- D. Replace line 13 with "for (Iterator iter : intList) {".
- E. Replace the method declaration with "sum(List<int> intList)".
- F. Replace the method declaration with "sum(List<Integer> intList)".

Correct Answer: ACF

Section: (none)

Explanation

Explanation/Reference:

Explanation: En este caso se elimina la línea 14 porque al hacer generica la lista con :sum(List<Integer> intList) se puede recorrer de esta forma la lista: for (int i : intList) y nos evita dar casting en la línea 14 para asignar el valor a la variable "i".

QUESTION 2

A programmer has an algorithm that requires a java.util.List that provides an efficient implementation of add (0, object), but does NOT need to support quick random access. What supports these requirements?

- A. java.util.Queue
- B. java.util.ArrayList
- C. java.util.LinearList
- D. java.util.LinkedList

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Características de las listas:

ArrayList

Es una Lista volcada en un Array. Se debe utilizar en lugar de Vector como almacenamiento de objetos de propósito general. Permite un acceso aleatorio muy rápido a los elementos, pero realiza con bastante

lentitud las operaciones de insertado y borrado de elementos en medio de la Lista. Se puede utilizar un `ListIterator` para moverse hacia atrás y hacia delante en la Lista, pero no para insertar y eliminar elementos.
En resumen: Mas lento para agregar borrar elementos
Rápido para búsquedas y accesos aleatorios

LinkedList

Proporciona un óptimo acceso secuencial, permitiendo inserciones y borrado de elementos de en medio de la Lista muy rápidas. Sin embargo es bastante lento el acceso aleatorio, en comparación con la `ArrayList`. Dispone además de los métodos `addLast()`, `getFirst()`, `getLast()`, `removeFirst()` y `removeLast()`, que no están definidos en ningún interfaz o clase base y que permiten utilizar la Lista Enlazada como una Pila, una Cola o una Cola Doble
En resumen: Mas rápido para agregar borrar elementos
Lento para búsquedas y accesos aleatorios

QUESTION 3

Given:

```
11. // insert code here
12. private N min, max;
13. public N getMin() { return min; }
14. public N getMax() { return max; }
15. public void add(N added) {
16. if (min == null || added.doubleValue() < min.doubleValue())
17. min = added;
18. if (max == null || added.doubleValue() > max.doubleValue())
19. max = added;
20. }
21. }
```

Which two, inserted at line 11, will allow the code to compile? (Choose two.)

- A. `public class MinMax<?> {`
- B. `public class MinMax<? extends Number> {`
- C. `public class MinMax<N extends Object> {`
- D. `public class MinMax<N extends Number> {`
- E. `public class MinMax<? extends Object> {`
- F. `public class MinMax<N extends Integer> {`

Correct Answer: DF

Section: (none)

Explanation

Explanation/Reference:

Explanation: Para usar el método `doubleValue` es necesario que el tipo de dato "N" extienda la clase "Number" o la Clase "Integer" que poseen el método `doubleValue()`, el cual que devuelve un dato "int" como un Double".

`doubleValue()`
Returns the value of this Integer as a double.

QUESTION 4

Given:

```
12. import java.util.*;
13. public class Explorer2 {

14.     public static void main(String[] args) {

15.         TreeSet<Integer> s = new TreeSet<Integer>();

16.         TreeSet<Integer> subs = new TreeSet<Integer>();

17.         for(int i = 606; i < 613; i++)

18.             if(i%2 == 0) s.add(i);

19.         subs = (TreeSet)s.subSet(608, true, 611, true);

20.         s.add(629);

21.         System.out.println(s + " " + subs);

22.     }

23. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. [608, 610, 612, 629] [608, 610]
- D. [608, 610, 612, 629] [608, 610, 629]
- E. [606, 608, 610, 612, 629] [608, 610]
- F. [606, 608, 610, 612, 629] [608, 610, 629]

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

Explanation:

subSet(E fromElement, boolean fromInclusive, E toElement, boolean toInclusive)

Returns a view of the portion of this set whose elements range from fromElement to toElement.

El for de la linea 17 nos indica que se debe iterar desde 606 hasta 613, la siguiente linea nos dice que solo debemos asignar a la TreeSet los numeros pares osea [606,608,610,612]. luego se adiciona a s 629 quedando:

[606, 608, 610, 612, 629] .

En la linea 19 se hace uso del metodo subSet para extraer de la anterior TreeSet("s") el elemento 608 (inclusive) hasta el elemento 611(inclusive) osea extrae : [608,610], finalmente en la linea 23 nos une las dos TreeSet dandonos el resultado:

[606, 608, 610, 612, 629] [608, 610].

QUESTION 5

Given:

```
1. public class Score implements Comparable<Score> {

2.     private int wins, losses;

3.     public Score(int w, int l) { wins = w; losses = l; }

4.     public int getWins() { return wins; }
```

```

5. public int getLosses() { return losses; }
6. public String toString() {
7.     return "<" + wins + "," + losses + ">";
8. }
9. // insert code here
10. }

```

Which method will complete this class?

- A. public int compareTo(Object o){/*more code here*/}
- B. public int compareTo(Score other){/*more code here*/}
- C. public int compare(Score s1,Score s2){/*more code here*/}
- D. public int compare(Object o1,Object o2){/*more code here*/}

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Al implementar comparable se convierte en genérica por la sentencia <Score> la cual restringe la comparación a este tipo de dato, por lo tanto la opción B es válida pues declara correctamente el método compareTo (perteneciente a la interfaz Comparable)

A es incorrecto pues se necesita declarar Score no Object.

C y D son incorrectos pues el nombre del metodo es compareTo(Score other) con solo un parámetro "Score"

La declaración se debe hacer de esta manera:

```

int compareTo(T o)
    Compares this object with the specified object for order.

```

QUESTION 6

Given:

```

11. public class Person {
12.     private name;
13.     public Person(String name) {
14.         this.name = name;
15.     }
16.     public int hashCode() {
17.         return 420;
18.     }
19. }

```

Which statement is true?

- A. The time to find the value from HashMap with a Person key depends on the size of the map.
- B. Deleting a Person key from a HashMap will delete all map entries for all keys of type Person.
- C. Inserting a second Person object into a HashSet will cause the first Person object to be removed as a duplicate.

- D. The time to determine whether a Person object is contained in a HashSet is constant and does NOT depend on the size of the map.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La clase HashMap de Java es una implementación de la interface java.util.Map basada en una tabla hash. Los Maps (o mapas) pueden ser vistos como dos colecciones que contienen, en una colección, los valores y en la otra, las claves mediante las cuales podemos ubicar los objetos que queremos almacenar. Un HashMap es una estructura de datos que permite crear un mapa en memoria para la rápida identificación de elementos a partir de un dato usado como llave en una colección, los valores y en la otra, las claves mediante las cuales podemos ubicar los objetos que queremos almacenar.

```
public class HashMap<K,V>
extends AbstractMap<K,V>
implements Map<K,V>, Cloneable, Serializable
```

A es correcto. El acceso a los mapas depende siempre del tamaño del mismo.

B es incorrecto porque cada registro es único y contiene una única clave de acceso por lo tanto al borrar un elemento solo se borra el mismo.

C. Es incorrecto, solo se adiciona un nuevo valor al HashMap con la misma clave el valor en cuestión será reescrito no se removido.

D. Es incorrecto porque el tiempo requerido para almacenar un dato en el HashMap es proporcional a su tamaño.

QUESTION 7

Given:

```
5. import java.util.*;
6. public class SortOf {
7.     public static void main(String[] args) {
8.         ArrayList<Integer> a = new ArrayList<Integer>();
9.         a.add(1); a.add(5); a.add(3);
11.        Collections.sort(a);
12.        a.add(2);
13.        Collections.reverse(a);
14.        System.out.println(a);
15.    }
16. }
```

What is the result?

- A. [1, 2, 3, 5]
- B. [2, 1, 3, 5]
- C. [2, 5, 3, 1]
- D. [5, 3, 2, 1]
- E. [1, 3, 5, 2]
- F. Compilation fails.
- G. An exception is thrown at runtime.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

La línea 8 crea un ArrayList genérica de tipo Integer, la línea 9 adiciona elementos a esta lista quedando así:[1,5,3], la línea 11 ordena de forma ascendente y natural los elementos quedando[1,3,5], la línea 12 adiciona un elemento quedando la lista:[1,3,5,2] por ultimo se invierte el orden de la lista con la línea 13 quedando [2,5,3,1].

Metodo sort de la clase Collections:

```
static<T extends Comparable<? super T>> void  
    sort(List<T> list)  
    Sorts the specified list into ascending order, according to the natural ordering of its elements.
```

Metodo reverse de la clase Collections:

```
static void reverse(List<?> list)  
    Reverses the order of the elements in the specified list.
```

QUESTION 8

Given

11. public interface Status {

12. /* insert code here */ int MY_VALUE = 10;

13. } Which three are valid on line

12?

(Choose three.)

- A. final
- B. static
- C. native
- D. public
- E. private
- F. abstract
- G. protected

Correct Answer: ABD

Section: (none)

Explanation

Explanation/Reference:

A, B y D son verdaderos pues en la declaracion de una variable se pueden agregar los atributos final, static o public.

C es incorrecto pues native solo se aplica para metodos.

E y G. Private son incorrectos pues nunca un atributo de una interface puede ser privado; solo public o default.

F. Abstract es un modificador solo aplicable a Métodos, Interfaces y Clases.

QUESTION 9

Given:

5. class Atom {

```

6.  Atom() { System.out.print("atom "); }
7. }
8. class Rock extends Atom {
9.  Rock(String type) { System.out.print(type); }
10. }
11. public class Mountain extends Rock {
12.  Mountain() {
13.  super("granite ");
14.  new Rock("granite ");
15. }
16. public static void main(String[] a) { new Mountain(); }
17. }

```

What is the result?

- A. Compilation fails.
- B. atom granite
- C. granite granite
- D. atom granite granite
- E. An exception is thrown at runtime.
- F. atom granite atom granite

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

A. es incorrecta la sintaxis esta bien.

F. es correcta la clase mountain llama en la linea 13 a su super constructor Rock(String type) el cual lo que que hace es ocultamente llamar primero a su superconstructor en atom que es Atom() el cual genera la cadena "atom " y posteriormente ejecuta System.out.print(type) generando "granite " y este proceso se repite en la llamada de la clase de la linea 14;

Constructores y herencia en Java

Sean las clases "Base" y "Derivada".

TODAS las clases de Java tienen AL MENOS un constructor. Siempre.

Si no ponemos ninguno, Java les pone automáticamente uno de este estilo: Base(){}

Si ponemos aunque sólo sea uno (sea como sea), Java ya no añadirá el suyo.

La primera línea de un constructor es SIEMPRE una llamada al constructor de la clase base.

Si no ponemos ninguna llamada, Java pondrá automáticamente una de este estilo: super().

Si ponemos una nuestra, Java ya no añadirá la suya.

En una clase derivada, se ejecuta antes el constructor de la clase base que el suyo (recuérdese que la llamada a "super" es la primera instrucción del constructor).

QUESTION 10

Click the Exhibit button. Which three 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. }
```

Close File Comment Help

- 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.

Correct Answer: BEF

Section: (none)

Explanation

Explanation/Reference:

A es incorrecta pues la sintaxis esta bien.

B es correcta pues la linea 32 crea un objeto de tipo Beta y llama a su metodo testFoo() el cual declara una clase "A" y da a imprimir fubar(new A()) este submetodo crea un objeto tipo "A" y retorna un tipo int que en este caso es 2 el cual es el valor arrojado en la clase interna.

E es correcto pues no afecta en nada eliminar la primera declaracion de la clase "A".Se utilizará la segunda declaración de "A".

F es correcto pues si no se usa esta segunda declaración de "A" se utilizara la primera arrojando 1 como resultado.

QUESTION 11

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();

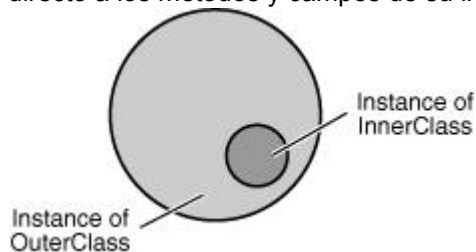
Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Una instancia de InnerClass puede existir solamente dentro de una instancia de OuterClass y tiene acceso directo a los métodos y campos de su instancia contenedora. La siguiente figura ilustra esta idea.



Una clase interna InnerClass existe dentro de una instancia de la clase externa OuterClass

Para llamar un metodo de una clase anidada es necesario referirse primero a la clase externa y luego a la interna, instanciar una clase es crear un objeto nuevo de ella mediante la palabra reservada "new" por ello la respuesta correcta es la D.

QUESTION 12

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.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

D es la respuesta correcta. en la linea 17 se crea una instancia de Alpha del tipo Beta. En la linea 18 se declara una variable de tipo Beta y se iguala a el objeto "a" osea que hace referencia en memoria a el mismo objeto "a" por lo tanto dara el mismo resultado a.foo() y b.foo(). Como se realiza un override sobre el método foo() el resultado sera "Bfoo Bfoo"

QUESTION 13

Click the Exhibit button. 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 Almpl will fail because of an error in line 2.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Asumiendo que en todos los sitios donde dice "//more code here" agregaremos un return con el tipo requerido, el error estaría en la línea 6 de la clase "C" pues al intentar redefinir el método execute() el tipo de dato java.lang.Object no es compatible con el tipo java.lang.String que desea redefinir de la clase "B".

QUESTION 14

Which two code fragments correctly create and initialize a static array of int elements? (Choose two.)

- A. `static final int[] a = { 100,200 };`
- B. `static final int[] a;`
`static { a=new int[2]; a[0]=100; a[1]=200; }`
- C. `static final int[] a = new int[2]{ 100,200 };`
- D. `static final int[] a;`
`static void init() { a = new int[3]; a[0]=100; a[1]=200; }`

Correct Answer: AB

Section: (none)

Explanation

Explanation/Reference:

La sintaxis de la opción A es la correcta para declarar instanciar e inicializar un arreglo.

B es correcto pues con el metodo static{} se pueden instanciar e inicializar las variables estaticas.

C es incorrecta pues la sintaxis no esta bien escrita en: `new int[2]{ 100,200 }` debería ser igual que la opcion A.

D es incorrecta pues la inicializacion de una variable static final solo es posible en la declaración, en el constructor o en el bloque de inicializacion static{}.

Bloques de Inicializacion {} static{}
Son bloques pero sin nombre. Si se dice que un constructor es como un método sin tipo, un inicializador es un bloque sin nombre

Los bloques de inicialización se ejecutan cuando la clase es cargada por primera vez (un bloque de inicialización estático) o cuando se crea una instancia (un bloque de inicialización estático)

Los bloques de inicialización se ejecutan cuando la clase es cargada por primera vez (un bloque de inicialización estático) o cuando se crea una instancia (un bloque de inicialización estático)

Primero se ejecutarán los inicializadores de clase , los de instancia y al final el constructor.

QUESTION 15

Given:

- 10. `interface Foo { int bar(); }`
- 11. `public class Sprite {`
- 12. `public int fubar(Foo foo) { return foo.bar(); }`
- 13. `public void testFoo() {`
- 14. `fubar(`
- 15. `// insert code here`
- 16. `);`
- 17. `}`

18. }

Which code, inserted at line 15, allows the class Sprite to compile?

- A. Foo { public int bar() { return 1; } }
- B. new Foo { public int bar() { return 1; } }
- C. new Foo() { public int bar() { return 1; } }
- D. new class Foo { public int bar() { return 1; } }

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

El metodo fubar() requiere que dentro de el sea incluido o creado un nuevo tipo de objeto de tipo "Foo" ya que la interface Foo no ha sido concretada se instancia una clase anonima dentro del método como argumento.

A es incorrecto pues le falta la palabra reservada "new" para crear la clase abstracta.

QUESTION 16

Given:

- 1. class Alligator {
- 2. public static void main(String[] args) {
- 3. int [][] x = {{1,2}, {3,4,5}, {6,7,8,9}};
- 4. int [][] y = x;
- 5. System.out.println(y[2][1]);
- 6. }
- 7. }

What is the result?

- A. 2
- B. 3
- C. 4
- D. 6
- E. 7
- F. Compilation fails.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

En la línea 3 se declara instancia e inicializa un arreglo con un tamaño definido por los valores que se le asignan: {{1,2}, {3,4,5}, {6,7,8,9}} de esta manera se crean 3 columnas con cantidad de filas variable. La línea 4 asigna la dirección de x al arreglo declarado "int y[][]", por último en la línea 5 se imprime el ítem de la columna número 3 y fila número 2 (ya que empiezan por cero) y ese valor es: 7.

QUESTION 17

Given:

- 22. StringBuilder sb1 = new StringBuilder("123");
- 23. String s1 = "123";

24. // insert code here
25. System.out.println(sb1 + " " + s1);

Which code fragment, inserted at line 24, outputs "123abc 123abc"?

- A. sb1.append("abc"); s1.append("abc");
- B. sb1.append("abc"); s1.concat("abc");
- C. sb1.concat("abc"); s1.append("abc");
- D. sb1.concat("abc"); s1.concat("abc");
- E. sb1.append("abc"); s1 = s1.concat("abc");
- F. sb1.concat("abc"); s1 = s1.concat("abc");
- G. sb1.append("abc"); s1 = s1 + s1.concat("abc");
- H. sb1.concat("abc"); s1 = s1 + s1.concat("abc");

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

La clase StringBuilder tiene varios metodos entre los cuales esta:

append(String str)

Appends the specified string to this character sequence.

la clase String tiene un metodo llamado :

concat(String str)

Concatenates the specified string to the end of this string.

entonces

sb1.append("abc") da como resultado : "123abc".

s1.concat("abc") es igual a "123abc".

La respuesta correcta es entonces la **E**.

A es incorrecto porque el metodo append en una clase String no existe.

B no es correcto porque nunca se le asigna el valor nuevo a s1.

C , D ,F y H son incorrectos porque el método concat() en una clase StringBuilder no existe.

G es incorrecto porque asi los métodos sean correctamente aplicados la respuesta sería:"123abc 123123abc"

QUESTION 18

Given that the current directory is empty, and that the user has read and write permissions, and the following:

```
11. import java.io.*;
12. public class DOS {
13.     public static void main(String[] args) {
14.         File dir = new File("dir");
15.         dir.mkdir();
16.         File f1 = new File(dir, "f1.txt");
17.         try {
18.             f1.createNewFile();
19.         } catch (IOException e) { ; }
```

```

20.     File newDir = new File("newDir");
21.     dir.renameTo(newDir);
22. }
23. }

```

Which statement is true?

- A. Compilation fails.
- B. The file system has a new empty directory named dir.
- C. The file system has a new empty directory named newDir.
- D. The file system has a directory named dir, containing a file f1.txt.
- E. The file system has a directory named newDir, containing a file f1.txt.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

Class **File**

extended by java.io.File

E. es correcto en la línea 14 se crea un objeto de tipo file con los datos del directorio que se va a crear, la línea 15 crea el directorio mediante llamada al método mkdir() ,la línea 18 crea el nuevo archivo, la línea 20 crea un nuevo objeto de tipo File con un directorio predeterminado, y finalmente en la línea 21 se renombra el viejo directorio por este ultimo.

QUESTION 19

Given:

```

11. class Converter {
12.     public static void main(String[] args) {
13.         Integer i = args[0];
14.         int j = 12;
15.         System.out.println("It is " + (j==i) + " that j==i.");
16.     }
17. }

```

What is the result when the programmer attempts to compile the code and run it with the command java Converter 12?

- A. It is true that j==i.
- B. It is false that j==i.
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error in line 13.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

D. es correcto ya que el dato args[] como esta declarado en el método main devuelve un String no un

Integer por ello la compilacion falla en la linea 13.

QUESTION 20

Given:

```
11. String test = "Test A. Test B. Test C.";
```

```
12. // insert code here
```

```
13. String[] result = test.split(regex);
```

Which regular expression, inserted at line 12, correctly splits test into "Test A", "Test B", and "Test C"?

- A. String regex = "";
- B. String regex = " ";
- C. String regex = ".*";
- D. String regex = "\\s";
- E. String regex = "\\s*";
- F. String regex = "\\w[\\.]+";

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

Una expresión regular es un patrón que describe a una cadena de caracteres. Todos hemos utilizado alguna vez la expresión *.doc. Las expresiones regulares se rigen por una serie de normas y hay una construcción para cualquier patrón de caracteres. Una expresión regular sólo puede contener (aparte de letras y números) los siguientes caracteres:

< \$, ^, ., *, +, ?, [,], . >

Una expresión regular, nos servirá para buscar patrones en una cadena de texto, por ejemplo encontrar cuantas veces se repite una palabra en un texto, para comprobar que una cadena tiene una determinada estructura.

Construct	Description
.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

Greedy	Reluctant	Possessive	Meaning
X?	X??	X?+	X, once or not at all
X*	X*?	X*+	X, zero or more times
X+	X+?	X++	X, one or more times
X{n}	X{n}?	X{n}+	X, exactly n times
X{n,}	X{n,}?	X{n,}+	X, at least n times
X{n,m}	X{n,m}?	X{n,m}+	X, at least n but not more than m times

E es correcta puesto que el metodo split requiere una "regular expression" o regex en este caso hace el corte de la cadena de caracteres donde se encuentre un "." \\. seguido de un espacio en blanco \s y

ademas se dice queeste espacio en blanco podria estar una o mas veces con el identificador ""

QUESTION 21

Given:

```
5. import java.util.Date;
6. import java.text.DateFormat;
21. DateFormat df;
22. Date date = new Date();
23. // insert code here
24. String s = df.format(date);
```

Which code fragment, inserted at line 23, allows the code to compile?

- A. df = new DateFormat();
- B. df = Date.getFormat();
- C. df = date.getFormat();
- D. df = DateFormat.getFormat();
- E. df = DateFormat.getInstance();

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

E es correcto. Pasar la fecha actual un cobjeto de tipo DateFormat es df = DateFormat.getInstance();

A.es incorrecta pues DateFormat es una clase abstracta y no se puede instanciar.

B y C.son incorrectas pues la clase Date no tiene el emtodo getFormat().

D. es incorrecto la clase abstracta DateFormat no contiene el emtodo getFormat.

Class DateFormat

```
public abstract class DateFormat
extends Format
```

DateFormat is an abstract class for date/time formatting subclasses which formats and parses dates or time in a language-independent manner. The date/time formatting subclass, such as SimpleDateFormat, allows for formatting (i.e., date -> text), parsing (text -> date), and normalization. The date is represented as a Date object or as the milliseconds since January 1, 1970, 00:00:00 GMT.

Class Date

```
public class Date
extends Object
implements Serializable, Cloneable, Comparable<Date>
```

The class Date represents a specific instant in time, with millisecond precision.

QUESTION 22

Given a class Repetition:

- 1. package utils;
- 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;

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

F es la respuesta correcta pues cumple con la verdadera sintaxis para importar metodos estaticos de otra clase.

Importacion Metodos Estaticos

Una innovación de java 5.0 son las llamadas importaciones estáticas que permiten llamar a un método o propiedad estática sin necesidad de hacer referencia al nombre de su clase.

La sintaxis general, es:

```
import static paquete.Clase.metodo_o_propiedad_static; //Para un sólo método o propiedad.
```

QUESTION 23

A UNIX user named Bob wants to replace his chess program with a new one, but he is not sure where the old one is installed. Bob is currently able to run a Java chess program starting from his home directory /home/bob using the command: `java -classpath /test:/home/bob/downloads/*.jar games.Chess` Bob's CLASSPATH is set (at login time) to: `/usr/lib:/home/bob/classes:/opt/java/lib:/opt/java/lib/*.jar` What is a possible location for the Chess.class file?

- A. /test/Chess.class
- B. /home/bob/Chess.class
- C. /test/games/Chess.class
- D. /usr/lib/games/Chess.class
- E. /home/bob/games/Chess.class
- F. inside jarfile /opt/java/lib/Games.jar (with a correct manifest)

G. inside jarfile /home/bob/downloads/Games.jar (with a correct manifest)

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

?

QUESTION 24

Given:

```
3. interface Animal { void makeNoise(); }

4. class Horse implements Animal {

5.     Long weight = 1200L;

6.     public void makeNoise() { System.out.println("whinny"); }

7. }

8. public class Icelandic extends Horse {

9.     public void makeNoise() { System.out.println("vinny"); }

10.    public static void main(String[] args) {

11.        Icelandic i1 = new Icelandic();

12.        Icelandic i2 = new Icelandic();

13.        Icelandic i3 = new Icelandic();

14.        i3 = i1; i1 = i2; i2 = null; i3 = i1;

15.    }

16. }
```

When line 15 is reached, how many objects are eligible for the garbage collector?

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4
- F. 6

Correct Answer: E

Section: (none)

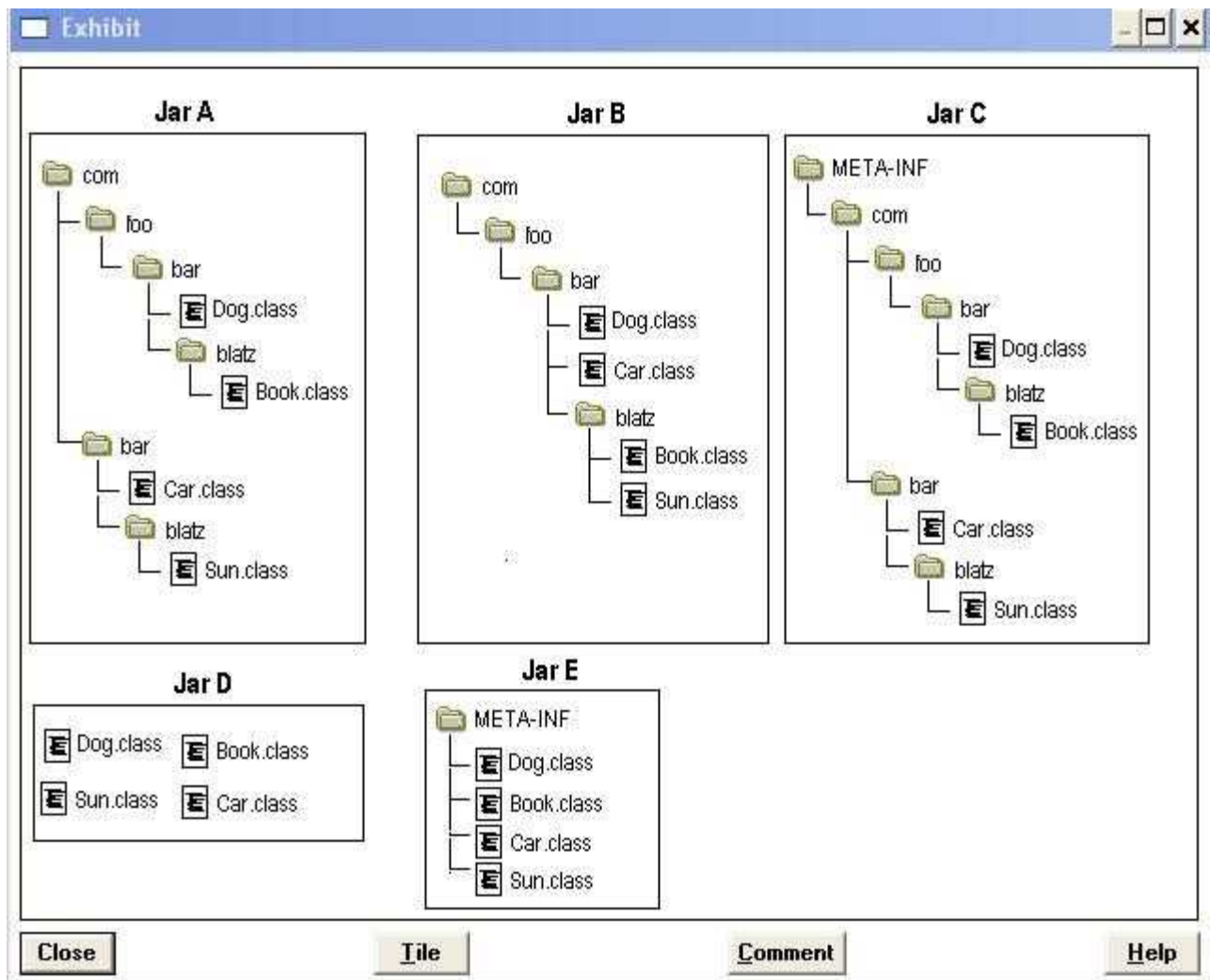
Explanation

Explanation/Reference:

E es correcto. Al final de la línea 14 todos los objetos "i1" e "i3" quedan apuntando solo un objeto y el objeto i2 es nulo. Siendo así dos objetos quedan listos para eliminar por el garbage collector. Además se crean dos objetos mas tipo String que son "vinny" y "whinny" , quedando 4 objetos para eliminar.

QUESTION 25

Click the Exhibit button. Given the fully-qualified class names: com.foo.bar.Dog com.foo.bar.blatz.Book com.bar.Car com.bar.blatz.Sun Which graph represents the correct directory structure for a JAR file from which those classes can be used by the compiler and JVM?



- A. Jar A
- B. Jar B
- C. Jar C
- D. Jar D
- E. Jar E

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Nos dan los nombres de clase completos para las clases de este modo:

`com.foo.bar.Dog`
`com.foo.bar.blatz.Book`
`com.bar.Car`
`com.bar.blatz.Sun`

Siendo de este modo observamos que todas las clases estan alojadas en el paquete "com", la clase "sun" esta dentro de `bar/blatz/` y la clase "book" esta alojada en el paquete `foo/bar/blatz` y dentro del paquete "bar" estan la clase "Dog", el unico diagrama que corresponde es el Jar A. Respuesta correcta "A".

QUESTION 26

Given classes defined in two different files:

1. package util;

```

2. public class BitUtils {
3.     private static void process(byte[] b) {}
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. }

```

What is required at line 5 in class SomeApp to use the process 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.

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

Para que se pueda acceder al método process de la clase BitUtils ya que es un método estático y esta en paquete diferente se necesita obligatoriamente importar este metodo estatico de esta manera: import static util.BitUtils.process; para poder acceder a el por lo tanto la respuesta correcta es la F nunca podra acceder a este método sin importarlo.

QUESTION 27

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.println(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.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La compilacion falla en la linea 14 pues despues que se asigna un valor a una variable Final en un bloque de inicializacion o en un constructor no puede ser asignado ningun valor. En esta linea se intenta asignar de nuevo un valor a la variable Final "id". La respuesta correcta es **A**.

QUESTION 28

Given:

```
13. public class Pass {  
14.     public static void main(String [] args) {  
15.         int x = 5;  
16.         Pass p = new Pass();  
17.         p.doStuff(x);  
18.         System.out.print(" main x = " + x);  
19.     }  
20.  
21.     void doStuff(int x) {  
22.         System.out.print(" doStuff x = " + x++);  
23.     }  
24. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. doStuff x = 6 main x = 6
- D. doStuff x = 5 main x = 5
- E. doStuff x = 5 main x = 6
- F. doStuff x = 6 main x = 5

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es la correcta y el resultado es: doStuff x = 5 main x = 5, ya que se esta trabajando con dos variables locales distintas llamadas en ambos casos "x" y se accede primero a la variable "x" del método doStuff y luego a la variable "x" del método "main" de la clase Pass.

QUESTION 29

Given:

```
1. public class GC {  
2.     private Object o;  
3.     private void doSomethingElse(Object obj) { o = obj; }  
4.     public void doSomething() {  
5.  
6.         doSomethingElse(o);  
7.         o = new Object();  
8.         doSomethingElse(null);  
9.         o = null;  
10.    }  
11. }
```

When the doSomething method is called, after which line does the Object created in line 5 become available for garbage collection?

- A. Line 5
- B. Line 6
- C. Line 7
- D. Line 8
- E. Line 9
- F. Line 10

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La llamada que se hace en la línea 8 llama al método doSomethingElse() el cual hace que el objeto llamado "o" se haga "null" y en este momento queda listo para eliminar, no es necesario pasar a la línea 9 para asignarle "null".

QUESTION 30

Given:

```
11. public static void test(String str) {  
12.     int check = 4;  
13.     if (check = str.length()) {  
14.         System.out.print(str.charAt(check -= 1) + ", ");  
15.     } else {  
16.         System.out.print(str.charAt(0) + ", ");  
17.     }  
18. } and the invocation:
```

21. test("four");
22. test("tee");
23. test("to");

What is the result?

- A. r, t, t,
- B. r, e, o,
- C. Compilation fails.
- D. An exception is thrown at runtime.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

La compilación falla en la línea 13, se debe utilizar el operador relacional == para que nos retorne un booleano válido para el "if". Por lo demás la sintaxis es correcta.

QUESTION 31

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(){}  
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.

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

Ya que el objeto "e" se crea como de clase "E" ó (new E) siempre que se invoque el método accederá al de

su clase, en este caso llama al método bMethod() de la línea 9 que es el de la clase "D".
A,B y C son incorrectas la sintaxis esta bien.

QUESTION 32

Given that: Gadget has-a Sprocket and Gadget 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{ }`

Correct Answer: AC

Section: (none)

Explanation

Explanation/Reference:

A y B son correctas pues en ambos casos las clases pueden acceder a los objetos que se requieren y las relaciones son las correctas.

B es incorrecto porque la clase Widget no tendría un Sprocket.

D,E y F son incorrectas porque no cumple con la premisa Gadget is a Widget, puesto que Widget esta extendiendo a Gadget.

QUESTION 33

A company that makes Computer Assisted Design (CAD) software has, within its application, some utility classes that are used to perform 3D rendering tasks. The company's chief scientist has just improved the performance of one of the utility classes' key rendering algorithms, and has assigned a programmer to replace the old algorithm with the new algorithm. When the programmer begins researching the utility classes, she is happy to discover that the algorithm to be replaced exists in only one class. The programmer reviews that class's API, and replaces the old algorithm with the new algorithm, being careful that her changes adhere strictly to the class's API. Once testing has begun, the programmer discovers that other classes that use the class she changed are no longer working properly. What design flaw is most likely the cause of these new bugs?

- A. Inheritance
- B. Tight coupling
- C. Low cohesion
- D. High cohesion
- E. Loose coupling
- F. Object immutability

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

El problema anterior se debe al estrecho acoplamiento (Tight coupling) que la clase que se va a reemplazar tenía con otros miembros del programa o sea no era del todo independiente como debiera.

coupling (acoplamiento)

El acoplamiento es en qué medida esta clase se integra con las implementaciones de otras clases, que un cambio en cualquier otra clase se traducirá en un cambio en esta clase. Por lo tanto siempre es mejor programar con interfaces.

cohesion (cohesion)

Cohesión significa que el conjunto de una clase está integrada. Una clase debe ser responsable de sí misma, debe hacer una cosa y en la medida de lo posible, hacer todo lo posible para hacer funcionar esa cosa. Un elemento es cohesivo a medida que cambia el elemento entero cuando el sistema necesita cambiar.

QUESTION 34

Which Man class properly represents the relationship "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>; }

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

En las relaciones "has a" hay que tener un objeto de un tipo dentro de la clase en el caso de la opción "D" la clase "Man" tiene un objeto perro dentro de ella "private Dog bestFriend"

A es incorrecto pues Man no es un Dog.

B es incorrecto porque la declaración dice un "Man" usa(implements) un "Perro"

C es incorrecto porque la clase "Man" tiene un objeto de tipo mejor amigo llamado "dog".

E y F usan genericos que no corresponden con lo que se pide.

QUESTION 35

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. f 13
- G. Compilation fails.
- H. An exception is thrown at runtime.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

En este tipo de creación de objetos del tipo ClasePadre `a = new Subclase();` al crearse los objetos y llamar los métodos se invocan los de la subclase y al invocar las variables se llaman las de la ClasePadre. En el ejercicio anterior el resultado de la llamada al método `f.addFive()` da como resultado "b " y la llamada a la variable `f.a` da como resultado la de el padre o sea 3.

QUESTION 36

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.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

La línea 31 genera el error en tiempo de ejecución pues el objeto "animal" es un "Dog" y de ninguna manera se puede hacer "cast"(transformación) por uno de tipo "Cat" como lo trata de hacer esta línea.

QUESTION 37

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); }

Correct Answer: CD

Section: (none)

Explanation

Explanation/Reference:

Dentro de cada constructor si no se encuentra una llamada al constructor de la clase padre siempre se crea uno por defecto sin parametros del tipo: `super()`, la linea 13 da error pues se hace llamada a este constructor `super()` de la clase padre y no se encuentra. Se debe por lo tanto llamar a algun constructor de la clase padre que si exista, por lo tanto se debe invocar al constructor `super(int a)` o redireccionar la llamada al constructor de la subclase `this(int a)` el cual a su vez llamara a el constructor de la clase padre.

E es incorrecto pues no se puede llamar dos veces el mismo constructor de la clase padre.

cannot reference a before supertype constructor has been called.

Constructores y herencia en Java

Sean las clases "Base" y "Derivada".

TODAS las clases de Java tienen AL MENOS un constructor. Siempre.

Si no ponemos ninguno, Java les pone automáticamente uno de este estilo: `Base(){}`

Si ponemos aunque sólo sea uno (sea como sea), Java ya no añadirá el suyo.

La primera línea de un constructor es SIEMPRE una llamada al constructor de la clase base.

Si no ponemos ninguna llamada, Java pondrá automáticamente una de este estilo: `super()`.

Si ponemos una nuestra, Java ya no añadirá la suya.

En una clase derivada, se ejecuta antes el constructor de la clase base que el suyo (recuérdese que la llamada a "super" es la primera instrucción del constructor).

QUESTION 38

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. fofoofoofoofoo
- B. foobarfoobarbar
- C. foobarfoofoofoo
- D. foobarfoobarfoo
- E. barbarbarbarbar
- F. fofoofooobarbar
- G. fofoofooobarfoo

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Despues de ser definidos dos objetos "b" y "s" la salida de las siguientes lineas es:

linea 6: "foo"

linea 7 "bar"

linea 8: "foo"

linea 9: "bar"

linea 10: "foo"

El total impreso es : "foobarfoobarfoo".Por lo tanto la opción **D** es correcta-

QUESTION 39

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.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

La variable "base" tiene acceso publico, pertenece a la clase anidada: InnerTriangle , pero dado que esta variable esta sujeta al modificador de acceso de su clase "InnerTriangle" toma el modificador que es default o de acceso solo para su paquete. Por ello la respuesta correcta es la C solo se accederá a la variable "base" desde su paquete "geometry".

QUESTION 40

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() { }

Correct Answer: ABEF

Section: (none)

Explanation

Explanation/Reference:

Para sobrescribir métodos de su clase padre, los métodos de las subclases deben tener un modificador de acceso menor(o mas debil) que el de su superclase. por ello las respuestas **A,B,E y F** son válidas.

C es incorrecto private es mayor que default.

D es incorrecto default es de mayor proteccion que protected

G es incorrecto private es de mayor nivel que protected.

QUESTION 41

Which two code fragments are most likely to cause a StackOverflowError? (Choose two.)

- A.

```
int []x = {1,2,3,4,5};  
for(int y = 0; y < 6; y++)  
    System.out.println(x[y]);
```

- B. `static int[] x = {7,6,5,4};
static { x[1] = 8;
x[4] = 3; }`
- C. `for(int y = 10; y < 10; y++)
doStuff(y);`
- D. `void doOne(int x) { doTwo(x); }
void doTwo(int y) { doThree(y); }
void doThree(int z) { doTwo(z); }`
- E. `for(int x = 0; x < 1000000000; x++)
doStuff(x);`
- F. `void counter(int i) { counter(++i); }`

Correct Answer: DF

Section: (none)

Explanation

Explanation/Reference:

Este error se produce cuando hay llamadas recursivas que nunca acabaran parecidas a un bucle infinito, son errores por funciones que se llaman a si mismas infinitamente y desbordan el stack, que es el la pila de almacenamiento de parametros y variables locales. Las opciones **D** y **F** crean este `StackOverflowError` por lo tanto son las respuestas correctas.

Parameters and local variables are allocated on the stack (with reference types the object lives on the heap and a variable references that object). The stack typically lives at the upper end of your address space and as it is used up it heads towards the bottom of the address space (ie towards zero).

Your process also has a heap, which lives at the bottom end of your process. As you allocate memory this heap can grow towards the upper end of your address space. As you can see, there is the potential for the heap to "collide" with the stack (a bit like techtonic plates!!!).

The common cause for a stack overflow is a bad recursive call. Typically this is caused when your recursive functions doesn't have the correct termination condition, so it ends up calling itself for ever. However, with gui programming it's possible to generate indirect recursion. For example, your app may be handling paint messages and whilst processing them it may call a function that causes the system to send another paint message. Here you've not explicitly called yourself, but the OS/VM has done it for you.

To deal with them you'll need to examine your code. If you've got functions that call themselves then check that you've got a terminating condition. If you have then check than when calling the function you have at least modified one of the arguments, otherwise there'll be no visible change for the recusivly called function and the terminating condition is useless.

If you've got no obvious recursive functions then check to see if you're calling any library functions that indirectly will cause your function to be called (like the implicit case above).

QUESTION 42

Given:

- 11. `class A {`
- 12. `public void process() { System.out.print("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.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Hay varios errores en el código uno de ellos es que dentro de una Inner class no se pueden tener métodos estáticos como es el caso del main de la línea 19. En segundo lugar el error dentro del método process() de la clase B es lanzado pero el método trabajador process() de la clase A no tiene throws IOException dentro de su declaración.

El primer error que se detecta en compilación está en la línea 14 pues IOException esta clase no es encontrada ya que pertenece a la Api Java.io y por ende hay que importarlo de esta forma :import java.io. IOException. La respuesta correcta es la **D**.

QUESTION 43

Given:

```

11. public void go(int x) {
12.     assert (x > 0);
13.     switch(x) {
14.         case 2: ;
15.         default: assert false;
16.     }
17. }
18. private void go2(int x) { assert (x < 0); }

```

Which statement is true?

- A. All of the assert statements are used appropriately.
- B. Only the assert statement on line 12 is used appropriately.
- C. Only the assert statement on line 15 is used appropriately.
- D. Only the assert statement on line 18 is used appropriately.
- E. Only the assert statements on lines 12 and 15 are used appropriately.
- F. Only the assert statements on lines 12 and 18 are used appropriately.
- G. Only the assert statements on lines 15 and 18 are used appropriately.

Correct Answer: G

Section: (none)

Explanation

Explanation/Reference:

?

QUESTION 44

Given:

```
1. public class Breaker2 {  
2.     static String o = "";  
3.     public static void main(String[] args) {  
4.         z:  
5.         for(int x = 2; x < 7; x++) {  
6.             if(x==3) continue;  
7.             if(x==5) break z;  
8.             o = o + x;  
9.         }  
10.    System.out.println(o);  
11. }  
12. }
```

What is the result?

- A. 2
- B. 24
- C. 234
- D. 246
- E. 2346
- F. Compilation fails.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

La respuesta correcta es la **B**. Dentro de la iteración planteada en la línea 5 las condiciones que se cumplen para que llegue a la línea 8 son $x == 2$ y $x == 4$, la línea o va adicionando estos valores dentro de la variable "o" dando como resultado "24".

Las palabras reservadas **break** y **continue**, se utilizan en Java para detener completamente un bucle (break) o detener únicamente la iteración actual y saltar a la siguiente (continue). Normalmente si usamos break o continue, lo haremos dentro de una sentencia if, que indicará cuándo debemos detener el bucle al cumplirse o no una determinada condición.

La gran diferencia entre ambos es que, break, detiene la ejecución del bucle y salta a la primera línea del programa tras el bucle y continue, detiene la iteración actual y pasa a la siguiente iteración del bucle sin salir de él (a menos, que el propio bucle haya llegado al límite de sus iteraciones).

QUESTION 45

Given:

```

11. public static void main(String[] args) {
12.     String str = "null";
13.     if (str == null) {
14.         System.out.println("null");
15.     } else (str.length() == 0) {
16.         System.out.println("zero");
17.     } else {
18.         System.out.println("some");
19.     }
20. }

```

What is the result?

- A. null
- B. zero
- C. some
- D. Compilation fails.
- E. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La línea 15 arroja un error pues la cláusula else no está bien declarada no puede seguir de parentesis y booleano solo debe seguir de corchetes o de una declaración.

QUESTION 46

Given:

```

11. public class Test {
12.     public static void main(String [] args) {
13.         int x = 5;
14.         boolean b1 = true;
15.         boolean b2 = false;
16.
17.         if ((x == 4) && !b2 )
18.             System.out.print("1 ");
19.         System.out.print("2 ");
20.         if ((b2 = true) && b1 )
21.             System.out.print("3 ");

```

22. }

23. }

What is the result?

- A. 2
- B. 3
- C. 1 2
- D. 2 3
- E. 1 2 3
- F. Compilation fails.
- G. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La sintaxis esta bien , en la linea 19 escribe la cadena "2" ya que no esta dentro de el "if" , la linea 21 imprime "3" ya que la condicion del if se cumple, dando como resultado "2 3 ".

QUESTION 47

Given:

```
11. static void test() throws Error {  
12.     if (true) throw new AssertionError();  
13.     System.out.print("test ");  
14. }  
15. public static void main(String[] args) {  
16.     try { test(); }  
17.     catch (Exception ex) { System.out.print("exception "); }  
18.     System.out.print("end ");  
19. }
```

What is the result?

- A. end
- B. Compilation fails.
- C. exception end
- D. exception test end
- E. A Throwable is thrown by main.
- F. An Exception is thrown by main.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es la correcta la llamada del método testr en la linea 16 es verdadero y lanza una AssertionError, no es necesario declarar "throws" en el main porque la clase Error pertenece a la Api "lang" y es cargada por defecto.

QUESTION 48

Given:

```
10. public class Foo {  
11.     static int[] a;  
12.     static { a[0]=2; }  
13.     public static void main( String[] args ) {}  
14. }
```

Which exception or error will be thrown when a programmer attempts to run this code?

- A. java.lang.StackOverflowError
- B. java.lang.IllegalStateException
- C. java.lang.ExceptionInInitializerError
- D. java.lang.ArrayIndexOutOfBoundsException

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

En el anterior código ocurre un error de inicialización de la variable estática "a" ; esto es porque un array siempre se debe instanciar dándole el tamaño que va a tener ya que los array son de tamaño fijo. El error se corrige cambiando la línea 12 por: static { a=new int[10];a[0]=2;}

ExceptionInInitializerError

```
public class ExceptionInInitializerError  
extends LinkageError
```

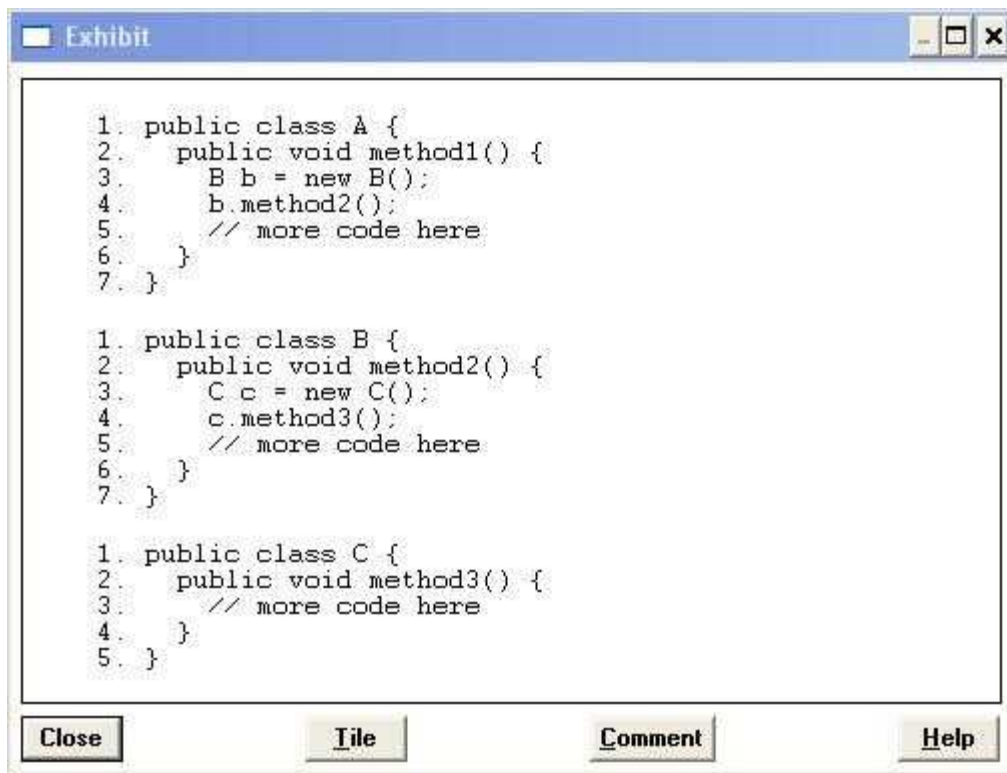
Signals that an unexpected exception has occurred in a static initializer. An ExceptionInInitializerError is thrown to indicate that an exception occurred during evaluation of a static initializer or the initializer for a static variable.

QUESTION 49

Click the Exhibit button. Given:

```
25. try {  
26.     A a = new A();  
27.     a.method1();  
28. } catch (Exception e) {  
29.     System.out.print("an error occurred");  
30. }
```

Which two statements are true if a NullPointerException is thrown on line 3 of class C? (Choose two.)



- A. The application will crash.
- B. The code on line 29 will be executed.
- C. The code on line 5 of class A will execute.
- D. The code on line 5 of class B will execute.
- E. The exception will be propagated back to line 27.

Correct Answer: BE

Section: (none)

Explanation

Explanation/Reference:

B y **E** son correctos. Cuando se llega a la línea 27 se invoca el método `method1()` el cual crea un objeto tipo "B" e invoca a `method2()` que a su vez crea un objeto tipo "C" donde según el encabezado se lanza una `NullPointerException` esta hace que se ejecute la línea 29 y se devuelva a la línea 27 terminando la ejecución.

QUESTION 50

Given:

```
11. public static void main(String[] args) {  
12.     for (int i = 0; i <= 10; i++) {  
13.         if (i > 6) break;  
14.     }  
15.     System.out.println(i);  
16. }
```

What is the result?

- A. 6
- B. 7

- C. 10
- D. 11
- E. Compilation fails.
- F. An exception is thrown at runtime.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

La compilación falla en la línea 15 ya que la variable `i` solo es conocida dentro de la declaración `"if"` y `System.out.println` no puede acceder a ella.

QUESTION 51

Given:

```
11. static class A {  
12.     void process() throws Exception { throw 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.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es la correcta y la línea 18 crea una clase anónima que ejecuta el método `process` de la clase `"B"` imprimiendo `"B"`.

QUESTION 52

Given:

```
1. public class Threads5 {  
2.     public static void main (String[] args) {  
3.         new Thread(new Runnable() {  
4.             public void run() {  
5.                 System.out.print("bar");
```

```
6.    }).start();  
7.    }  
8. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes normally and prints "bar".
- D. The code executes normally, but nothing prints.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

La sintaxis esta correcta en la linea 3 se crea una clase anónima tipo "Thread" y se le da como argumento otra clase anónima tipo "Runnable" definiendo su método run() dentro de ella para dar como resultado "bar" en la salida.

QUESTION 53

Given:

```
1. public class TestOne implements Runnable {  
2.     public static void main (String[] args) throws Exception {  
3.         Thread t = new Thread(new TestOne());  
4.         t.start();  
5.         System.out.print("Started");  
6.         t.join();  
7.         System.out.print("Complete");  
8.     }  
9.     public void run() {  
10.        for (int i = 0; i < 4; i++) {  
11.            System.out.print(i);  
12.        }  
13.    }  
14. }
```

What can be a result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes and prints "StartedComplete".
- D. The code executes and prints "StartedComplete0123".
- E. The code executes and prints "Started0123Complete".

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

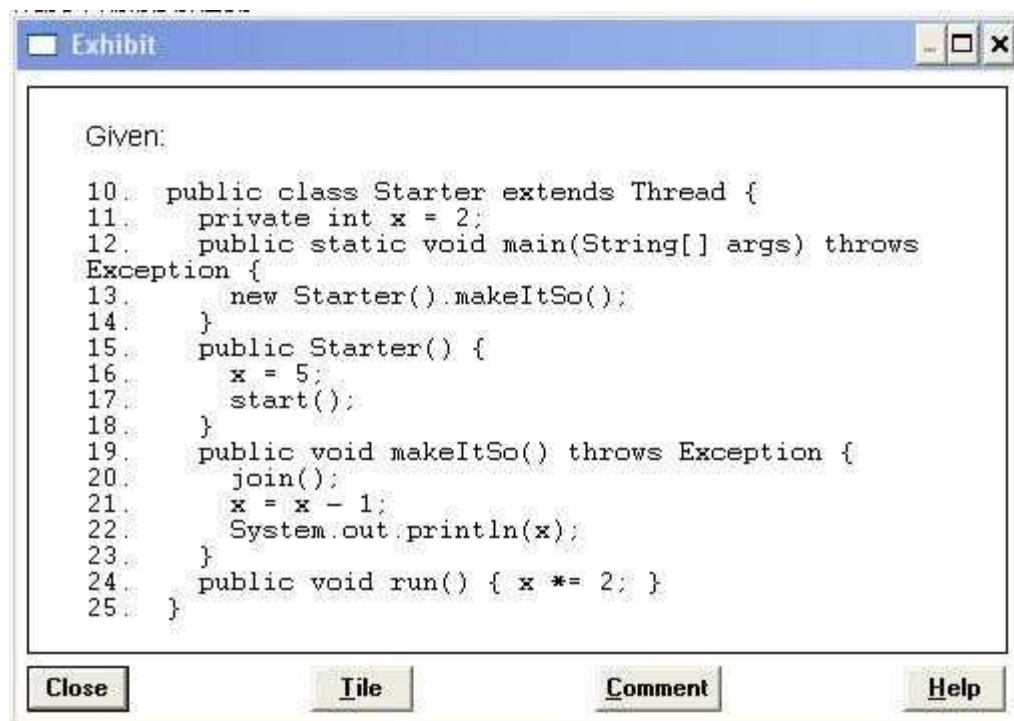
La respuesta correcta es la **E**. La línea 6 "t.join()" hace que el hilo "t" tenga que esperar hasta que el método main() que es el hilo principal complete los procesos hasta esta línea, en este caso se alcanza a imprimir "Started", posteriormente se imprime el proceso dentro del método run() del hilo en espera: "0123" y finalmente se imprime la línea 7 "Complete", dando como resultado "Started0123Complete".

El método join():

El método no estático join() permite al hilo "formarse en la cola de espera" de otro hilo. Si tienes un hilo B que no puede comenzar a ejecutarse hasta que se complete el proceso del hilo A, entonces querrás que B se forme en la cola de espera de A. Esto significa que B nunca podrá ejecutarse si A no completa su proceso.

QUESTION 54

Click the Exhibit button. What is the output if the main() method is run?



- A. 4
- B. 5
- C. 8
- D. 9
- E. Compilation fails.
- F. An exception is thrown at runtime.
- G. It is impossible to determine for certain.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es correcta, en este código se ejecuta primero el constructor Starter() asignando a x un valor de 5, posteriormente se ejecuta el método run() o la declaración de makeItSo, en este último método el hilo principal "main" es detenido en la línea 20 hasta tanto no termine el método "run" que pertenece a un segundo hilo "Thread-0" por lo tanto run hace que "x" sea igual a "10", por último se le resta 1 en la línea 21 y da como resultado 9.

QUESTION 55

Given:

```
1. public class TestFive {  
2.     private int x;  
3.     public void foo() {  
4.         int current = x;  
5.         x = current + 1;  
6.     }  
7.     public void go() {  
8.         for(int i = 0; i < 5; i++) {  
9.             new Thread() {  
10.                 public void run() {  
11.                     foo();  
12.                     System.out.print(x + ", ");  
13.                 } }.start();  
14. } }
```

Which two changes, taken together, would guarantee the output: 1, 2, 3, 4, 5, ? (Choose two.)

- A. move the line 12 print statement into the foo() method
- B. change line 7 to public synchronized void go() {
- C. change the variable declaration on line 2 to private volatile int x;
- D. wrap the code inside the foo() method with a synchronized(this) block
- E. wrap the for loop code inside the go() method with a synchronized block synchronized(this) { // for loop code here }

Correct Answer: AD

Section: (none)

Explanation

Explanation/Reference:

El "for" de la línea 8 crea 5 hilos los cuales estan compitiendo por acceder y ejecutar el método **foo()**; que esta dentro del método **run()**, para solucionar esto hay que meter linea 12 dentro de **foo()** y declararlo como synchronized para que ejecute uno a uno los hilos en el orden que fueron creados.

El lenguaje Java y el sistema de ejecución soportan la **sincronización** de threads mediante el uso de monitores. En general, un monitor está asociado con un objeto específico (una condición variable) y funciona como un bloqueo para ese dato. Cuando un thread mantiene el monitor para algún dato del objeto, los otros threads están bloqueados y no pueden ni inspeccionar ni modificar el dato.

Los segmentos de código dentro de programa que acceden al mismo dato dentro de threads concurrentes separados son conocidos como secciones críticas. En el lenguaje Java, se pueden marcar las secciones críticas del programa con la palabra clave synchronized.

QUESTION 56

Given:

```
1. public class Threads2 implements Runnable {
```

```

2.
3.  public void run() {
4.      System.out.println("run.");
5.      throw new RuntimeException("Problem");
6.  }
7.  public static void main(String[] args) {
8.      Thread t = new Thread(new Threads2());
9.      t.start();
10.     System.out.println("End of method.");
11. }
12. }

```

Which two can be results? (Choose two.)

- A. java.lang.RuntimeException: Problem
- B. run.
java.lang.RuntimeException: Problem
- C. End of method.
java.lang.RuntimeException: Problem
- D. End of method.
run.
java.lang.RuntimeException: Problem
- E. run.
java.lang.RuntimeException: Problem
End of method.

Correct Answer: DE

Section: (none)

Explanation

Explanation/Reference:

El código anterior contiene dos hilos uno es el "main" y otro es el creado en la línea 8, ya que no están sincronizados los dos compiten por ejecutar sus órdenes dando como resultado que podría ejecutarse primero "run. java.lang.RuntimeException: Problem" ó End of Method". Las respuestas **D** y **E** por lo tanto son correctas.

QUESTION 57

DRAG DROP

Click the Task button.

Drag and Drop

Given:

```
System.out.printf("Pi is approximately %f and E is approximately %b",
    Math.PI, Math.E);
```

Place the values where they would appear in the output.

Pi is approximately

and E is approximately

Values

3	3.141593	true	Math.PI
2	2.718282	false	Math.E

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Drag and Drop

Given:

```
System.out.printf("Pi is approximately %f and E is approximately %b",
    Math.PI, Math.E);
```

Place the values where they would appear in the output.

Pi is approximately

and E is approximately

Values

3	3.141593	true	Math.PI
2	2.718282	false	Math.E

QUESTION 58

DRAG DROP

Click the Task button.

Drag and Drop

Place code into the class so that it compiles and generates the output `answer=42`. Note: Code options may be used more than once.

Class

```
public class Placeholder {  
    private Placeholder object;  
    public Placeholder (Placeholder object) {  
        this.object = object;  
    }  
    public Placeholder getObject() {  
        return object;  
    }  
  
    public static void main(String[] args) {  
        Gen<String> str = new Gen<String>("answer");  
        Gen<Integer> intg = new Gen<Integer>(42);  
        System.out.println(str.getObject() + "=" +  
            intg.getObject());  
    }  
}
```

Code Options

- Gen<T>
- Gen<?>
- Gen
- ?
- T

Done

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Drag and Drop

Place code into the class so that it compiles and generates the output `answer=42`. Note: Code options may be used more than once.

Class

```
public class Gen<T> {
    private T object;
    public Gen (T object) {
        this.object = object;
    }
    public T getObject() {
        return object;
    }

    public static void main(String[] args) {
        Gen<String> str = new Gen<String>("answer");
        Gen<Integer> intg = new Gen<Integer>(42);
        System.out.println(str.getObject() + "=" +
            intg.getObject());
    }
}
```

Code Options

- Gen<T>
- Gen<?>
- Gen
- ?
- T

Done

QUESTION 59

DRAG DROP

Click the Task button.

Drag and Drop

Place the code fragments in position to complete the Displayable interface.

```
interface Reloadable {
    public void reload();
}

class Edit {
    public void edit() { /* Edit Here */ }
}

interface Displayable {
    Place here
    Place here
    Place here
}
```

Code Fragments

- extends
- implements
- public void display();
- public void display() { /* Display */ }
- Reloadable
- Edit

Done

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Drag and Drop

Place the code fragments in position to complete the Displayable interface.

```
interface Reloadable {  
    public void reload();  
}  
  
class Edit {  
    public void edit() { /* Edit Here */ }  
}  
  
interface Displayable  
    {  
        extends Reloadable  
        public void display();  
    }
```

Code Fragments

extends	public void display();	Reloadable
implements	public void display() { /* Display */ }	Edit

Done

QUESTION 60

DRAG DROP

Click the Task button.

The `doesFileExist` method takes an array of directory names representing a path from the root filesystem and a file name. The method returns true if the file exists, false if it does not.

Place the code fragments in position to complete this method.

```
public static boolean doesFileExist(String[] directories, String filename) {
```

Place here

```
for ( String dir : directories ) {
```

Place here

```
}
```

Place here

Place here

```
}
```

Code Fragments

path = path.getSubdirectory(dir);	return ! file.isNew();	return (file != null);
String path = "";	path = path.getFile(filename);	File path = new File("");
return file.exists();	return path.isFile();	File file = new File(path, filename);
path = new File(path, dir);	File path = new File(File.separator);	path = path + File.separator + dir;

- A.
- B.
- C.

D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

The `doesFileExist` method takes an array of directory names representing a path from the root filesystem and a file name. The method returns true if the file exists, false if it does not.

Place the code fragments in position to complete this method.

```
public static boolean doesFileExist(String[] directories, String filename) {  
    String path = "";  
    for ( String dir : directories ) {  
        path = path + File.separator + dir;  
    }  
    File file = new File(path, filename);  
    return file.exists();  
}
```

Code Fragments

<code>path = path.getSubdirectory(dir);</code>	<code>return ! file.isNew();</code>	<code>return (file != null);</code>
<code>String path = "";</code>	<code>path = path.getFile(filename);</code>	<code>File path = new File("");</code>
<code>return file.exists();</code>	<code>return path.isFile();</code>	<code>File file = new File(path, filename);</code>
<code>path = new File(path, dir);</code>	<code>File path = new File(File.separator);</code>	<code>path = path + File.separator + dir</code>

QUESTION 61

Given:

1. public class TestString1 {
2. public static void main(String[] args) {
3. String str = "420";
4. str += 42;
5. System.out.print(str);
6. }
7. }

What is the output?

- A. 42
- B. 420
- C. 462
- D. 42042
- E. Compilation fails.
- F. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La respuesta correcta es la **D**. El código anterior es una simple concatenación de el String `str = "420"` con el

número 42 en la línea 4 dando como resultado "42042".

QUESTION 62

Given:

```
12. Date date = new Date();  
13. df.setLocale(Locale.ITALY);  
14. String s = df.format(date);
```

The variable df is an object of type DateFormat that has been initialized in line 11. What is the result if this code is run on December 14, 2000?

- A. The value of s is 14-dic-2000.
- B. The value of s is Dec 14, 2000.
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error in line 13.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La línea 13 da error pues la clase abstracta DateFormat no posee el método citado: setlocale() , "Locale" se utiliza en su método llamado getDateInstance(int style, Locale aLocale) y en getTimeInstance(int style, Locale aLocale) .

QUESTION 63

Given:

```
1. public class KungFu {  
2.     public static void main(String[] args) {  
3.         Integer x = 400;  
4.         Integer y = x;  
5.         x++;  
6.         StringBuilder sb1 = new StringBuilder("123");  
7.         StringBuilder sb2 = sb1;  
8.         sb1.append("5");  
9.         System.out.println((x==y) + " " + (sb1==sb2));  
10.    }  
11. }
```

What is the result?

- A. true true
- B. false true
- C. true false
- D. false false
- E. Compilation fails.
- F. An exception is thrown at runtime.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Este código crea una variable `x` en la línea 3 inicializada en 400 y una variable `y` en la siguiente línea igualada al valor de la anterior solo pasa el valor no la referencia de la variable. por ello "`x`" queda = 401 y "`y`" =400, en cambio en la línea 6 y 7 se crean dos objetos `StringBuilder` pero apuntando a la misma dirección , quedando los dos en últimas con el mismo valor `sb1= sb2= "1235"`.

QUESTION 64

Given that the current directory is empty, and that the user has read and write privileges to the current directory, and the following:

```
1. import java.io.*;
2. public class Maker {
3.     public static void main(String[] args) {
4.         File dir = new File("dir");
5.         File f = new File(dir, "f");
6.     }
7. }
```

Which statement is true?

- A. Compilation fails.
- B. Nothing is added to the file system.
- C. Only a new file is created on the file system.
- D. Only a new directory is created on the file system.
- E. Both a new file and a new directory are created on the file system.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Para crear un directorio es necesario después de instanciar el objeto "`File`" , añadir la declaración `dir.mkdir()`; y para crear un archivo es necesario añadir la declaración `f.createNewFile()`. La respuesta correcta es la **B**.

QUESTION 65

Given:

```
12. String csv = "Sue,5,true,3";
13. Scanner scanner = new Scanner( csv );
14. scanner.useDelimiter(",");
15. int age = scanner.nextInt();
```

What is the result?

- A. Compilation fails.
- B. After line 15, the value of `age` is 5.
- C. After line 15, the value of `age` is 3.
- D. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La línea 15 arroja un error pues a pesar de ser valido el regex dela forma "," el resultado no arroja un entero que no se podria asignar a la variable age.

QUESTION 66

Given that t1 is a reference to a live thread, which is true?

- A. The Thread.sleep() method can take t1 as an argument.
- B. The Object.notify() method can take t1 as an argument.
- C. The Thread.yield() method can take t1 as an argument.
- D. The Thread.setPriority() method can take t1 as an argument.
- E. The Object.notify() method arbitrarily chooses which thread to notify.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

A es incorrecto pues el método sleep() acepta un dato numerico long : sleep(long millis)

B es incorrecto porque el método notify() no acepta argumentos.

C es incorrecto, el método yield() no acepta argumentos.

D es incorrecto, setPriority() acepta enteros como argumento no objetos Thread :setPriority(int newPriority)

La respuesta correcta es la **E**.

void notify()

Wakes up a single thread that is waiting on this object's monitor.

QUESTION 67

Given that Triangle implements Runnable, and:

- ```
31. void go() throws Exception {
32. Thread t = new Thread(new Triangle());
33. t.start();
34. for(int x = 1; x < 100000; x++) {
35. //insert code here
36. if(x%100 == 0) System.out.print("g");
37. }
38. public void run() {
39. try {
40. for(int x = 1; x < 100000; x++) {
41. // insert the same code here
42. if(x%100 == 0) System.out.print("t");
43. }
44. } catch (Exception e) { }
45. }
```

Which two statements, inserted independently at both lines 35 and 41, tend to allow both threads to

temporarily pause and allow the other thread to execute? (Choose two.)

- A. Thread.wait();
- B. Thread.join();
- C. Thread.yield();
- D. Thread.sleep(1);
- E. Thread.notify();

**Correct Answer:** CD

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Las respuestas correctas son al **C** y **D**, Los métodos yield() y sleep() son métodos que pausan los hilos que se están ejecutando.

El método join deja que se termine la ejecución de un hilo y luego el se ejecuta.

El método wait() y notify() trabajan en conjunto para pausar un hilo(wait()) mientras otro realiza una tarea y luego notificarlo (notify()) a continuar.

### QUESTION 68

Given:

1. public class Threads3 implements Runnable {
2.   public void run() {
3.     System.out.print("running");
4.   }
5. public static void main(String[] args) {
6.   Thread t = new Thread(new Threads3());
7.   t.run();
8.   t.run();
9.   t.start();
10. }
11. }

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes and prints "running".
- D. The code executes and prints "runningrunning".
- E. The code executes and prints "runningrunningrunning".

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

las líneas 7,8 y 9 ejecutan el mismo método "run()" el cual también puede ser llamado de la forma <miHilo>.run().

Al final la cadena resultante es "runningrunningrunning".

### QUESTION 69

Given:

```
1. public class Threads5 {
2. public static void main (String[] args) {
3. new Thread(new Runnable() {
4. public void run() {
5. System.out.print("bar");
6. }).start();
7. }
8. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes normally and prints "bar".
- D. The code executes normally, but nothing prints.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el código anterior se crea una clase anónima de tipo Thread que inicializa otra clase anónima dentro de tipo Threads5 y esta a su vez incluye el método run(). siendo válida la sintaxis, por lo tanto la respuesta correcta es la **C**

### QUESTION 70

Given:

```
11. public class PingPong implements Runnable {
12. synchronized void hit(long n) {
13. for(int i = 1; i < 3; i++)
14. System.out.print(n + "-" + i + " ");
15. }
16. public static void main(String[] args) {
17. new Thread(new PingPong()).start();
18. new Thread(new PingPong()).start();
19. }
20. public void run() {
21. hit(Thread.currentThread().getId());
22. }
23. }
```

Which two statements are true? (Choose two.)

- A. The output could be 8-1 7-2 8-2 7-1
- B. The output could be 7-1 7-2 8-1 6-1
- C. The output could be 8-1 7-1 7-2 8-2
- D. The output could be 8-1 8-2 7-1 7-2

**Correct Answer:** CD

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En realidad el método sincronized no funciona bien en el código pues en las líneas 17 y 18 se crean dos instancias diferentes de la clase PingPong independientes con su método run() tbn independiente entonces se protejera de acceder al mismo tiempo el citado run() pero dentro de cada instancia. Cada hilo correra en orden el método hit() pero se mezclara el resultado de los dos hilos.

**A** es incorrecto no puede arrojar el hilo 7-2 y luego 7-1 dentro de el esta siuncronizado run() deberia ser al reves 7-1 7-2

**B** es incorrecto solo hay dos hilos imprimiendo información y el hilo "main" pero este no imprime nada por ello 6-1 no se imprimira.

### QUESTION 71

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.

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

El código no compila pues en la línea 25 hay una llamada al método "y()" que no existe para la interfaz "A" .

QUESTION 72

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.

Correct Answer: ABCF

Section: (none)

Explanation

Explanation/Reference:

D es incorrecta se necesitaria que BabyRaccoon extendiera a Raccoon para ser verdad.

E es incorrecta dentro de la clase BabyRaccoon no existe ningun objeto de tipo Mammal.

QUESTION 73

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.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

C es correcto, La compilación falla en la línea 19, solo es autorizado llamar al un constructor super() de la clase padre, no se pueden anidar constructores de la misma clase.

QUESTION 74

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 ExtendedA(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 statement 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 modifier.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es la correcta:

La línea 12 crea un objeto de la clase ExtendedA llamado "ext", automáticamente se llama el constructor de esta clase

"ExtendedA(int numberOfInstances)"y se le da como argumento 420, el código pasa a la línea 9 donde una llamada al constructor super le pasa este argumento "420".

A continuación el constructor que se ejecuta es el de la clase "A" y con el argumento que le enviaron (420) lo asigna a la variable de clase numberOfInstances.

Por lo tanto la llamada a ext.numberOfInstances de la línea 13 nos imprime 420.

QUESTION 75

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;

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

A y B son incorrectos porque esto generaría un cambio en el valor de "i".

C es incorrecto porque al hacer el método addOne() privado, no se podrá acceder.

D es correcto porque en caso de utilizar el wrap: "Integer" funciona igual que el tipo int para la variable "i", este cambio crea un objeto tipo integer pero no afecta el resultado: i=1.

QUESTION 76

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

Correct Answer: ACEFG

Section: (none)

Explanation

Explanation/Reference:

B y D son incorrectos porque al sobrescribir un método se debe asignar un modificador de acceso mas débil que el del método a sobrescribir en este caso debería ser public.

QUESTION 77

Given:

Exhibit

Given:

```
10. public class Pizza {  
11.     ArrayList toppings;  
12.  
13.     public final void addTopping(String topping) {  
14.         toppings.add(topping);  
15.     }  
16.  
17.     public void removeTopping(String topping) {  
18.         toppings.remove(topping);  
19.     }  
20. }
```


And:

```
30. class PepperoniPizza extends Pizza {  
31.     public void addTopping(String topping) {  
32.         System.out.println("Cannot add Toppings");  
33.     }  
34.  
35.     public void removeTopping(String topping) {  
36.         System.out.println("Cannot remove Pepperoni");  
37.     }  
38. }
```


And:

```
50. Pizza pizza = new PepperoniPizza();  
51. pizza.addTopping("Mushrooms");  
52. pizza.removeTopping("Pepperoni");
```

CloseFileCommentHelp

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.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La respuesta correcta es la **A** La compilacion falla en la linea 31 pues un método final en este caso "addTopping(String topping)" no puede ser sobreescrito.

QUESTION 78

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;`

Correct Answer: AEF

Section: (none)

Explanation

Explanation/Reference:

B es incorrecto cada clase extiende por aparte a la clase padre "A" y por ello no son iguales.

C es incorrecto tienen tipos incompatibles uno es "ClassC" y el otro "ClassA"

D es incorrecto el casting no se puede realizar a la clase ClassA pues esta no conoce a la clase "ClassC".

QUESTION 79

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 GrizzlyBear.java

D. add `import animals.mammals.GrizzlyBear.*;` at line 2 in Salmon.java

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

B es correcto. La clase GrizzlyBear usa métodos de la clase Salmon que se encuentra dentro del paquete animal.fish por lo tanto para poder utilizar esta clase tenemos que realizar un import de la clase así: import animals.fish.*

QUESTION 80

Given:

1. package com.company.application;
- 2.
3. public class MainClass {
4. public static void main(String[] args) {}
5. }

And MainClass exists in the /apps/com/company/application directory. Assume the CLASSPATH environment variable is set to "." (current directory). Which two java commands entered at the command line will run MainClass? (Choose two.)

- A. java MainClass if run from the /apps directory
- B. java com.company.application.MainClass if run from the /apps directory
- C. java -classpath /apps com.company.application.MainClass if run from any directory
- D. java -classpath . MainClass if run from the /apps/com/company/application directory
- E. java -classpath /apps/com/company/application:. MainClass if run from the /apps directory
- F. java com.company.application.MainClass if run from the /apps/com/company/application directory

Correct Answer: BC

Section: (none)

Explanation

Explanation/Reference:

A es incorrecto. el compilador asume que la clase a ejecutar esta alojada en el directorio /apps y no es así.

D es incorrecto no se puede reconocer la ruta con el identificador "."

E es incorrecto el simbolo ":" no se reconoce en esta declaración.

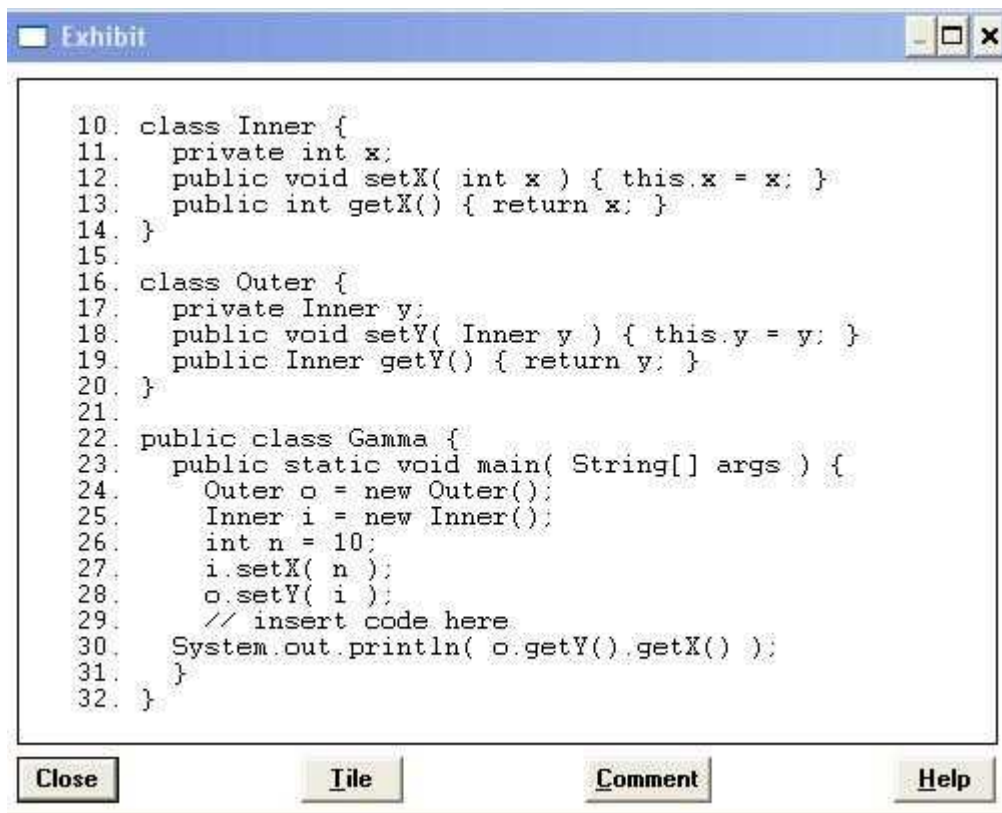
F es incorrecto el compilador intentara encontrar la ruta de la clase en: /apps/com/company/application/com/company/application/MainClass

Classpath (Java)

En el lenguaje de programación Java se entiende por Classpath una opción admitida en la línea de órdenes o mediante variable de entorno que indica a la Máquina Virtual de Java dónde buscar paquetes y clases definidas por el usuario a la hora de ejecutar programas.

QUESTION 81

Click the Exhibit button. 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 = new 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);

Correct Answer: BCF

Section: (none)

Explanation

Explanation/Reference:

QUESTION 82

A developer is creating a class Book, that needs to access class Paper. The Paper class is deployed in a JAR named myLib.jar. Which three, taken independently, will allow the developer to use the Paper class while compiling the Book class? (Choose three.)

- A. The JAR file is located at \$JAVA_HOME/jre/classes/myLib.jar.
- B. The JAR file is located at \$JAVA_HOME/jre/lib/ext/myLib.jar..
- C. The JAR file is located at /foo/myLib.jar and a classpath environment variable is set that includes /foo/myLib.jar/Paper.class.
- D. The JAR file is located at /foo/myLib.jar and a classpath environment variable is set that includes /foo/myLib.jar.
- E. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -cp /foo/myLib.jar/ Paper Book.java.
- F. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -d /foo/myLib.jar Book.java
- G. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -classpath /foo/myLib.jar Book.java

Correct Answer: BDG

Section: (none)

Explanation

Explanation/Reference:

B es correcto la carpeta `$JAVA_HOME/jre/lib/ext/myLib.jar`. es donde se alojan los archivos externos y donde java busca por defecto las librerías requeridas.

G es correcto el `-classpath` se declara bien incluyendo el `.jar` donde se aloja la clase `Paper`.

D es correcto si se incluye la línea `/foo/myLib.jar`. dentro de la variable `classpath` y se mete dentro de esta ruta el `.jar` se podrá compilar correctamente.

A es incorrecto la ruta de las librerías externas no es esa.

C es incorrecto se debe hacer la declaración sin el slash al final.

E es incorrecto se desea acceder a la clase `Paper` dentro de `myLib.jar` por ello debe aparecer de último dentro de la declaración del `classpath`.

F es incorrecto el parámetro `-d` solo se aplica para decir donde tienen que ir los archivos `.class` compilados.

QUESTION 83

Given:

```
11. interface DeclareStuff {  
12.     public static final int EASY = 3;  
13.     void doStuff(int t); }  
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 + ++s;  
21.         System.out.println("s " + s);  
22.     }  
23. }
```

What is the result?

- A. s 14
- B. s 16
- C. s 10
- D. Compilation fails.
- E. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La compilación falla en la línea 19 pues para implementar un método de una interfaz es necesario que el método que sobrescriba tenga un modificador de acceso `public` ya que los métodos de las interfaces son `public`. la interfaz debería quedar: `public void doStuff(int s)`.

QUESTION 84

Given:

```
11. public class Commander {  
12.     public static void main(String[] args) {  
13.         String myProp = /* insert code here */  
14.         System.out.println(myProp);  
15.     }  
16. }
```

and the command line: `java -Dprop.custom=gobstopper Commander` Which two, placed on line 13, will produce the output `gobstopper`? (Choose two.)

- A. `System.load("prop.custom");`
- B. `System.getenv("prop.custom");`
- C. `System.property("prop.custom");`
- D. `System.getProperty("prop.custom");`
- E. `System.getProperties().getProperty("prop.custom");`

Correct Answer: DE

Section: (none)

Explanation

Explanation/Reference:

De las dos formas **D** y **E** esta bien declarada la propiedad en **System.getProperty("prop.custom")** se llama el método `getProperty()` de la clase `System`, y en la otra primero se llama a **System.getProperties()** que devuelve un objeto tipo `Properties` que pertenece a `java.util` y almacena las propiedades completas, dentro de el tiene un metodo llamado `getProperty()` el cual funciona para devolver el valor de la propiedad nombrada.

Reading System Properties

The `System` class has two methods used to read system properties: `getProperty` and `getProperties`.

La clase `System` tiene dos versiones diferentes de `getProperty()`. Ambas versiones devuelven el valor de la propiedad nombrada en la lista de argumentos. La más simple de las dos `getProperty()` toma un sólo argumento: la clave de la propiedad que quiere buscar. Por ejemplo, para obtener el valor de `path.separator`, utilizamos la siguiente sentencia:

```
System.getProperty("path.separator");
```

Este método devuelve una cadena que contiene el valor de la propiedad. Si la propiedad no existe, esta versión de `getProperty()` devuelve `null`.

El último método proporcionado por la clase `System` para acceder a los valores de las propiedades es el método **getProperties()** que devuelve un objeto que contiene el conjunto completo de las propiedades del sistema. Se pueden utilizar varios métodos de la clase `Properties` para consultar valores específicos o para listar el conjunto completo de propiedades. Para más información sobre la clase `Properties`, puedes ver [Seleccionar y utilizar Propiedades](#).

QUESTION 85

Given:

```
3. public class Spock {  
4.     public static void main(String[] args) {
```

```

5.    Long tail = 2000L;
6.    Long distance = 1999L;

7.    Long story = 1000L;

8.    if((tail > distance) ^ ((story * 2) == tail))

9.    System.out.print("1");

10.   if((distance + 1 != tail) ^ ((story * 2) == distance))

11.   System.out.print("2");

12.   }

13. }

```

What is the result?

- A. 1
- B. 2
- C. 12
- D. Compilation fails.
- E. No output is produced.
- F. An exception is thrown at runtime.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

La respuesta correcta es la **E**. La primera declaración "if" de la línea 8 da como resultado `[true ^ true]` al ser xor evalúa verdadero solo si un argumento es verdadero y el otro falso, en este caso evalúa falso y no imprime nada.

El if de la línea 10 da como resultado `[false ^ false]` y al ser también conectado por xor(^) nos da como resultado falso, así que no se imprime nada para esta declaración tampoco.

QUESTION 86

Given:

```

1. public class GC {
2.     private Object o;
3.     private void doSomethingElse(Object obj) { o = obj; }
4.     public void doSomething() {
5.         Object o = new Object();
6.         doSomethingElse(o);
7.         o = new Object();
8.         doSomethingElse(null);
9.         o = null;
10.    }
11. }

```

When the doSomething method is called, after which line does the Object created in line 5 become

available for garbage collection?

- A. Line 5
- B. Line 6
- C. Line 7
- D. Line 8
- E. Line 9
- F. Line 10

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

El objeto creado en la línea 5 queda listo para la recolección de basura en la línea 8 veamos porque:

La línea 5 crea el objeto

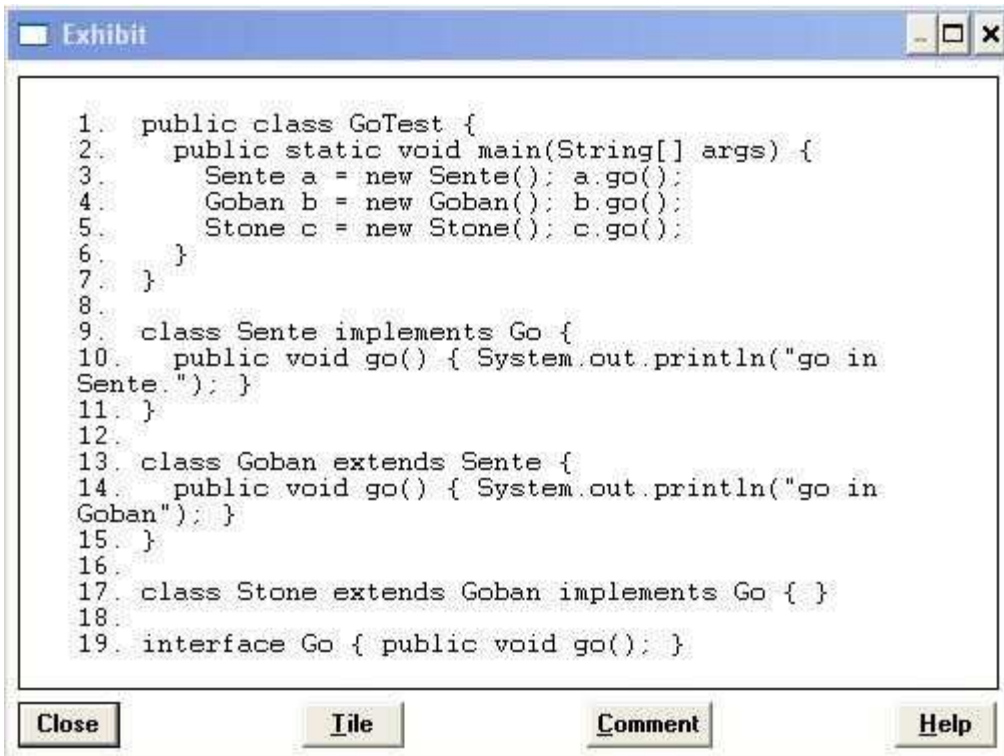
La línea 6 llama al método `doSomethingElse(Object obj)` el cual acepta el anterior objeto y lo iguala a la variable de instancia "o"

La línea 7 iguala la anterior variable "o" a un nuevo Objeto.

La línea 8 llama de nuevo al método `doSomethingElse(Object obj)` pero esta vez manda null como argumento lo cual hace que la variable de instancia "o" quede igualada a null y lista para el garbage collector.

QUESTION 87

Click the Exhibit button.



```
1. public class GoTest {
2.     public static void main(String[] args) {
3.         Sente a = new Sente(); a.go();
4.         Goban b = new Goban(); b.go();
5.         Stone c = new Stone(); c.go();
6.     }
7. }
8.
9. class Sente implements Go {
10.     public void go() { System.out.println("go in
Sente."); }
11. }
12.
13. class Goban extends Sente {
14.     public void go() { System.out.println("go in
Goban"); }
15. }
16.
17. class Stone extends Goban implements Go { }
18.
19. interface Go { public void go(); }
```

What is the result?

- A. go in Goban go in Sente go in Sente
- B. go in Sente go in Sente go in Goban
- C. go in Sente go in Goban go in Goban
- D. go in Goban go in Goban go in Sente
- E. Compilation fails because of an error in line 17.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

C es correcto. En el ejemplo anterior la línea 3 crea un objeto "Sence" y llama a su método go() imprimiendo "go in Sence".

La línea 4 crea un objeto "Goban" y llama a su método go() imprimiendo "go in Goban".

La línea 5 crea un objeto "Stone" y llama a su método go(), en este caso la clase Stone no necesita implementar el método "go()" de la interfaz "Go" puesto que extiende a otra clase que sí la implementa como es el caso de "Goban", por ello se ejecuta el primer método que implemente Go(); o sea imprime "go in Goban" de nuevo.

```
go in Goban go in Sente go in Sente
go in Sente go in Sente go in Goban
go in Sente go in Goban go in Goban
go in Goban go in Goban go in Sente
Compilation fails because of an error in line 17.
```

QUESTION 88

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("fern"); } is added to the Plant class.
- E. The code will compile if public Plant() { Plant("fern"); } is added to the Plant class.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

El anterior código no compila pues la clase "Tree" aunque no se declare un constructor java crea uno por defecto : "Tree()" el cual llama dentro de el a el constructor super() el que no existe en la clase "Plant", para corregir esto se debe crear un constructor de tipo Plant() y la solución la da la respuesta **D**.

E es incorrecto , para hacer referencia a un constructor dentro de la misma clase se utiliza la palabra reservada "this".

QUESTION 89

Click the Exhibit button.

Given:

25. A a = new A();

26. System.out.println(a.doit(4, 5));

What is the result?

```
1. public class A {
2.     public String doit(int x, int y) {
3.         return "a";
4.     }
5.
6.     public String doit(int... vals) {
7.         return "b";
8.     }
9. }
```

- A. Line 26 prints "a" to System.out.
- B. Line 26 prints "b" to System.out.
- C. An exception is thrown at line 26 at runtime.
- D. Compilation of class A will fail due to an error in line 6.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

En el anterior código se utiliza un parametro varargs, al llamar en la linea 26 al método sobrecargado "doit ()" tiene preferencia por el método sin varargs ya que el parámetro de longitud variable tiene que ser último y tener la mínima prioridad. Se imprime por lo tanto "a".

QUESTION 90

Given:

- ```
11. public enum Title {
12. MR("Mr."), MRS("Mrs."), MS("Ms.");
13. private final String title;
14. private Title(String t) { title = t; }
15. public String format(String last, String first) {
16. return title + " " + first + " " + last;
17. }
18. }
19. public static void main(String[] args) {
20. System.out.println(Title.MR.format("Doe", "John"));
21. }
```

What is the result?

- A. Mr. John Doe
- B. An exception is thrown at runtime.
- 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 20.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El enum anterior posee 3 valores constantes con un argumento cada uno los cuales se declaran en la linea 12.

la linea 13 declara la variable privada title a a cual se le pasaran los argumentos de las constantes anteriores, en la linea 14 se declara el constructor del enum que pasa los argumentos de las constantes a la variable privada "title".

la linea 15 declara un método llamado format el cual recibe dos argumentos de tipo String y retorna en pantalla el valor de la variable title mas el valor de los dos argumentos tomados.

Por ultimo un simple método main(String [] args) llama un objeto de tipo enum con su respectivo metodo format imprimiendo: Mr. John Doe

#### QUESTION 91

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 succeeds.
- 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.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

A es correcta, la sintaxis del código está bien escrita.

#### QUESTION 92

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.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el código existe una interfaz llamada TestA, también una Clase "Test" que en su línea 4 imprime el resultado de llamar al método toString() de la interfaz, para ello dentro de un System.out.println se crea una clase anónima tipo "TestA" y se implementa el método toString() el cual imprime "test".

### QUESTION 93

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.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En esta herencia de clases la línea 17 crea un nuevo objeto "Beta" de tipo "Alpha", la línea 18 le hace un casting a este objeto dejándolo de tipo "Beta". por ende las llamadas de los métodos foo de las líneas 19 y 20 ejecutan el método foo() de la clase Beta e imprimen ambos "Bfoo".

#### QUESTION 94

Given:

```
10. abstract public class Employee {
11. protected abstract double getSalesAmount();
12. public double getCommision() {
13. return getSalesAmount() * 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; }

**Correct Answer:** BD

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La clase "Sales" ya que es concreta debe implementar los métodos no implementados en la clase abstracta "Employee" con igual o menos proteccion de acceso por tanto el método getSalesAmount() debera ser protected o public.

#### QUESTION 95

Click the Exhibit button. 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 exception is thrown at runtime.
- D. Compilation fails because of an error in line 18.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La compilación falla pues internamente dentro de el constructor de la clase Employee en al linea 18 se esta llamando al super() constructor por defecto y este no existe en la clase person.

**QUESTION 96**

Given:

- 3. import java.util.\*;
- 4. public class Mapit {
- 5. public static void main(String[] args) {
- 6. Set<Integer> set = new HashSet<Integer>();
- 7. Integer i1 = 45;
- 8. Integer i2 = 46;
- 9. set.add(i1);
- 10. set.add(i1);
- 11. set.add(i2); System.out.print(set.size() + " ");
- 12. set.remove(i1); System.out.print(set.size() + " ");
- 13. i2 = 47;
- 14. set.remove(i2); System.out.print(set.size() + " ");

15. }

16. }

What is the result?

- A. 2 1 0
- B. 2 1 1
- C. 3 2 1
- D. 3 2 2
- E. Compilation fails.
- F. An exception is thrown at runtime.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El programa anterior en la línea 11 retorna "2" como tamaño pues el Hash set no permite duplicados y en la línea 10 no inserta doblemente "i1", posteriormente se remueve el objeto "i1" del arreglo y en la línea 12 devuelve "1" como tamaño.

Finalmente a la variable i2 se le asigna otro valor en la línea 13, dando como resultado que la referencia a este objeto dentro del hash set se pierda y en la línea 14 no se elimina "i2", por ultimo se imprime "1" de nuevo que es el tamaño final del arreglo.

#### QUESTION 97

Given:

1. public class Score implements Comparable<Score> {
2.   private int wins, losses;
3.   public Score(int w, int l) { wins = w; losses = l; }
4.   public int getWins() { return wins; }
5.   public int getLosses() { return losses; }
6.   public String toString() {
7.     return "<" + wins + "," + losses + ">";
8. }
9. // insert code here
10. }

Which method will complete this class?

- A. public int compareTo(Object o){/\*more code here\*/}
- B. public int compareTo(Score other){/\*more code here\*/}
- C. public int compare(Score s1,Score s2){/\*more code here\*/}
- D. public int compare(Object o1,Object o2){/\*more code here\*/}

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Cuando se implementa la interfaz Comparable el único método que existe en esta Interfaz y que hay que implementar se llama compareTo(T o) donde debe recibir un objeto de la misma clase Score para



compararlo con otro por lo tanto la respuesta correcta es la **B**.

#### QUESTION 98

A programmer has an algorithm that requires a java.util.List that provides an efficient implementation of add (0, object), but does NOT need to support quick random access. What supports these requirements?

- A. java.util.Queue
- B. java.util.ArrayList
- C. java.util.LinearList
- D. java.util.LinkedList

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Características de las listas:

#### ArrayList

Es una Lista volcada en un Array. Se debe utilizar en lugar de Vector como almacenamiento de objetos de propósito general. Permite un acceso aleatorio muy rápido a los elementos, pero realiza con bastante lentitud las operaciones de insertado y borrado de elementos en medio de la Lista. Se puede utilizar un ListIterator para moverse hacia atrás y hacia delante en la Lista, pero no para insertar y eliminar elementos.

**En resumen:** Mas lento para agregar borrar elementos  
Rápido para búsquedas y accesos aleatorios

#### LinkedList

Proporciona un óptimo acceso secuencial, permitiendo inserciones y borrado de elementos de en medio de la Lista muy rápidas. Sin embargo es bastante lento el acceso aleatorio, en comparación con la ArrayList. Dispone además de los métodos addLast(), getFirst(), getLast(), removeFirst() y removeLast(), que no están definidos en ningún interfaz o clase base y que permiten utilizar la Lista Enlazada como una Pila, una Cola o una Cola Doble

**En resumen:** Mas rápido para agregar borrar elementos  
Lento para búsquedas y accesos aleatorios

#### QUESTION 99

Given:

```
12. import java.util.*;
13. public class Explorer3 {
14. public static void main(String[] args) {
15. TreeSet<Integer> s = new TreeSet<Integer>();
16. TreeSet<Integer> subs = new TreeSet<Integer>();
17. for(int i = 606; i < 613; i++)
18. if(i%2 == 0) s.add(i);
19. subs = (TreeSet)s.subSet(608, true, 611, true);
20. subs.add(629);
21. System.out.println(s + " " + subs);
22. }
```

23. }

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. [608, 610, 612, 629] [608, 610]
- D. [608, 610, 612, 629] [608, 610, 629]
- E. [606, 608, 610, 612, 629] [608, 610]
- F. [606, 608, 610, 612, 629] [608, 610, 629]

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el código anterior el TreeSet "s" toma los valores pares del 606 al 613 o sea : [606,608,610,612], El TreeSet "subs" esta limitado a valores entre 608 y 611 <inclusive> extraido de "s" quedando [608,610]. La linea 20 adiciona el numero 629 a el TreeSet "s",quedando [606, 608, 610, 612, 629]. En la linea 21 se imprimen los dos TreeSets asi: [606, 608, 610, 612, 629] [608, 610]

## QUESTION 100

Given:

```
11. // insert code here
12. private N min, max;
13. public N getMin() { return min; }
14. public N getMax() { return max; }
15. public void add(N added) {
16. if (min == null || added.doubleValue() < min.doubleValue())
17. min = added;
18. if (max == null || added.doubleValue() > max.doubleValue())
19. max = added;
20. }
21. }
```

Which two, inserted at line 11, will allow the code to compile? (Choose two.)

- A. public class MinMax<?> {
- B. public class MinMax<? extends Number> {
- C. public class MinMax<N extends Object> {
- D. public class MinMax<N extends Number> {
- E. public class MinMax<? extends Object> {
- F. public class MinMax<N extends Integer> {

**Correct Answer:** DF

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En la línea 12 se declaran dos variables de tipo "N" las cuales como se ve en la línea 16 tienen que soportar métodos `doubleValue()` o sea tienen que ser numéricas. Por lo anterior hay que crear una clase genérica llamada "N" que extienda solo valores numéricos, `double` o `Integer`. Por lo tanto dos de las respuestas que cumplen este propósito son la **D** y la **F**.

#### QUESTION 101

Given:

```
12. import java.util.*;

13. public class Explorer1 {

14. public static void main(String[] args) {

15. TreeSet<Integer> s = new TreeSet<Integer>();

16. TreeSet<Integer> subs = new TreeSet<Integer>();

17. for(int i = 606; i < 613; i++)

18. if(i%2 == 0) s.add(i);

19. subs = (TreeSet)s.subSet(608, true, 611, true);

20. s.add(609);

21. System.out.println(s + " " + subs);

22. }

23. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. [608, 609, 610, 612] [608, 610]
- D. [608, 609, 610, 612] [608, 609, 610]
- E. [606, 608, 609, 610, 612] [608, 610]
- F. [606, 608, 609, 610, 612] [608, 609, 610]

**Correct Answer:** F

**Section:** (none)

**Explanation**

#### **Explanation/Reference:**

En el código anterior el `TreeSet "s"` toma los valores pares del 606 al 613 o sea : [606,608,610,612], El `TreeSet "subs"` está limitado a valores entre 608 y 611 <inclusive> extraído de "s" quedando [608,610]. La línea 20 adiciona el número 609 a el `TreeSet "s"`, quedando [606, 608,609, 610, 612] y ya que se puede incluir dentro de `subs` el valor 609, `subs` queda: [608, 609, 610]

En la línea 21 se imprimen los dos `TreeSets` así: [606, 608,609, 610, 612] [608,609, 610]

#### QUESTION 102

Given:

```
23. Object [] myObjects = {

24. new Integer(12),

25. new String("foo"),

26. new Integer(5),
```

```

27. new Boolean(true)

28. };

29. Arrays.sort(myObjects);

30. for(int i=0; i<myObjects.length; i++) {

31. System.out.print(myObjects[i].toString());
32. System.out.print(" ");

33. }

```

What is the result?

- A. Compilation fails due to an error in line 23.
- B. Compilation fails due to an error in line 29.
- C. A ClassCastException occurs in line 29.
- D. A ClassCastException occurs in line 31.
- E. The value of all four objects prints in natural order.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La Clase Array incluye el método sort(Objeto o) que ordena naturalmente el arreglo que se le da como argumento, pero siempre y cuando sean del mismo tipo, en el código anterior al intentar ordenar se produce una Exception en la línea 29 que es donde se da esta orden para el arreglo "myObjects" que contiene valores no compatibles.

### QUESTION 103

Given:

```

1. public class Donkey {

2. public static void main(String[] args) {

3. boolean assertsOn = false;

4. assert (assertsOn) : assertsOn = true;

5. if(assertsOn) {

6. System.out.println("assert is on");

7. }

8. }

9. }

```

If class Donkey is invoked twice, the first time without assertions enabled, and the second time with assertions enabled, what are the results?

- A. no output
- B. no output  
assert is on
- C. assert is on
- D. no output  
An AssertionError is thrown.
- E. assert is on

An AssertionError is thrown.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Cuando assert esta activo "-ea" se lanza una assercion en la linea 4 llamada : "true" ya que en la descripción de esta linea aparece :assertOn = true;

Cuando assert esta inactivo "-da" se compila pero no ejecuta nada porque nunca se entra a la linea 4 y pasa a la siguiente linea que contiene una declaracion "if" en la cual no entra puesto que la variable "assertsOn" permanece en el valor "false".

#### QUESTION 104

Given:

```
11. Float pi = new Float(3.14f);
12. if (pi > 3) {
13. System.out.print("pi is bigger than 3. ");
14. }
15. else {
16. System.out.print("pi is not bigger than 3. ");
17. }
18. finally {
19. System.out.println("Have a nice day.");
20. }
```

What is the result?

- A. Compilation fails.
- B. pi is bigger than 3.
- C. An exception occurs at runtime.
- D. pi is bigger than 3. Have a nice day.
- E. pi is not bigger than 3. Have a nice day.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La compilación falla en la linea 18 dado que la declaración finally no se puede ejecutar si no hay un bloque try y catch antes.

#### QUESTION 105

Given:

```
11. public static void main(String[] args) {
12. try {
13. args = null;
14. args[0] = "test";
```

```

15. System.out.println(args[0]);
16. } catch (Exception ex) {
17. System.out.println("Exception");
18. } catch (NullPointerException npe) {
19. System.out.println("NullPointerException");
20. }
21. }

```

What is the result?

- A. test
- B. Exception
- C. Compilation fails.
- D. NullPointerException

**Correct Answer: C**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

Este código produce un error en compilación en la línea 16, se da porque la excepción en la línea 16 es capturada como "Exception" ya que es más global o general que el catch de la línea 18 : NullPointerException.

#### QUESTION 106

Given:

```

22. public void go() {
23. String o = "";
24. z:
25. for(int x = 0; x < 3; x++) {
26. for(int y = 0; y < 2; y++) {
27. if(x==1) break;
28. if(x==2 && y==1) break z;
29. o = o + x + y;
30. }
31. }
32. System.out.println(o);
33. }

```

What is the result when the go() method is invoked?

- A. 00
- B. 0001
- C. 000120

- D. 00012021
- E. Compilation fails.
- F. An exception is thrown at runtime.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Las salidas del codigo anterior son:

|           |   |   |          |
|-----------|---|---|----------|
| variables | x | y | o        |
| valores   | 0 | 0 | "00"     |
|           | 0 | 1 | "0001"   |
|           | 2 | 0 | "000120" |

### QUESTION 107

Given:

```

12. public class Test {
13. public enum Dogs {collie, harrier};
14. public static void main(String [] args) {
15. Dogs myDog = Dogs.collie;
16. switch (myDog) {
17. case collie:
18. System.out.print("collie ");
19. case harrier:
20. System.out.print("harrier ");
21. }
22. }
23. }

```

What is the result?

- A. collie
- B. harrier
- C. Compilation fails.
- D. collie harrier
- E. An exception is thrown at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**D** es correcto. La sintaxis es la correcta y como no existe ningun break; en la declaración switch el codigo imprime la linea 18 "collie" y la linea 20 "harrier":

### QUESTION 108

Click the Exhibit button. Given:

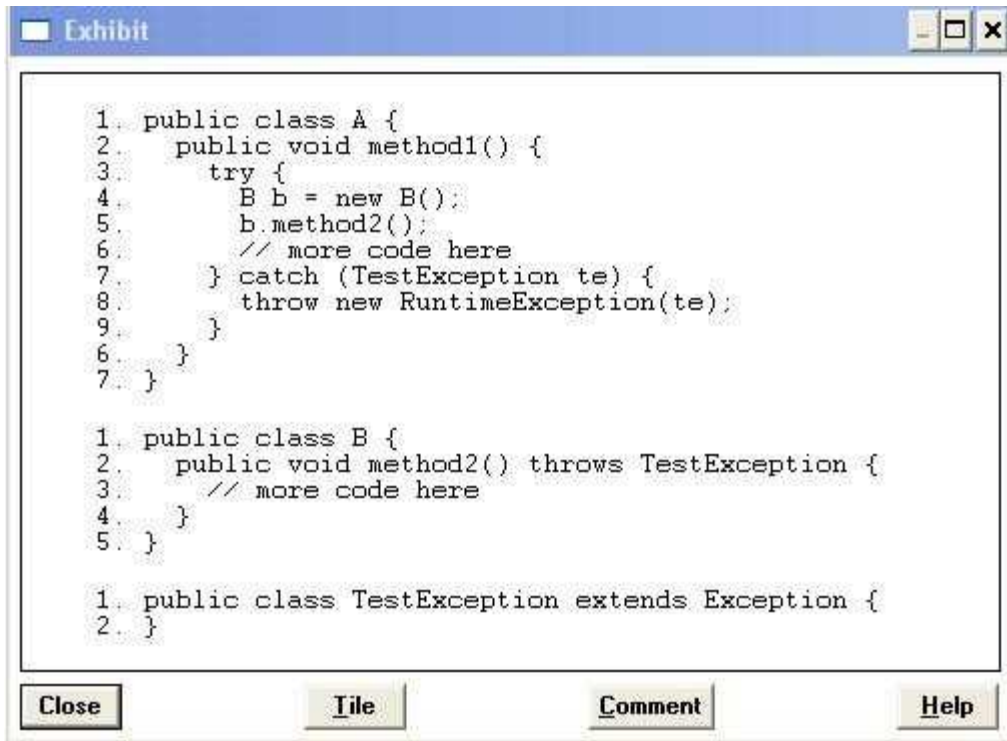
```

31. public void method() {

```

32. A a = new A();  
33. a.method1();  
34. }

Which statement is true if a TestException is thrown on line 3 of class B?



```
1. public class A {
2. public void method1() {
3. try {
4. B b = new B();
5. b.method2();
6. // more code here
7. } catch (TestException te) {
8. throw new RuntimeException(te);
9. }
10. }
11. }

1. public class B {
2. public void method2() throws TestException {
3. // more code here
4. }
5. }

1. public class TestException extends Exception {
2. }
```

- A. Line 33 must be called within a try block.
- B. The exception thrown by method1 in class A is not required to be caught.
- C. The method declared on line 31 must be declared to throw a RuntimeException.
- D. On line 5 of class A, the call to method2 of class B does not need to be placed in a try/catch block.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La excepción si se lanza en el método "method2" es reenviada a su metodo llamante en este caso method1() el cual la trata, por ello no se necesita capturar.

#### QUESTION 109

Given:

```
1. public class Boxer1{
2. Integer i;
3. int x;
4. public Boxer1(int y) {
5. x = i+y;
6. System.out.println(x);
```



```

7. }
8. public static void main(String[] args) {
9. new Boxer1(new Integer(4));
10. }
11. }

```

What is the result?

- A. The value "4" is printed at the command line.
- B. Compilation fails because of an error in line 5.
- C. Compilation fails because of an error in line 9.
- D. A NullPointerException occurs at runtime.
- E. A NumberFormatException occurs at runtime.
- F. An IllegalStateException occurs at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el anterior código ocurre una NullPointerException en la línea 19 ya que la variable "i" al ser de tipo Integer es inicializada con un valor "Null" y al intentarla sumar con la variable "y" nos genera este error.

#### QUESTION 110

Given:

```

11. static class A {
12. void process() throws Exception { throw 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.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el programa anterior, crean dos clases anidadas estáticas A y B dentro de otra clase externa, en el main línea 18 se crea una clase anonima ("B") y se llama a su metodo process(); el cual imprime "B" ya que apesar de ser una subclase de A como es de tipo "B" llama a su método en la línea 15 y no llama al método de la línea 12.

#### QUESTION 111

Given:

```
1. public class Venus {
2. public static void main(String[] args) {
3. int [] x = {1,2,3};
4. int y[] = {4,5,6};
5. new Venus().go(x,y);
6. }
7. void go(int[]... z) {
8. for(int[] a : z)
9. System.out.print(a[0]);
10. }
11. }
```

What is the result?

- A. 1
- B. 12
- C. 14
- D. 123
- E. Compilation fails.
- F. An exception is thrown at runtime.

**Correct Answer:** C

**Section:** (none)

**Explanation**

#### Explanation/Reference:

El código crea en la línea 13 y 14 dos arreglos de enteros "x" y "y" en la línea 5 se crea una clase anónima "Venus" y se llama su método "go()" el cual acepta como argumentos uno o muchos arreglos de enteros en este caso se le entregan los dos : "x" y "y". en la línea 8 el for recorre el arreglo de arreglos z[][] e imprime para ambos casos la posición 0 que es : 1 y 4.

#### QUESTION 112

Given:

```
10. public class Foo {
11. static int[] a;
12. static { a[0]=2; }
13. public static void main(String[] args) {}
14. }
```

Which exception or error will be thrown when a programmer attempts to run this code?

- A. java.lang.StackOverflowError
- B. java.lang.IllegalStateException
- C. java.lang.ExceptionInInitializerError
- D. java.lang.ArrayIndexOutOfBoundsException

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El código arroja un error en la línea 12 cuando se intenta agregar un ítem a un arreglo que nunca se dimensionó o instanció.

#### QUESTION 113

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.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Se crea una clase SubX que extiende otra clase creada "X", en el main se crea una clase anónima de SubXy se llama su método foo(), el cual llama en la línea 15 el constructor de la clase "X" imprimiendo "X", luego en la línea 16 se evalúa si el if es verdadero y como por defecto es "true" se lanza una RuntimeException y termina el programa porque el método llamante "main" no trata la excepción.

#### QUESTION 114

DRAG DROP

Click the Task button.

Drag and Drop

Place the Fragments into the program, so that the program will get lines from a text file, display them, and then close all the resources.

**Program**

```
import java.io.*

public class ReadFile {
 public static void main(String [] args) {
 try {
 File ? = new File("MyText.txt");
 ? = new ? (x1);
 ? x4 = new ? (x2);
 String x3 = null;
 while ((x3 = ? . ? ()) != null) {
 System.out.println(x3);
 } ? . ? ();
 } catch (Exception ex) {
 ex.printStackTrace();
 }
 }
}
```

**Code Fragments**

- BufferedReader
- StreamReader
- FileReader
- readLine
- readLn
- read
- closeFile
- close
- x1
- x2
- x3
- x4

Done

- A. revisar...
- B. revisar...
- C. revisar...
- D. revisar...

**Correct Answer:** A  
**Section:** (none)  
**Explanation**

**Explanation/Reference:**

Drag and Drop

Place the Fragments into the program, so that the program will get lines from a text file, display them, and then close all the resources.

**Program**

```
import java.io.*

public class ReadFile {
 public static void main(String [] args) {
 try {
 File x1 = new File("MyText.txt");
 FileReader x2 = new FileReader(x1);
 BufferedReader x4 = new BufferedReader(x2);
 String x3 = null;
 while ((x3 = x4.readLine()) != null) {
 System.out.println(x3);
 } x4.close();
 } catch (Exception ex) {
 ex.printStackTrace();
 }
 }
}
```

**Code Fragments**

- BufferedReader
- StreamReader
- FileReader
- readLine
- readLn
- read
- closeFile
- close
- x1
- x2
- x3
- x4

Done

**QUESTION 115****DRAG DROP**

Click the Task button.

Given: 

```
public class Doubler {
 public static int doubleMe(Holder h) {
 return h.getAmount() * 2;
 }
}
```

and: 

```
public class Holder {
 int amount = 10;
 public void doubleAmount(){ amount = Doubler.doubleMe(this);}
 public int getAmount(){ return amount;}
 //more code here
}
```

Place the code fragments in position to reduce the coupling between Doubler and Holder.

```
public class Doubler {
 public static int doubleMe(Place here h) {
 return Place here * 2;
 }
}
```

```
public class Holder {
 int amount = 10;
 public void doubleAmount(){ amount = Doubler.doubleMe(Place here);}
 public int getAmount(){ return amount;}
 //more code here
}
```

**Code Fragments**

|               |        |      |         |
|---------------|--------|------|---------|
| void          | Holder | int  | Doubler |
| h.getAmount() | h      | this | amount  |

Done

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Given: 

```
public class Doubler {
 public static int doubleMe(Holder h) {
 return h.getAmount() * 2;
 }
}
```

and: 

```
public class Holder {
 int amount = 10;
 public void doubleAmount(){ amount = Doubler.doubleMe(this);}
 public int getAmount(){ return amount;}
 //more code here
}
```

Place the code fragments in position to reduce the coupling between Doubler and Holder.

```
public class Doubler {
 public static int doubleMe(int h) {
 return h * 2;
 }
}

public class Holder {
 int amount = 10;
 public void doubleAmount(){ amount = Doubler.doubleMe(amount);}
 public int getAmount(){ return amount;}
 //more code here
}
```

#### Code Fragments

|               |        |      |         |
|---------------|--------|------|---------|
| void          | Holder | int  | Doubler |
| h.getAmount() | h      | this | amount  |

Done

Cuando se este desarrollando un proyecto java y/o cualquier proyecto orientado a objetos, lo ideal es que este proyecto tenga: Bajo acoplamiento (loosely coupled) y alta cohesión (high cohesion).

### Coupling

El acoplamiento es el grado de conocimiento que una clase A tiene sobre una clase B. Lo ideal es conseguir que la clase A, sólo conozca de la clase B lo necesario para que la clase A pueda hacer uso de los métodos de la clase B, pero no conozca nada a cerca de cómo estos métodos están implementados. Por tanto es nuestra labor definir el interfaz de la clase B de manera que, únicamente tengamos como públicos aquellos métodos con los que nos interesa que el resto de las clases/objetos puedan interactuar.

Por supuesto, es un requisito fundamental para un buen diseño orientado a objetos, que todas las variables de instancia de una clase sean privadas y la única forma de acceder a ellas sea a través de los métodos getter y setter.

Cuanto menos cosas conozca la clase A sobre la clase B, menor será su acoplamiento.

### Cohesion

Mientras que el coupling se refiere a cómo interactúa una clase con la otra, la cohesión indica el grado de especialización de una clase. El objetivo es enfocar de la forma más precisa posible el propósito de la clase. Cuanto más enfoquemos el propósito de la clase, mayor será su cohesión.

Los beneficios que proporciona la cohesión es que hará que nuestras clases sean más fáciles de mantener, más fáciles de entender y además podremos reutilizar nuestras clases de una manera mucho más cómoda y eficiente.

### QUESTION 116

DRAG DROP

Click the Task button.



**Drag and Drop**

Place the Types in one of the Type columns, and the Relationships in the Relationship column, to define appropriate has-a and is-a relationships.

| Type       | Relationship | Type             | Relationships | Types     |
|------------|--------------|------------------|---------------|-----------|
| Place here | Place here   | Animal           | is-a          | Dog       |
| Forest     | Place here   | Place here       | has-a         | Side      |
| Rectangle  | Place here   | Place here       |               | Tail      |
| Place here | Place here   | Programming Book |               | Square    |
|            |              |                  |               | Tree      |
|            |              |                  |               | Book      |
|            |              |                  |               | Java Book |
|            |              |                  |               | Pen       |

Done

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**Drag and Drop**

Place the Types in one of the Type columns, and the Relationships in the Relationship column, to define appropriate has-a and is-a relationships.

| Type      | Relationship | Type             | Relationships | Types     |
|-----------|--------------|------------------|---------------|-----------|
| Dog       | is-a         | Animal           | is-a          | Dog       |
| Forest    | has-a        | Tree             | has-a         | Side      |
| Rectangle | has-a        | Side             |               | Tail      |
| Java Book | is-a         | Programming Book |               | Square    |
|           |              |                  |               | Tree      |
|           |              |                  |               | Book      |
|           |              |                  |               | Java Book |
|           |              |                  |               | Pen       |

Done

#### QUESTION 117

DRAG DROP

Click the Task button.

Drag and Drop

Place code fragments into position so the output is: The quantity is 420

```

 Place here update(int quantity, int adjust) {
 Place here
}

public void callUpdate() {
 int quant = 100;
 Place here
 System.out.println("The quantity is " + quant);
}

```

**Code Fragments**

|             |                               |                                                   |
|-------------|-------------------------------|---------------------------------------------------|
| public int  | quantity = quantity + adjust; | update(quant, 320);                               |
| public void | quant = update(quant, 320);   | quantity = quantity + adjust;<br>return quantity; |

Done

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**



Drag and Drop

Place code fragments into position so the output is: The quantity is 420

```
public int update(int quantity, int adjust) {
 quantity = quantity + adjust;
 return quantity;
}

public void callUpdate() {
 int quant = 100;
 quant = update(quant, 320);
 System.out.println("The quantity is " + quant);
}
```

**Code Fragments**

|             |                               |                                                   |
|-------------|-------------------------------|---------------------------------------------------|
| public int  | quantity = quantity + adjust; | update(quant, 320);                               |
| public void | quant = update(quant, 320);   | quantity = quantity + adjust;<br>return quantity; |

Done

**QUESTION 118**  
DRAG DROP

Click the Task button.

Drag and Drop

Place the lines in the correct order to complete the enum.

enum Element {

1st

2nd

3rd

4th

5th

Lines

public String info() { return "element"; }

}

FIRE { public String info() { return "Hot"; } }

EARTH, WIND

}

Done

- A.
- B.
- C.
- D.

Correct Answer: A  
Section: (none)  
Explanation

Explanation/Reference:

Drag and Drop

Place the lines in the correct order to complete the enum.

enum Element {

EARTH WIND

FIRE { public String info() { return "Hot"; }

};

public String info() { return "element"; }

}

Lines

public String info() { return "element"; }

};

FIRE { public String info() { return "Hot"; }

EARTH WIND

}

Done

**QUESTION 119**  
DRAG DROP

Click the Task button.



Drag and Drop

Given the class definitions:

```
class Animal { }
class Dog extends Animal { }
```

and the code:

```
public void go() {
 ArrayList<Dog> aList = new ArrayList<Dog>();
 takeList(aList);
}
// insert definition of the takeList() method here
```

Place the correct Compilation Result on each takeList() method definition to indicate whether or not the go() method would compile given that definition.

**takeList() Method Definition**

☐ public void takeList(ArrayList list) { }

☐ public void takeList(ArrayList<Animal> list) { }

☐ public void takeList(ArrayList<? extends Animal> list) { }

☐ public void takeList(ArrayList<?> list) { }

☐ public void takeList(ArrayList<Object> list) { }

**Compilation Result**

☐ Compilation succeeds.

☐ Compilation fails

Done

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Drag and Drop

Given the class definitions:

```
class Animal { }
class Dog extends Animal { }
```

and the code:

```
public void go() {
 ArrayList<Dog> aList = new ArrayList<Dog>();
 takeList(aList);
}
// insert definition of the takeList() method here
```

Place the correct Compilation Result on each takeList() method definition to indicate whether or not the go() method would compile given that definition.

**takeList() Method Definition**

Compilation succeeds.

Compilation fails.

Compilation succeeds.

Compilation succeeds.

Compilation fails.

**Compilation Result**

Compilation succeeds.

Compilation fails.

Done

#### QUESTION 120

Which Man class properly represents the relationship "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>; }

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La relacion "tiene un" se representa con una variable dentro de una clase, en este caso una variable tipo "Dog" dentro de la clase "Man"

#### QUESTION 121

A company has a business application that provides its users with many different reports: receivables reports, payables reports, revenue projects, and so on. The company has just purchased some new, state-of-the-art, wireless printers, and a programmer has been assigned the task of enhancing all of the reports to use not only the company's old printers, but the new wireless printers as well. When the programmer starts looking into the application, the programmer discovers that because of the design of the application, it is necessary to make changes to each report to support the new printers. Which two design concepts most likely explain this situation? (Choose two.)

- A. Inheritance
- B. Low cohesion
- C. Tight coupling
- D. High cohesion
- E. Loose coupling
- F. Object immutability

**Correct Answer:** BC

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Cuando se este desarrollando un proyecto java y/o cualquier proyecto orientado a objetos, lo ideal es que este proyecto tenga: Bajo acoplamiento (loosely coupled) y alta cohesión (high cohesion).

**Coupling**

El acoplamiento es el grado de conocimiento que una clase A tiene sobre una clase B. Lo ideal es conseguir que la clase A, sólo conozca de la clase B lo necesario para que la clase A pueda hacer uso de los métodos de la clase B, pero no conozca nada a cerca de cómo estos métodos están implementados. Por tanto es nuestra labor definir el interfaz de la clase B de manera que, únicamente tengamos como públicos aquellos métodos con los que nos interesa que el resto de las clases/objetos puedan interactuar.

Por supuesto, es un requisito fundamental para un buen diseño orientado a objetos, que todas las variables de instancia de una clase sean privadas y la única forma de acceder a ellas sea a través de los métodos getter y setter.

Cuantas menos cosas conozca la clase A sobre la clase B, menor será su acoplamiento.

**Cohesion**

Mientras que el coupling se refiere a cómo interactúa una clase con la otra, la cohesión indica el grado de especialización de una clase. El objetivo es enfocar de la forma más precisa posible el propósito de la clase. Cuanto más enfoquemos el propósito de la clase, mayor será su cohesión.

Los beneficios que proporciona la cohesión es que hará que nuestras clases sean más fáciles de mantener, más fáciles de entender y además podremos reutilizar nuestras clases de una manera mucho más cómoda y eficiente.

**QUESTION 122**

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() { }

**Correct Answer:** ABEF

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Para reescribir un método el método que reescribe debe tener un modificador de acceso mas debil o igual al método reescrito. Por ello las respuestas correctas son **A** , **B**, **E**, **F**.

### QUESTION 123

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.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La compilación falla en la línea 19 porque para llamar a un constructor de la misma clase debe utilizarse la palabra clave "this" y debe estar en la primera línea del método.

### QUESTION 124

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 commission;
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.commission = commission;`
- 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);`

**Correct Answer:** AE

**Section:** (none)

**Explanation**

#### **Explanation/Reference:**

Solo compila las opciones **A** y la **B** porque por defecto una subclase en cualquiera de sus constructores ejecuta el constructor `super()`, al no encontrarlo en "Employee" genera un error de compilación que solo se corrige si se adiciona en el constructor de `SalesPerson` una llamada a `super(String name, double baseSalary)`. de esta forma no se llamara al `super()` y correrá normalmente.

### QUESTION 125

A team of programmers is reviewing a proposed API for a new utility class. After some discussion, they realize that they can reduce the number of methods in the API without losing any functionality. If they implement the new design, which two OO principles will they be promoting?

- A. Looser coupling
- B. Tighter coupling
- C. Lower cohesion



- D. Higher cohesion
- E. Weaker encapsulation
- F. Stronger encapsulation

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Cuando se este desarrollando un proyecto java y/o cualquier proyecto orientado a objetos, lo ideal es que este proyecto tenga: Bajo acoplamiento (loosely coupled) y alta cohesión (high cohesion).

**Coupling**

El acoplamiento es el grado de conocimiento que una clase A tiene sobre una clase B. Lo ideal es conseguir que la clase A, sólo conozca de la clase B lo necesario para que la clase A pueda hacer uso de los métodos de la clase B, pero no conozca nada a cerca de cómo estos métodos están implementados. Por tanto es nuestra labor definir el interfaz de la clase B de manera que, únicamente tengamos como públicos aquellos métodos con los que nos interesa que el resto de las clases/objetos puedan interactuar.

Por supuesto, es un requisito fundamental para un buen diseño orientado a objetos, que todas las variables de instancia de una clase sean privadas y la única forma de acceder a ellas sea a través de los métodos getter y setter.

Cuanto menos cosas conozca la clase A sobre la clase B, menor será su acoplamiento.

**Cohesion**

Mientras que el coupling se refiere a cómo interactúa una clase con la otra, la cohesión indica el grado de especialización de una clase. El objetivo es enfocar de la forma más precisa posible el propósito de la clase. Cuanto más enfoquemos el propósito de la clase, mayor será su cohesión.

Los beneficios que proporciona la cohesión es que hará que nuestras clases sean más fáciles de mantener, más fáciles de entender y además podremos reutilizar nuestras clases de una manera mucho más cómoda y eficiente.

**QUESTION 126**

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 ExtendedA(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 statement 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 modifier.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

la sintaxis es correcta e imprime 420 en la salida.

#### QUESTION 127

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 = (Barn) build1;

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.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El código tiene 2 errores:

1. la línea 10 genera un error en tiempo de ejecución, el objeto "build1 no se puede transformar en (Barn), puesto que la clase Building no conoce a la clase Barn apesar de esto deja ejecutar y compila.

2. El error que hay que eliminar para que compile esta la linea 12, el objeto "build2" nunca se podrá transformar en uno de tipo String.

En la linea 13 se transforma correctamente el objeto barn a Building puesto que la clase build si conoce a la clase Building.

#### QUESTION 128

Given:

```
1. public class TestOne {
2. public static void main (String[] args) throws Exception {
3. Thread.sleep(3000);
4. System.out.println("sleep");
5. }
6. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes normally and prints "sleep".
- D. The code executes normally, but nothing is printed.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La sintaxis del código es la correcta, el hilo principal (main) duerme 3 segundos antes de continuar.

#### QUESTION 129

Given:

```
1. public class Threads4 {
2. public static void main (String[] args) {
3. new Threads4().go();
4. }
5. public void go() {
6. Runnable r = new Runnable() {
7. public void run() {
8. System.out.print("foo");
9. }
10. };
11. Thread t = new Thread(r);
12. t.start();
13. t.start();
```

14. }

15. }

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes normally and prints "foo".
- D. The code executes normally, but nothing is printed.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Se genera una excepción en tiempo de ejecución pues no se puede llamar dos veces al método start(); consecutivamente esto genera un IllegalStateException que es un error que se genera cuando un hilo no está en un estado apropiado para recibir la solicitud, en este caso no puede recibir la segunda solicitud "start()". hasta que termine de ejecutarse.

#### QUESTION 130

Which two statements are true? (Choose two.)

- A. It is possible for more than two threads to deadlock at once.
- B. The JVM implementation guarantees that multiple threads cannot enter into a deadlocked state.
- C. Deadlocked threads release once their sleep() method's sleep duration has expired.
- D. Deadlocking can occur only when the wait(), notify(), and notifyAll() methods are used incorrectly.
- E. It is possible for a single-threaded application to deadlock if synchronized blocks are used incorrectly.
- F. If a piece of code is capable of deadlocking, you cannot eliminate the possibility of deadlocking by inserting invocations of Thread.yield().

**Correct Answer:** AF

**Section:** (none)

**Explanation**

**Explanation/Reference:**

#### QUESTION 131

Given:

1. public class Threads3 implements Runnable {
2.   public void run() {
3.       System.out.print("running");
4.   }
5.   public static void main(String[] args) {
6.       Thread t = new Thread(new Threads3());
7.       t.run();
8.       t.run();
9.       t.start();
10.   }

11. }

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes and prints "running".
- D. The code executes and prints "runningrunning".
- E. The code executes and prints "runningrunningrunning".

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La sintaxis es la correcta ejecuta el método run() tres veces imprimiendo "runningrunningrunning".

### QUESTION 132

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. }
```

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

- 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);

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Para usar un método de otra clase y de otro paquete se debe nombrar primero el <nombre del paquete > "." <nombre de la clase> "." <nombre del método>

QUESTION 133

A developer is creating a class Book, that needs to access class Paper. The Paper class is deployed in a JAR named myLib.jar. Which three, taken independently, will allow the developer to use the Paper class while compiling the Book class? (Choose three.)

- A. The JAR file is located at \$JAVA_HOME/jre/classes/myLib.jar.
- B. The JAR file is located at \$JAVA_HOME/jre/lib/ext/myLib.jar..
- C. The JAR file is located at /foo/myLib.jar and a classpath environment variable is set that includes /foo/myLib.jar/Paper.class.
- D. The JAR file is located at /foo/myLib.jar and a classpath environment variable is set that includes /foo/myLib.jar.
- E. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -cp /foo/myLib.jar/ Paper Book.java.
- F. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -d /foo/myLib.jar Book.java
- G. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -classpath /foo/myLib.jar Book.java

Correct Answer: BDG

Section: (none)

Explanation

Explanation/Reference:

Explanation:

QUESTION 134

Given:

```

11. class Snoochy {
12.     Boochy booch;
13.     public Snoochy() { booch = new Boochy(this); }
14. }
15.
16. class Boochy {
17.     Snoochy snooch;
18.     public Boochy(Snoochy s) { snooch = s; }
19. } And the statements:
21.     public static void main(String[] args) {
22.         Snoochy snoog = new Snoochy();
23.         snoog = null;
24. // more code here
25. }
```

Which statement is true about the objects referenced by snoog, snooch, and booch immediately after line 23 executes?

- A. None of these objects are eligible for garbage collection.
- B. Only the object referenced by booch is eligible for garbage collection.
- C. Only the object referenced by snoog is eligible for garbage collection.
- D. Only the object referenced by snooch is eligible for garbage collection.
- E. The objects referenced by snooch and booch are eligible for garbage collection.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

Al hacer en la linea 23 a snogg null todas las variables que pertenezcan a el objeto creado tipo Snoochy en la linea 22 quedan sin referenciar y pueden ser eliminados es el caso de la variable de referencia booch. Siendo asi tanto la variable de objeto "snoog" como "booch" pueden ser apropiados para que el gc los deseche.

QUESTION 135

Given:

```
3. public class Batman {  
4.     int squares = 81;  
5.     public static void main(String[] args) {  
6.         new Batman().go();  
7.     }  
8.     void go() {  
9.         incr(++squares);  
10.    System.out.println(squares);  
11.    }  
12.    void incr(int squares) { squares += 10; }  
13. }
```

What is the result?

- A. 81
- B. 82
- C. 91
- D. 92
- E. Compilation fails.
- F. An exception is thrown at runtime.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

El resultado de la variable de instancia squares es 82 valor que toma despues del incremento en la linea 9. La variable local squares es independiente de la de instancia y almacena el valor 92. el valor que se imprime es el de la variable de instancia por ello es 82.

QUESTION 136

Given classes defined in two different files:

```
1. package util;  
2. public class BitUtils {  
3.     private static void process(byte[] b) {}  
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. }

```

What is required at line 5 in class SomeApp to use the process 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.

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

La clase SomeApp no puede usar directamente el método process() pues esta marcado como privado o sea se accede solo en su clase.

QUESTION 137

A UNIX user named Bob wants to replace his chess program with a new one, but he is not sure where the old one is installed. Bob is currently able to run a Java chess program starting from his home directory /home/bob using the command: java -classpath /test:/home/bob/downloads/*.jar games.Chess Bob's CLASSPATH is set (at login time) to: /usr/lib:/home/bob/classes:/opt/java/lib:/opt/java/lib/*.jar What is a possible location for the Chess.class file?

- A. /test/Chess.class
- B. /home/bob/Chess.class
- C. /test/games/Chess.class
- D. /usr/lib/games/Chess.class
- E. /home/bob/games/Chess.class
- F. inside jarfile /opt/java/lib/Games.jar (with a correct manifest)
- G. inside jarfile /home/bob/downloads/Games.jar (with a correct manifest)

Correct Answer: C

Section: (none)

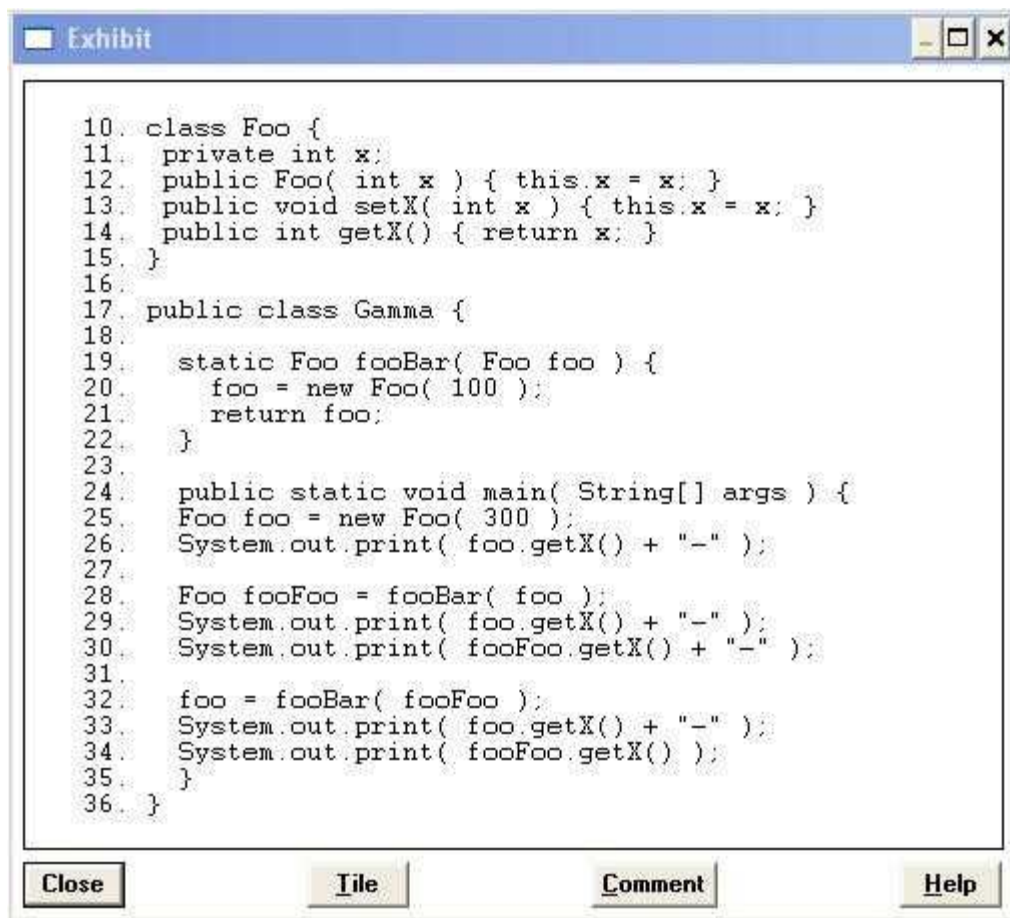
Explanation

Explanation/Reference:

Si el usuario ejecuta el comando java -classpath /test:/home/bob/downloads/*.jar games.Chess para arrancar el juego, el comando -classpath nos dice que esta dentro de /test y ademas que esta dentro de un directorio games/ y su clase se llama Chess.class por lo tanto al unir los dos tramos nos da: /test/games/Chess.class

QUESTION 138

Click the Exhibit button. What is the output of the program shown in the exhibit?



```
10. class Foo {
11.     private int x;
12.     public Foo( int x ) { this.x = x; }
13.     public void setX( int x ) { this.x = x; }
14.     public int getX() { return x; }
15. }
16.
17. public class Gamma {
18.
19.     static Foo fooBar( Foo foo ) {
20.         foo = new Foo( 100 );
21.         return foo;
22.     }
23.
24.     public static void main( String[] args ) {
25.         Foo foo = new Foo( 300 );
26.         System.out.print( foo.getX() + "-" );
27.
28.         Foo fooFoo = fooBar( foo );
29.         System.out.print( foo.getX() + "-" );
30.         System.out.print( fooFoo.getX() + "-" );
31.
32.         foo = fooBar( fooFoo );
33.         System.out.print( foo.getX() + "-" );
34.         System.out.print( fooFoo.getX() );
35.     }
36. }
```

- A. 300-100-100-100-100
- B. 300-300-100-100-100
- C. 300-300-300-100-100
- D. 300-300-300-300-100

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Tabla de valores paso a paso:

linea	foo.x	fooFoo.x	foo.x(local)
25	300	---	
28	300	100	100
32	100	100	100

QUESTION 139

Given the following directory structure: bigProject |--source | |--Utils.java | |--classes |-- And the following command line invocation: javac -d classes source/Utils.java Assume the current directory is bigProject, what is the result?

- A. If the compile is successful, Utils.class is added to the source directory.
- B. The compiler returns an invalid flag error.
- C. If the compile is successful, Utils.class is added to the classes directory.
- D. If the compile is successful, Utils.class is added to the bigProject directory.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

C es correcto. El parametro -d se usa para indicar donde se deben alojar las -class compiladas por lo tanto si compila el código se almacenara en /classes.

QUESTION 140

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.print("f-p ");
13.        if(w instanceof Fish) System.out.print("w-f ");
14.        if(b instanceof Fish) System.out.print("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.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Para que una subclase sea instancia de "instanceof" otra superclase o de interface debe o bien extenderla o implementar la interfaz.

El objetivo del operador instanceof es conocer si un objeto es de un tipo determinado. Por tipo nos referimos a clase o interfaz (interface), es decir si el objeto pasaría el test ES-UN para esa clase o ese interfaz, especificado a la derecha del operador.

QUESTION 141

Given:

```

1. public class Breaker2 {
2.     static String o = "";
3.     public static void main(String[] args) {
4.         z:
5.         for(int x = 2; x < 7; x++) {
6.             if(x==3) continue;
7.             if(x==5) break z;
8.             o = o + x;
9.         }
10.        System.out.println(o);
11.    }
12.}

```

What is the result?

- A. 2
- B. 24
- C. 234
- D. 246
- E. 2346
- F. Compilation fails.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es la correcta y solo imprime los valores unidos "2" y "4" pues para los valores x=3 el for salta a la siguiente iteración y para el valor i="5" la instrucción "break" hace que se salga completamente del ciclo "for".

QUESTION 142

Given:

```

11. public void testIfA() {
12.     if (testIfB("True")) {
13.         System.out.println("True");
14.     } else {
15.         System.out.println("Not true");
16.     }
17.}

18. public Boolean testIfB(String str) {
19.     return Boolean.valueOf(str);

```

20. }

What is the result when method testIfA is invoked?

- A. True
- B. Not true
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error at line 12.
- E. Compilation fails because of an error at line 19.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La clase o el wrap : "Boolean" posee métodos para pasar de booleano a String : toString(boolean b) y para pasar un String a boolean: valueOf(). Por ello el código anterior en la línea 12 testIfB da "true" e imprime "True" en la siguiente línea.

Class Boolean

```
public final class Boolean
extends Object
implements Serializable
```

The Boolean class wraps a value of the primitive type boolean in an object. An object of type Boolean contains a single field whose type is boolean.

In addition, this class provides many methods for converting a boolean to a String and a String to a boolean, as well as other constants and methods useful when dealing with a boolean.

QUESTION 143

Given:

1. public class Donkey {
2. public static void main(String[] args) {
3. boolean assertsOn = false;
4. assert (assertsOn) : assertsOn = true;
5. if(assertsOn) {
6. System.out.println("assert is on");
7. }
8. }
9. }

If class Donkey is invoked twice, the first time without assertions enabled, and the second time with assertions enabled, what are the results?

- A. no output
- B. no output
assert is on
- C. assert is on
- D. no output
An AssertionError is thrown.
- E. assert is on

An AssertionError is thrown.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Explanation:

QUESTION 144

Given:

```
31. // some code here
32. try {
33.     // some code here
34. } catch (SomeException se) {
35.     // some code here
36. } finally {
37.     // some code here
38. }
```

Under which three circumstances will the code on line 37 be executed? (Choose three.)

- A. The instance gets garbage collected.
- B. The code on line 33 throws an exception.
- C. The code on line 35 throws an exception.
- D. The code on line 31 throws an exception.
- E. The code on line 33 executes successfully.

Correct Answer: BCE

Section: (none)

Explanation

Explanation/Reference:

B, C y D son correctos La clausula finally ejecuta si o si al final de un bloque try-catch.

D es incorrecto porque la Exception esta fuera del bloque try -catch y nunca se ejecuta el finally.

QUESTION 145

Given:

```
22. public void go() {
23.     String o = "";
24.     z:
25.     for(int x = 0; x < 3; x++) {
26.         for(int y = 0; y < 2; y++) {
27.             if(x==1) break;
28.             if(x==2 && y==1) break z;
29.             o = o + x + y;
```

```

30.     }
31. }
32.     System.out.println(o);
33. }

```

What is the result when the go() method is invoked?

- A. 00
- B. 0001
- C. 000120
- D. 00012021
- E. Compilation fails.
- F. An exception is thrown at runtime.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Las salidas del código anterior son:

variables	x	y	o
valores	0	0	"00"
	0	1	"0001"
	2	0	"000120"

QUESTION 146

Given:

```

11. static void test() {
12.     try {
13.         String x = null;
14.         System.out.print(x.toString() + " ");
15.     }
16.     finally { System.out.print("finally "); }
17. }
18. public static void main(String[] args) {
19.     try { test(); }
20.     catch (Exception ex) { System.out.print("exception "); }
21. }

```

What is the result?

- A. null
- B. finally
- C. null finally
- D. Compilation fails.
- E. finally exception

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

Los wrap en su método toString() lanzan una NullPointerException si el valor de la variable es null, por ello la única sentencia que se ejecuta es el finally del método test. La "exception" como no la captura el método test() con algún "catch" la trata de nuevo el método main por ello se imprime "finally exception".

QUESTION 147

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;

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

El cast no se puede realizar en la opción **B** porque "x" es tipo Beta y esta clase conoce a la clase Delta.

QUESTION 148

Given:

```
33. try {
34.     // some code here
35. } catch (NullPointerException e1) {
36.     System.out.print("a");
37. } catch (Exception e2) {
38.     System.out.print("b");
39. } finally {
40.     System.out.print("c");
41. }
```

If some sort of exception is thrown at line 34, which output is possible?

- A. a
- B. b
- C. c
- D. ac
- E. abc

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Si se lanzara una excepcion en la linea 34 sería capturada por el catch "NullPointerException" de la linea 34 u otra "Exception" que se capturaría en el catch de la linea 37. la sentencia finally siempre se ejecutaría entonces se imprimiría o bien ac o bc por ello la respuesta correcta es la **D**.

QUESTION 149

Given:

```
11. public class Test {  
12.     public enum Dogs {collie, harrier, shepherd};  
13.     public static void main(String [] args) {  
14.         Dogs myDog = Dogs.shepherd;  
15.         switch (myDog) {  
16.             case collie:  
17.                 System.out.print("collie ");  
18.             case default:  
19.                 System.out.print("retriever ");  
20.             case harrier:  
21.                 System.out.print("harrier ");  
22.         }  
23.     }  
24. }
```

What is the result?

- A. harrier
- B. shepherd
- C. retriever
- D. Compilation fails.
- E. retriever harrier
- F. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

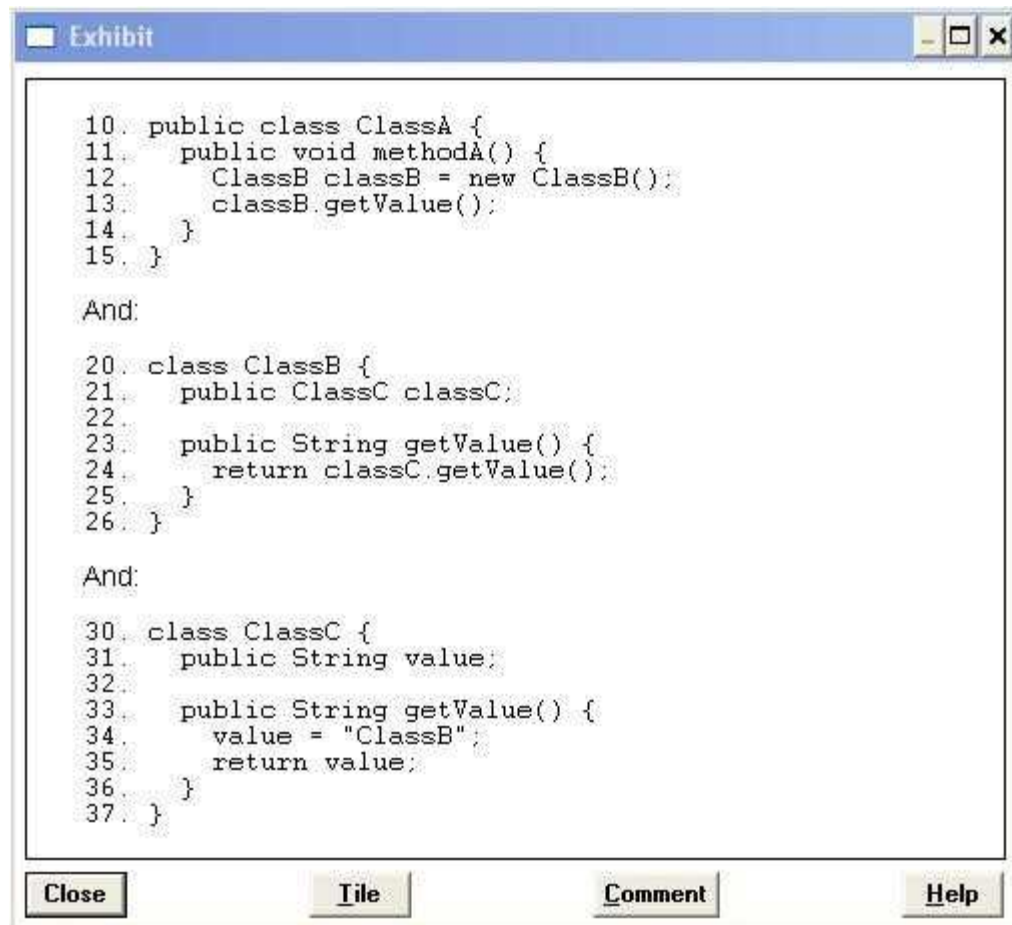
Explanation

Explanation/Reference:

A pesar que un "default" se puede escribir en cualquier posicion del "case" esta mal escrito y genera un error en la linea 18, solo debe ir la palabra reservada "default". Si estuviera bien escrito el resultado sería "retriever harrier"

QUESTION 150

Click the Exhibit button. Given: `ClassA a = new ClassA(); a.methodA();` What is the result?



- A. Compilation fails.
- B. ClassC is displayed.
- C. The code runs with no output.
- D. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

El error de el código anterior se encuentra en la linea 24 de la clase "ClassB" no se puede invocar el método `getValue()` de la clase "ClassC" puesto que no es un método estatico, hay que invocarlo a través de una instancia.

El bloque del método debería quedar así:

```
ClassC miClase= new ClassC();
return miClase.getValue();
```

QUESTION 151

Given:

- ```
11. static void test() throws RuntimeException {
12. try {
```

```

13. System.out.print("test ");
14. throw new RuntimeException();
15. }
16. catch (Exception ex) { System.out.print("exception "); }
17. }
18. public static void main(String[] args) {
19. try { test(); }
20. catch (RuntimeException ex) { System.out.print("runtime "); }
21. System.out.print("end ");
22. }

```

What is the result?

- A. test end
- B. Compilation fails.
- C. test runtime end
- D. test exception end
- E. A Throwable is thrown by main at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El código anterior llama primero el método test() que imprime "test" luego lanza una RuntimeException que es tratada de una vez en su catch ya que el maneja todas las Exception imprimiendo "exception", después finaliza el método main imprimiendo "end". Resultado: test exception end

## QUESTION 152

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("fern"); }` is added to the Plant class.
- E. The code will compile if `public Plant() { Plant("fern"); }` is added to the Plant class.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La subclase Tree por defecto llama el super constructor `super()`; pero este como no existe una opción es añadirlo en Plant y llamar de hay al otro constructor `Plant(String name)`.

#### QUESTION 153

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() ; l.Point p = new l.Point();`

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Para crear una instancia de una clase anidada se llama primero su clase externa seguida de punto y luego el nombre de la clase anidada, luego se crea la instancia como se crean todos los objetos en java.

#### QUESTION 154

Given:

```
10. class Nav{
11. public enum Direction { NORTH, SOUTH, EAST, WEST }
12. }
13. public class Sprite{
14. // insert code here
15. }
```

Which code, inserted at line 14, allows the Sprite class to compile?

- A. `Direction d = NORTH;`

- B. Nav.Direction d = NORTH;
- C. Direction d = Direction.NORTH;
- D. Nav.Direction d = Nav.Direction.NORTH;

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Para referirse aun enum dentro de una clase se cita primero el nombre de la clase seguido de punto y luego el nombre del "enum".

#### QUESTION 155

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\*/ }  
}

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**B** es incorrecto no se puede implementar una clase ella se extiende.

**C** es incorrecto no se puede referir al metodo load() de esa forma: Info.load

**D y E** son ncorrectos porque las clases no se implementan se extienden.

**F** es incorrecto se implementa de forma erronea el metodo load().

#### QUESTION 156

Given:

11. public class Rainbow {

12. public enum MyColor {

13. RED(0xff0000), GREEN(0x00ff00), BLUE(0x0000ff);

14. private final int rgb;

15. MyColor(int rgb) { this.rgb = rgb; }

```

16. public int getRGB() { return rgb; }
17. };
18. public static void main(String[] args) {
19. // insert code here
20. }
21. }

```

Which code fragment, inserted at line 19, allows the Rainbow class to compile?

- A. MyColor skyColor = BLUE;
- B. MyColor treeColor = MyColor.GREEN;
- C. if(RED.getRGB() < BLUE.getRGB()) { }
- D. Compilation fails due to other error(s) in the code.
- E. MyColor purple = new MyColor(0xff00ff);
- F. MyColor purple = MyColor.BLUE + MyColor.RED;

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

B es la forma correcta de establecer un enum MyColor.

#### QUESTION 157

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 \*/ }

**Correct Answer:** BCE

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Para sobrescribir un método el método que sobrescribe debe tener un nivel de acceso mas debil o igual que el sobrescrito. en este caso solo sirven los modificadores default, public y protected.

#### QUESTION 158

Click the Exhibit button. 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.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Asumiendo que en todos los sitios donde dice "//more code here" agregaremos un return con el tipo requerido, el error estaría en la línea 6 de la clase "C" pues al intentar redefinir el método execute() el tipo de dato java.lang.Object no es compatible con el tipo java.lang.String que desea redefinir de la clase "B".

#### QUESTION 159

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 succeeds.
- B. Exactly one class does NOT compile.
- C. Exactly two classes do NOT compile.
- D. Exactly four classes do NOT compile.
- E. Exactly three classes do NOT compile.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Las clases que no compilan son:

La clase B línea 13 genera error porque el método m1() no se implementa.

La clase D línea 15 no compila, el método m1 no se implementa bien debe estar sin parámetros.

Las clases abstractas **E, F y G** si compilan ya que no es necesario que declaren o implementen métodos de la Interfaz puesto que son abstractas.

#### QUESTION 160

Given:

```
1. class Alligator {
2. public static void main(String[] args) {
3. int [][]x = {{1,2}, {3,4,5}, {6,7,8,9}};
4. int [][]y = x;
5. System.out.println(y[2][1]);
6. }
7. }
```

What is the result?

- A. 2
- B. 3
- C. 4
- D. 6
- E. 7
- F. Compilation fails.

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Las posiciones en los arrays comienzan de cero "0" el array x queda de esta forma:

fila0 [1,2]

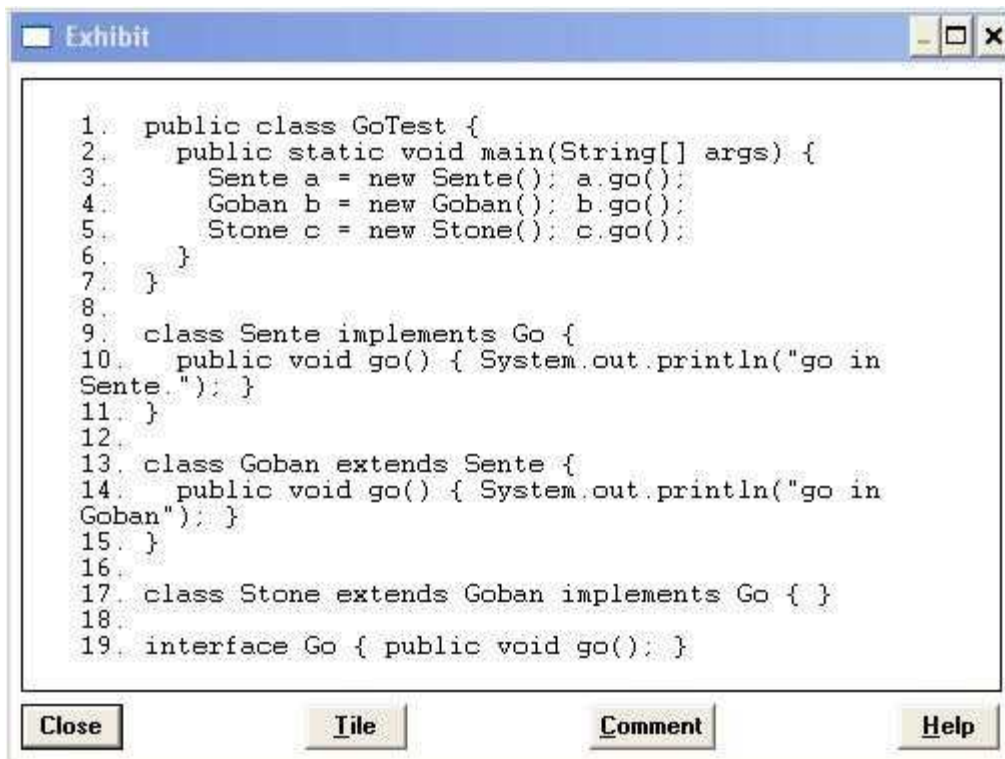
fila1 [3,4,5]

fila2 [6,7,8,9]

Entonces fila 2, columna 1 : "7"

**QUESTION 161**

Click the Exhibit button. What is the result?



- A. go in Goban go in Sente go in Sente
- B. go in Sente go in Sente go in Goban
- C. go in Sente go in Goban go in Goban
- D. go in Goban go in Goban go in Sente
- E. Compilation fails because of an error in line 17.

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

C es correcto. En el ejemplo anterior la línea 3 crea un objeto "Sence" y llama a su método go() imprimiendo "go in Sence".

La línea 4 crea un objeto "Goban" y llama a su método go() imprimiendo "go in Goban".

La línea 5 crea un objeto "Stone" y llama a su método go(), en este caso la clase Stone no necesita implementar el método "go()" de la interfaz "Go" puesto que extiende a otra clase que sí la implementa como es el caso de "Goban", por ello se ejecuta el primer método que implemente Go(); o sea imprime "go in Goban" de nuevo.

**QUESTION 162**

Given:

12. `NumberFormat nf = NumberFormat.getInstance();`

13. `nf.setMaximumFractionDigits(4);`

14. `nf.setMinimumFractionDigits(2);`

15. `String a = nf.format(3.1415926);`

16. `String b = nf.format(2);`

Which two statements are true about the result if the default locale is `Locale.US`? (Choose two.)



- A. The value of b is 2.
- B. The value of a is 3.14.
- C. The value of b is 2.00.
- D. The value of a is 3.141.
- E. The value of a is 3.1415.
- F. The value of a is 3.1416.
- G. The value of b is 2.0000.

**Correct Answer:** CF

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el código anterior se crea un objeto de tipo `NumberFormat` el cual se le da las características actuales de formato del equipo con `getInstance()`, posteriormente se usan sus métodos `setMaximumFractionDigits()` y `setMinimumFractionDigits()` para limitar los dígitos decimales: mínimo 2 y máximo 4 redondeando el número al valor más cercano.

```
public abstract class NumberFormat
extends Format
```

`NumberFormat` es una clase abstracta para formatear números permitiendo que el código sea independiente de las convenciones propias de cada país.

Métodos:

```
setMaximumFractionDigits(int newValue)
 Sets the maximum number of digits allowed in the fraction portion of a number.
```

```
setMinimumFractionDigits(int newValue)
 Sets the minimum number of digits allowed in the fraction portion of a number.
```

```
getInstance()
 Returns a general-purpose number format for the current default locale.
```

### QUESTION 163

Given:

11. `String test = "a1b2c3";`
12. `String[] tokens = test.split("\\d");`
13. `for(String s: tokens) System.out.print(s + " ");`

What is the result?

- A. a b c
- B. 1 2 3
- C. a1b2c3
- D. a1 b2 c3
- E. Compilation fails.
- F. The code runs with no output.
- G. An exception is thrown at runtime.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En este código se usa el método de String llamado `split()` el cual recibe un regex para hacer divisiones de una cadena. En este caso se va a dividir con un carácter `\d` o sea por Dígito quedando almacenado en el arreglo `tokens` de esta forma: `[a][b][c]`.

#### QUESTION 164

Given:

```
11. class Converter {
12. public static void main(String[] args) {
13. Integer i = args[0];
14. int j = 12;
15. System.out.println("It is " + (j==i) + " that j==i.");
16. }
17. }
```

What is the result when the programmer attempts to compile the code and run it with the command `java Converter 12`?

- A. It is true that `j==i`.
- B. It is false that `j==i`.
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error in line 13.

**Correct Answer:** D

**Section:** (none)

**Explanation**

#### Explanation/Reference:

La compilación falla en la línea 13 pues la variable `"i"` es de tipo `Integer` y recibe `args[0]` que es de tipo `String`.

#### QUESTION 165

Given:

```
1. public class BuildStuff {
2. public static void main(String[] args) {
3. Boolean test = new Boolean(true);
4. Integer x = 343;
5. Integer y = new BuildStuff().go(test, x);
6. System.out.println(y);
7. }
8. int go(Boolean b, int i) {
9. if(b) return (i/7);
10. return (i/49);
11. }
12. }
```

What is the result?

- A. 7
- B. 49
- C. 343
- D. Compilation fails.
- E. An exception is thrown at runtime.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En este caso la línea 5 se iguala la variable "y" al valor resultante de la invocación al método `BuilStuff().go (Boolean b, int i)`

El resultado se evalúa en la línea 9 dando 49 ya que la variable local `b` es "true" y nunca llega al return de la línea 10.

#### QUESTION 166

Given:

- ```
12. String csv = "Sue,5,true,3";  
13. Scanner scanner = new Scanner( csv );  
14. scanner.useDelimiter(",");  
15. int age = scanner.nextInt();
```

What is the result?

- A. Compilation fails.
- B. After line 15, the value of `age` is 5.
- C. After line 15, the value of `age` is 3.
- D. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La línea 15 arroja un error pues no es válido el regex que se introduce `,` para dividir la cadena, debería ser de la forma `"[,]"` si se desea delimitar por una coma, obviamente al no separar bien la cadena la variable `"int"` no recibe un entero como valor.

QUESTION 167

Given:

- ```
1. import java.util.*;
2. public class WrappedString {
3. private String s;
4. public WrappedString(String s) { this.s = s; }
5. public static void main(String[] args) {
6. HashSet<Object> hs = new HashSet<Object>();
7. WrappedString ws1 = new WrappedString("aardvark");
```

8.     WrappedString ws2 = new WrappedString("aardvark");
9.     String s1 = new String("aardvark");
10.    String s2 = new String("aardvark");
11.    hs.add(ws1); hs.add(ws2); hs.add(s1); hs.add(s2);
12.    System.out.println(hs.size()); } }

What is the result?

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4
- F. Compilation fails.
- G. An exception is thrown at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Una característica de los wrappers es que si son iguales dos variables sus hash codes son iguales, en el caso de comparar los hash codes de otras clases arrojan valores diferentes. por ello los hash codes hacen que la variable s1 y s2 sean iguales pero ws1 y ws2 diferentes, como los hashCode permiten almacenar valores no repetidos solo almacenar tres variables.

**hashCode()**

Returns a hash code value for the object.

#### QUESTION 168

Given a class whose instances, when found in a collection of objects, are sorted by using the compareTo() method, which two statements are true? (Choose two.)

- A. The class implements java.lang.Comparable.
- B. The class implements java.util.Comparator.
- C. The interface used to implement sorting allows this class to define only one sort sequence.
- D. The interface used to implement sorting allows this class to define many different sort sequences.

**Correct Answer:** AC

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Para usar el método compareTo() en colecciones es necesario implementar la interfaz "Comparable", este método permite ordenar las listas de acuerdo a un solo criterio.

**B** es falso la interfaz comparator tiene el metodo compare(Object o1, Object o2) que compara dos objetos para ordenar.

**int compareTo(T o)**

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

#### QUESTION 169

Given:

1. import java.util.\*;

```

2. public class Example {
3. public static void main(String[] args) {
4. // insert code here
5. set.add(new Integer(2));
6. set.add(new Integer(1));
7. System.out.println(set);
8. }
9. }

```

Which code, inserted at line 4, guarantees that this program will output [1, 2]?

- A. Set set = new TreeSet();
- B. Set set = new HashSet();
- C. Set set = new SortedSet();
- D. List set = new SortedList();
- E. Set set = new LinkedHashSet();

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La coleccion Tree set ordena naturalmente sus componentes garantizando que la salida sea [1,2].

```

public class TreeSet<E>
extends AbstractSet<E>
implements NavigableSet<E>, Cloneable, Serializable

```

A NavigableSet implementation based on a TreeMap. The elements are ordered using their natural ordering, or by a Comparator provided at set creation time, depending on which constructor is used.

## QUESTION 170

Given:

```

11. public class Person {
12. private name;
13. public Person(String name) {
14. this.name = name;
15. }
16. public int hashCode() {
17. return 420;
18. }
19. }

```

Which statement is true?

- A. The time to find the value from HashMap with a Person key depends on the size of the map.
- B. Deleting a Person key from a HashMap will delete all map entries for all keys of type Person.
- C. Inserting a second Person object into a HashSet will cause the first Person object to be removed as a duplicate.
- D. The time to determine whether a Person object is contained in a HashSet is constant and does NOT depend on the size of the map.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**B** es incorrecto, cada elemento de un HashMapo es independiente y tiene una clave de acceso unica.

**C** es incorrecto, los HashSet no permiten duplicados ademas si se insertara un objeto Person nuevo dentro de un HashSet tendria un hascode diferente y por ende se insertaría en un nuevo campo.

**D** es incorrecto, acceder a los datos de una colección obviamente depende siempre del tamaño de la misma.

### QUESTION 171

DRAG DROP

Click the Task button.

Drag and Drop

Given:

```

class A {
 String name = "A";
 String getName() {
 return name;
 }
 String greeting(){
 return "class A";
 }
}
class B extends A {
 String name = "B";
 String greeting() {
 return "class B";
 }
}
public class Client {
 public static void main(String[] args) {
 A a = new A();
 A b = new B();
 System.out.println(a.greeting() + " has name " + a.getName());
 System.out.println(b.greeting() + " has name " + b.getName());
 }
}

```

Place the names "A" and "B" in the following output.

class Place here has name Place here

class Place here has name Place here

**Names**

A
B

Done

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

## Explanation

### Explanation/Reference:

Drag and Drop

Given:

```
class A {
 String name = "A";
 String getName() {
 return name;
 }
 String greeting(){
 return "class A";
 }
}
class B extends A {
 String name = "B";
 String greeting() {
 return "class B";
 }
}
public class Client {
 public static void main(String[] args) {
 A a = new A();
 B b = new B();
 System.out.println(a.greeting() + " has name " + a.getName());
 System.out.println(b.greeting() + " has name " + b.getName());
 }
}
```

Place the names "A" and "B" in the following output.

class  has name

class  has name

**Names**

## QUESTION 172

### DRAG DROP

Click the Task button.

Place the code elements into the class so that the code compiles and prints "Run. Run. doIt." in exactly that order. Note that there may be more than one correct solution.

```
public class TestTwo extends Thread {
 public static void main (String[] a) throws Exception {
 TestTwo t = new TestTwo();
 t.start();

 }
 public void run() {
 System.out.print("Run. ");
 }
 public void doIt() {
 System.out.print("doIt. ");
 }
}
```

### Code Elements

A.



- B.
- C.
- D.

**Correct Answer: A**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

Place the code elements into the class so that the code compiles and prints "Run. Run. doIt." in exactly that order. Note that there may be more than one correct solution.

```
public class TestTwo extends Thread {
 public static void main (String[] a) throws Exception {
 TestTwo t = new TestTwo();
 t.start();
 t.join();
 t.run();
 t.doIt();
 }
 public void run() {
 System.out.print("Run. ");
 }
 public void doIt() {
 System.out.print("doIt. ");
 }
}
```

#### Code Elements

|            |           |              |        |      |
|------------|-----------|--------------|--------|------|
| t.start(); | t.join(); | t.pause(10); | run(); | Done |
| t.run();   | t.doIt(); | doIt();      |        |      |

#### QUESTION 173

DRAG DROP

Click the Task button.

Drag and Drop

Place the code fragments into position to use a BufferedReader to read in an entire text file.

```

class PrintFile {
 public static void main(String[] args){
 BufferedReader buffReader = null;
 //more code here to initialize buffReader
 try {
 String temp;
 while(
 Place here
 Place here
) {
 System.out.println(temp);
 }
 } catch
 Place here
 {
 e.printStackTrace();
 }
 }
}

```

**Code Fragments**

|                                 |                             |      |
|---------------------------------|-----------------------------|------|
| (temp = buffReader.readLine()); | && buffReader.hasNext();    | Done |
| (temp = buffReader.nextLine()); | [IOException e] {           |      |
| temp = null;                    | [FileNotFoundException e] { |      |



- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**Drag and Drop**

Place the code fragments into position to use a BufferedReader to read in an entire text file.

```

class PrintFile {
 public static void main(String[] args){
 BufferedReader buffReader = null;
 //more code here to initialize buffReader
 try {
 String temp;
 while([] []) {
 System.out.println(temp);
 }
 } catch [] {
 e.printStackTrace();
 }
 }
}

```

**Code Fragments**

|                                  |                               |
|----------------------------------|-------------------------------|
| [ temp = buffReader.readLine() ] | [ &&buffReader.hasNext() ]    |
| [ temp = buffReader.nextLine() ] | [ IOException e ] {           |
| [ temp = null ]                  | [ FileNotFoundException e ] { |

#### QUESTION 174

DRAG DROP

Click the Task button.

Drag and Drop

Place the correct description of the compiler output on the code fragments to be inserted at lines 4 and 5. The same compiler output may be used more than once.

```
1 import java.util.*;
2 public class X {
3 public static void main(String[] args) {
4 // insert code here
5 // insert code here
6 }
7 public static void foo(List<Object> list) {
8 }
9 }
```

**Code**

- ArrayList<String> x1 = new ArrayList<String>();  
foo(x1);
- ArrayList<Object> x2 = new ArrayList<String>();  
foo(x2);
- ArrayList<Object> x3 = new ArrayList<Object>();  
foo(x3);
- ArrayList x4 = new ArrayList();  
foo(x4);

**Compiler Output**

- Compilation succeeds.
- Compilation fails due to an error in the first statement.
- Compilation of the first statement succeeds, but compilation fails due to an error in the second statement.

Done

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**



**Drag and Drop**

Place the correct description of the compiler output on the code fragments to be inserted at lines 4 and 5. The same compiler output may be used more than once.

```

1 import java.util.*;
2 public class X {
3 public static void main(String[] args) {
4 // insert code here
5 // insert code here
6 }
7 public static void foo(List<Object> list) {
8 }

```

**Code**

- Compilation of the first statement succeeds, but compilation fails due to an error in the second statement.
- Compilation fails due to an error in the first statement.
- Compilation succeeds.
- Compilation succeeds.

**Compiler Output**

- Compilation succeeds.
- Compilation fails due to an error in the first statement.
- Compilation of the first statement succeeds, but compilation fails due to an error in the second statement.

Done

#### QUESTION 175

##### DRAG DROP

Click the Task button.

**Drag and Drop**

Given:

```

1 import java.util.*;
2 class A { }
3 class B extends A { }
4 public class Test {
5 public static void main(String[] args) {
6 List<A> listA = new LinkedList<A>();
7 List listB = new LinkedList();
8 List<Object> listO = new LinkedList<Object>();
9 // insert code here
10 }
11 public static void m1(List<? extends A> list) { }
12 public static void m2(List<A> list) { }
13 }

```

Place a result onto each method call to indicate what would happen if the method call were inserted at line 9. Note: Results can be used more than once.

| Method Calls |            | Result                            |
|--------------|------------|-----------------------------------|
| m1(listA);   | m2(listA); | Does not compile                  |
| m1(listB);   | m2(listB); | Compiles and runs without error   |
| m1(listO);   | m2(listO); | An exception is thrown at runtime |

Done

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**Drag and Drop**

Given:

```

1. import java.util.*;
2. class A { }
3. class B extends A { }
4. public class Test {
5. public static void main(String[] args) {
6. List<A> listA = new LinkedList<A>();
7. List listB = new LinkedList();
8. List<Object> listO = new LinkedList<Object>();
9. // insert code here
10. }
11. public static void m1(List<? extends A> list) { }
12. public static void m2(List<A> list) { }
13. }

```

Place a result onto each method call to indicate what would happen if the method call were inserted at line 9. Note: Results can be used more than once.

| Method Calls                     |                                  | Result                             |
|----------------------------------|----------------------------------|------------------------------------|
| Compiles and runs without error. | Compiles and runs without error. | Does not compile.                  |
| Compiles and runs without error. | Does not compile.                | Compiles and runs without error.   |
| Does not compile.                | Does not compile.                | An exception is thrown at runtime. |

Done

#### QUESTION 176

DRAG DROP

Click the Task button.



Drag and Drop

Place the code into the GenericB class definition to make the class compile successfully.

```
import java.util.*;

public class GenericB<Place> {

 public Place foo;

 public void setFoo(Place foo) {
 this.foo = foo;
 }

 public Place getFoo() {
 return foo;
 }

 public static void main (String[] args) {
 GenericB<Cat> bar = new GenericB<Cat>();
 bar.setFoo(new Cat());
 Cat c = bar.getFoo();
 }
}

interface Pet { }
class Cat implements Pet{ }
```

**Code**

- ? extends Pet
- T extends Pet
- ? implements Pet
- T implements Pet
- Pet extends T
- ?
- T
- <?>
- Pet

Done

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Drag and Drop

Place the code into the GenericB class definition to make the class compile successfully.

```
import java.util.*;

public class GenericB<T extends Pet> {
 public T foo;
 public void setFoo(T foo) {
 this.foo = foo;
 }
 public T getFoo() {
 return foo;
 }
 public static void main (String[] args) {
 GenericB<Cat> bar = new GenericB<Cat>();
 bar.setFoo(new Cat());
 Cat c = bar.getFoo();
 }
}

interface Pet { }
class Cat implements Pet{ }
```

**Code**

- ? extends Pet
- T extends Pet
- ? implements Pet
- T implements Pet
- Pet extends T
- ?
- T
- <?>
- Pet

Done

#### QUESTION 177

Given:

1. class TestException extends Exception { }
2. class A {
3.     public String sayHello(String name) throws TestException {
4.         if(name == null) throw new TestException();
5.         return "Hello " + name;
6.     }
7. }
8. public class TestA {
9.     public static void main(String[] args) {
10.         new A().sayHello("Aiko");
11.     }
12. }

Which statement is true?

- A. Compilation succeeds.
- B. Class A does not compile.
- C. The method declared on line 9 cannot be modified to throw TestException.
- D. TestA compiles if line 10 is enclosed in a try/catch block that catches TestException.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El código anterior en el método sayHello(String name) lanza la excepción TestException al método llamante (main) el cual debe tratarla con un bloque try -catch que en el código no se encuentra.

#### QUESTION 178

Given:

```
11. public static void main(String[] args) {
12. for (int i = 0; i <= 10; i++) {
13. if (i > 6) break;
14. }
15. System.out.println(i);
16. }
```

What is the result?

- A. 6
- B. 7
- C. 10
- D. 11
- E. Compilation fails.
- F. An exception is thrown at runtime.

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La compilación falla en la línea 15 porque se trata de imprimir la variable "i" la cual es solo válida para el bloque "for".

#### QUESTION 179

Given:

```
3. public class Breaker {
4. static String o = "";
5. public static void main(String[] args) {
6. z:
7. o = o + 2;
8. for(int x = 3; x < 8; x++) {
9. if(x==4) break;
10. if(x==6) break z;
11. o = o + x;
12. }
```

```
13. System.out.println(o);
14. }
15. }
```

What is the result?

- A. 23
- B. 234
- C. 235
- D. 2345
- E. 2357
- F. 23457
- G. Compilation fails.

**Correct Answer:** G

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La compilación falla en la línea 10 porque no se encuentra la label "z" esta debe ser declarada justo antes de los bucles osea en este caso en la línea 8.

#### QUESTION 180

Given:

```
5. class A {
6. void foo() throws Exception { throw new Exception(); }
7. }
8. class SubB2 extends A {
9. void foo() { System.out.println("B "); }
10. }
11. class Tester {
12. public static void main(String[] args) {
13. A a = new SubB2();
14. a.foo();
15. }
16. }
```

What is the result?

- A. B
- B. B, followed by an Exception.
- C. Compilation fails due to an error on line 9.
- D. Compilation fails due to an error on line 14.
- E. An Exception is thrown with no other output.

**Correct Answer:** D

**Section:** (none)

**Explanation**



**Explanation/Reference:**

Como la variable a se declara de tipo "A" y esta clase en su método foo lanza una Exception al método llamante entonces en el main se debe incluir un bloque try-catch para tratar el error.

**QUESTION 181**

Given:

```
11. public static void main(String[] args) {
12. String str = "null";
13. if (str == null) {
14. System.out.println("null");
15. } else (str.length() == 0) {
16. System.out.println("zero");
17. } else {
18. System.out.println("some");
19. }
20. }
```

What is the result?

- A. null
- B. zero
- C. some
- D. Compilation fails.
- E. An exception is thrown at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

hay dos errores en el código uno es en la línea 15 pues no es válida la declaración de else con corchetes. y la línea 17 también genera un error pues hay un else sin el if correspondiente.

**QUESTION 182**

Given:

```
1. public class Mule {

2. public static void main(String[] args) {

3. boolean assert = true;

4. if(assert) {

5. System.out.println("assert is true");

6. }

7. }

8. }
```

Which command-line invocations will compile?

- A. javac Mule.java
- B. javac -source 1.3 Mule.java
- C. javac -source 1.4 Mule.java
- D. javac -source 1.5 Mule.java

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Ya que el parámetro -source se usa para hacer compatible el código con una versión de Java, dado que los `assert` se crearon en Java 1.4 se debe compilar con una versión menor para que no genere error la palabra reservada `assert` de la línea 3.

El error es el siguiente: as of release 1.4, 'assert' is a keyword, and may not be used as an identifier (use -source 1.3 or lower to use 'assert' as an identifier)

En la ayuda de `javac` nos dice:

`-source<release>` Provide source compatibility with specified release

**QUESTION 183**

Given:

```
11. static void test() {
12. try {
13. String x = null;
14. System.out.print(x.toString() + " ");
15. }
16. finally { System.out.print("finally "); }
17. }

18. public static void main(String[] args) {
19. try { test(); }
20. catch (Exception ex) { System.out.print("exception "); }
21. }
```

What is the result?

- A. null
- B. finally
- C. null finally
- D. Compilation fails.
- E. finally exception

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Los `wrap` en su método `toString()` lanzan una `NullPointerException` si el valor de la variable es `null`, por ello la única sentencia que se ejecuta es el `finally` del método `test`. La "exception" como no la captura el método `test()` con algún "catch" la trata de nuevo el método `main` por ello se imprime "finally exception".

**QUESTION 184**

Given:

```
1. public class Boxer1{
2. Integer i;
3. int x;
4. public Boxer1(int y) {
5. x = i+y;
6. System.out.println(x);
7. }
8. public static void main(String[] args) {
9. new Boxer1(new Integer(4));
10. }
11. }
```

What is the result?

- A. The value "4" is printed at the command line.
- B. Compilation fails because of an error in line 5.
- C. Compilation fails because of an error in line 9.
- D. A NullPointerException occurs at runtime.
- E. A NumberFormatException occurs at runtime.
- F. An IllegalStateException occurs at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el anterior código ocurre una NullPointerException en la línea 19 ya que la variable "i" al ser de tipo Integer es inicializada con un valor "Null" y al intentarla sumar con la variable "y" nos genera este error.

**QUESTION 185**

Which two code fragments are most likely to cause a StackOverflowError? (Choose two.)

- A. 

```
int []x = {1,2,3,4,5};
for(int y = 0; y < 6; y++)
 System.out.println(x[y]);
```
- B. 

```
static int[] x = {7,6,5,4};
static { x[1] = 8;
 x[4] = 3; }
```
- C. 

```
for(int y = 10; y < 10; y++)
 doStuff(y);
```
- D. 

```
void doOne(int x) { }
void doTwo(int y) { doThree(y); }
void doThree(int z) { doTwo(z); }
```
- E. 

```
for(int x = 0; x < 1000000000; x++)
 doStuff(x);
```
- F. 

```
void counter(int i) { counter(++i); }
```

**Correct Answer:** DF

**Section: (none)****Explanation****Explanation/Reference:**

Este error se produce cuando hay llamadas recursivas que nunca acabarán parecidas a un bucle infinito, son errores por funciones que se llaman a sí mismas infinitamente y desbordan el stack, que es la pila de almacenamiento de parámetros y variables locales. Las opciones **D** y **F** crean este `StackOverflowError` por lo tanto son las respuestas correctas.

Parameters and local variables are allocated on the stack (with reference types the object lives on the heap and a variable references that object). The stack typically lives at the upper end of your address space and as it is used up it heads towards the bottom of the address space (ie towards zero).

Your process also has a heap, which lives at the bottom end of your process. As you allocate memory this heap can grow towards the upper end of your address space. As you can see, there is the potential for the heap to "collide" with the stack (a bit like tectonic plates!!!).

The common cause for a stack overflow is a bad recursive call. Typically this is caused when your recursive functions doesn't have the correct termination condition, so it ends up calling itself for ever. However, with GUI programming it's possible to generate indirect recursion. For example, your app may be handling paint messages and whilst processing them it may call a function that causes the system to send another paint message. Here you've not explicitly called yourself, but the OS/VM has done it for you.

To deal with them you'll need to examine your code. If you've got functions that call themselves then check that you've got a terminating condition. If you have then check that when calling the function you have at least modified one of the arguments, otherwise there'll be no visible change for the recursively called function and the terminating condition is useless.

If you've got no obvious recursive functions then check to see if you're calling any library functions that indirectly will cause your function to be called (like the implicit case above).

**QUESTION 186**

Given:

```
11. static void test() throws RuntimeException {
12. try {
13. System.out.print("test ");
14. throw new RuntimeException();
15. }
16. catch (Exception ex) { System.out.print("exception "); }
17. }
18. public static void main(String[] args) {
19. try { test(); }
20. catch (RuntimeException ex) { System.out.print("runtime "); }
21. System.out.print("end ");
22. }
```

What is the result?

- A. test end
- B. Compilation fails.
- C. test runtime end

- D. test exception end
- E. A Throwable is thrown by main at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El código anterior llama primero el método test() que imprime "test" luego lanza una RuntimeException que es tratada de una vez en su catch ya que el maneja todas las Exception imprimiendo "exception", despues finaliza el método main imprimiendo "end".Resultado:test exception end

#### QUESTION 187

Given:

```
11. public static void main(String[] args) {
12. Integer i = new Integer(1) + new Integer(2);
13. switch(i) {
14. case 3: System.out.println("three"); break;
15. default: System.out.println("other"); break;
16. }
17. }
```

What is the result?

- A. three
- B. other
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error on line 12.
- E. Compilation fails because of an error on line 13.
- F. Compilation fails because of an error on line 15.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La variable "i" antes de entrar en el switch tiene un valor de 3 por ello se ejecuta la condición de la linea 14 y luego con el break se sale del bloque switch.

#### QUESTION 188

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.

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La compilación falla porque la variable privada country declarada en la línea 22 solo se puede acceder a través de su método getC.

#### QUESTION 189

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;

**Correct Answer:** AEF

**Section: (none)**

**Explanation**

**Explanation/Reference:**

**B** es incorrecto cada clase extiende por aparte a la clase padre "A" y por ello no son iguales.

**C** es incorrecto tienen tipos incompatibles uno es "ClassC" y el otro "ClassA"

**D** es incorrecto el casting no se puede realizar a la clase ClassA pues esta no conoce a la clase "ClassC".

#### QUESTION 190

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 overridden 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 overridden in any subclass of X.
- E. A public static method in class X can be called by a subclass of X without explicitly 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 overridden by a subclass of X only if the subclass is in the same package as X.

**Correct Answer:** BEF

**Section: (none)**

**Explanation**

**Explanation/Reference:**

**E** es correcto porque:

The "final" modifier for a method, means that the method can't be overridden. However, overriding requires that the method be accessible. If the subclass can't access the method, then it is not overriding -- it is just another method of the subclass.

**A** es incorrecto porque abstracto y final son opuestos un método no puede ser ambas cosas.

**C** es incorrecto, los métodos estáticos pueden ser llamados desde "no static" métodos.

**D** es incorrecto ningún método público y final puede ser sobrescrito.

#### QUESTION 191

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.

**Correct Answer:** F

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El error se genera en la línea 25 porque la variable "a" es de tipo "A" y al invocar el método "y()" este no se conoce por la interfaz "A".

#### QUESTION 192

Given:

1. package test;
- 2.
3. class Target {
4.     public String name = "hello";
5. }

What can directly access and change the value of the variable name?

- A. any class
- B. only the Target class
- C. any class in the test package
- D. any class that extends Target

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Si el modificador es público se pueden acceder a las variables desde cualquier clase, pero directamente con el nombre de la clase solo dentro del paquete; de otra manera hay que anteponer el nombre del paquete.

#### QUESTION 193

Click the Exhibit button. What two must the programmer do to correct the compilation errors? (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() in the MeGo constructor
- C. insert a call to 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;

**Correct Answer:** DE

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El constructor de la clase MeGo invoca el superconstructor super(); el cual no encuentra por ello es necesario hacer un llamado al super constructor Car(String vin). La variable wheelCount es privada y no se puede acceder desde MeGo por ello es necesario cambiarle el modificador a protected o public.

#### QUESTION 194

A team of programmers is involved in reviewing a proposed design for a new utility class. After some discussion, they realize that the current design allows other classes to access methods in the utility class that should be accessible only to methods within the utility class itself. What design issue has the team discovered?

- A. Tight coupling
- B. Low cohesion
- C. High cohesion
- D. Loose coupling
- E. Weak encapsulation
- F. Strong encapsulation

**Correct Answer:** E

**Section:** (none)

## Explanation

### Explanation/Reference:

Cuando la encapsulacion es debil las variables son públicas, el problema anterior que fue descubierto fue este tipo de encapsulamiento debil.

### QUESTION 195

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 DeluxeThingy 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.

**Correct Answer:** AF

**Section:** (none)

### Explanation

### Explanation/Reference:

Ambas llamadas al método go() (línea 12 y 14), pertenecen a la clase Meter la cual imprime "m" como salida. Si Thingy tiene un "Meter" cualquier subclase como DeluxeThingy tienen un "Meter" también.

### QUESTION 196

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

**Correct Answer:** BCF

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**A** es falso Cat implementa un Jumper osea que es un Jumper.

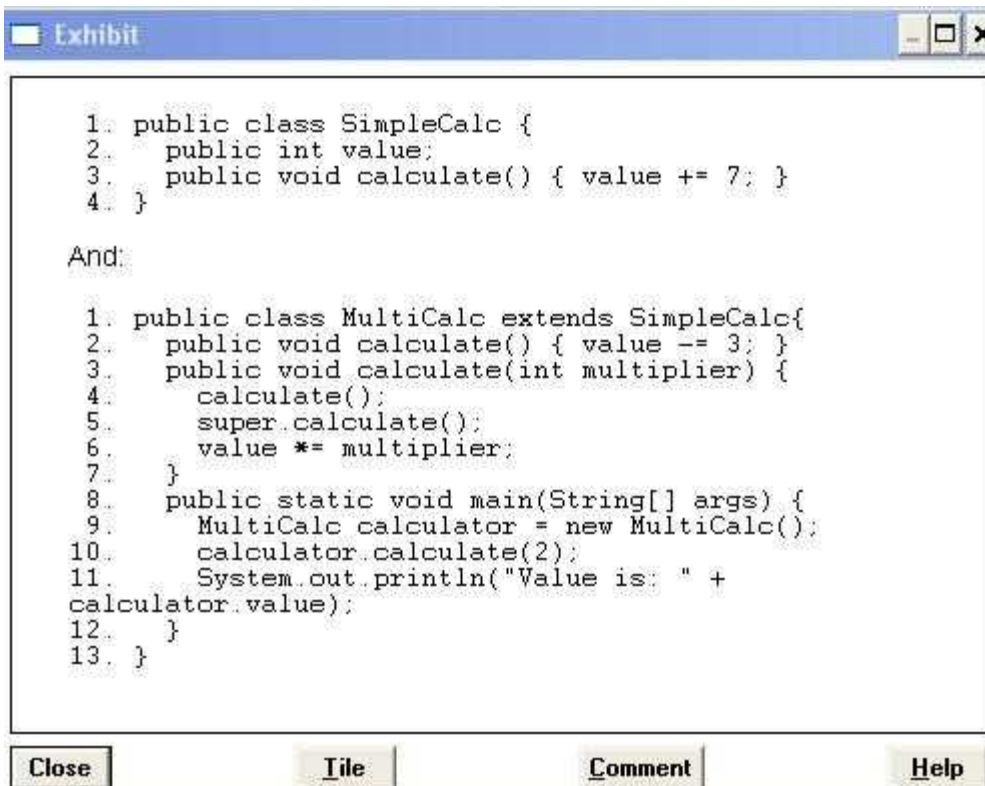
**D** Dog extiende animal osea que es un "Animal"

**E** es incorrecto cat no tiene un animal solo es un Jumper.

**G** es incorrecto Beagle tiene un Tail porque la superclase Dog tiene un "Tail".

#### QUESTION 197

Click the Exhibit button. What is the result?



```
1. public class SimpleCalc {
2. public int value;
3. public void calculate() { value += 7; }
4. }

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. }
```

Close    File    Comment    Help

- 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.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La sintaxis es la correcta el valor final es 8.

#### QUESTION 198

Given a valid DateFormat object named df, and

- 16. Date d = new Date(0L);
- 17. String ds = "December 15, 2004";
- 18. // insert code here

What updates d's value with the date represented by ds?

- A. 18. d = df.parse(ds);
- B. 18. d = df.getDate(ds);
- C. 18. try {  
19. d = df.parse(ds);  
20. } catch(ParseException e) { };
- D. 18. try {  
19. d = df.getDate(ds);  
20. } catch(ParseException e) { };

**Correct Answer:** C

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El código válido es el **C** porque el método de DateFormat parse() convierte un string en una fecha válida pero necesita capturar la ParseException que podría surgir con una sentencia Try-catch.

#### QUESTION 199

Which two scenarios are NOT safe to replace a StringBuffer object with a StringBuilder object? (Choose two.)

- A. When using versions of Java technology earlier than 5.0.
- B. When sharing a StringBuffer among multiple threads.
- C. When using the java.io class StringBufferInputStream.
- D. When you plan to reuse the StringBuffer to build more than one string.

**Correct Answer:** AB

**Section:** (none)

**Explanation**

**Explanation/Reference:**

StringBuilder se incorporó en la version 5 por ello no es bueno el remplazo en versiones mas antiguas. No se debe remplazar StringBuffer por StringBuilder cuando se trabajan hilos pues StringBuilder no soporta ser sincronizado.

### QUESTION 200

Given:

11. String test = "a1b2c3";

12. String[] tokens = test.split("\\d");

13. for(String s: tokens) System.out.print(s + " ");

What is the result?

- A. a b c
- B. 1 2 3
- C. a1b2c3
- D. a1 b2 c3
- E. Compilation fails.
- F. The code runs with no output.
- G. An exception is thrown at runtime.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En este código se usa el método de String llamado split() el cual recibe un regex para hacer divisiones de una cadena. En este caso se va a dividir con un carácter \d o sea por Dígito quedando almacenado en el arreglo tokens de esta forma: [a][b][c].

### QUESTION 201

Given:

```
1. public class TestString3 {
2. public static void main(String[] args) {
3. // insert code here
5. System.out.println(s);
6. }
7. }
```

Which two code fragments, inserted independently at line 3, generate the output 4247? (Choose two.)

- A. String s = "123456789";  
s = (s-"123").replace(1,3,"24") - "89";
- B. StringBuffer s = new StringBuffer("123456789");  
s.delete(0,3).replace(1,3,"24").delete(4,6);
- C. StringBuffer s = new StringBuffer("123456789");  
s.substring(3,6).delete(1,3).insert(1, "24");
- D. StringBuilder s = new StringBuilder("123456789");  
s.substring(3,6).delete(1,2).insert(1, "24");
- E. StringBuilder s = new StringBuilder("123456789");  
s.delete(0,3).delete(1,3).delete(2,5).insert(1, "24");

**Correct Answer:** BE

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**C y D** son incorrectas porque el método substring devuelve un tipo String y por ende no existe submétodo delete en este.

**A** es incorrecto, entre otras cosas porque el operador "-" no aplica para cadenas String.

#### QUESTION 202

Given:

11. String test = "Test A. Test B. Test C.";

12. // insert code here

13. String[] result = test.split(regex);

Which regular expression, inserted at line 12, correctly splits test into "Test A", "Test B", and "Test C"?

- A. String regex = "";
- B. String regex = " ";
- C. String regex = ". \*";
- D. String regex = "\\s";
- E. String regex = "\\s.\*";
- F. String regex = "\\w[\\.]+";

**Correct Answer:** E

**Section:** (none)

**Explanation**

#### Explanation/Reference:

La única opción válida es la **E** porque se requiere una regex que divida el String en un punto seguido de un espacio y eso significa \\s. y \\s.

#### QUESTION 203

Which statement is true?

- A. A class's finalize() method CANNOT be invoked explicitly.
- B. super.finalize() is called implicitly by any overriding finalize() method.
- C. The finalize() method for a given object is called no more than once by the garbage collector.
- D. The order in which finalize() is called on two objects is based on the order in which the two objects became finalizable.

**Correct Answer:** C

**Section:** (none)

**Explanation**

#### Explanation/Reference:

El método Finalize es invocado por el recolector de basura cuando se ha determinado que un objeto ya no se usa, y puede ser recolectado. Dado que un objeto puede ser recolectado sólo una vez, el método de finalizar también se invoca sólo una vez.

#### QUESTION 204

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.println(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.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El check-error en compilacion ocurre en la linea 14 ya que no se le puede asignar un nuevo valor a una variable "final".

## QUESTION 205

Given:

```

11. interface DeclareStuff {
12. public static final int EASY = 3;
13. void doStuff(int t); }
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 + ++s;
21. System.out.println("s " + s);
22. }
23. }

```

What is the result?

- A. s 14
- B. s 16
- C. s 10

- D. Compilation fails.
- E. An exception is thrown at runtime.

**Correct Answer:** D

**Section:** (none)

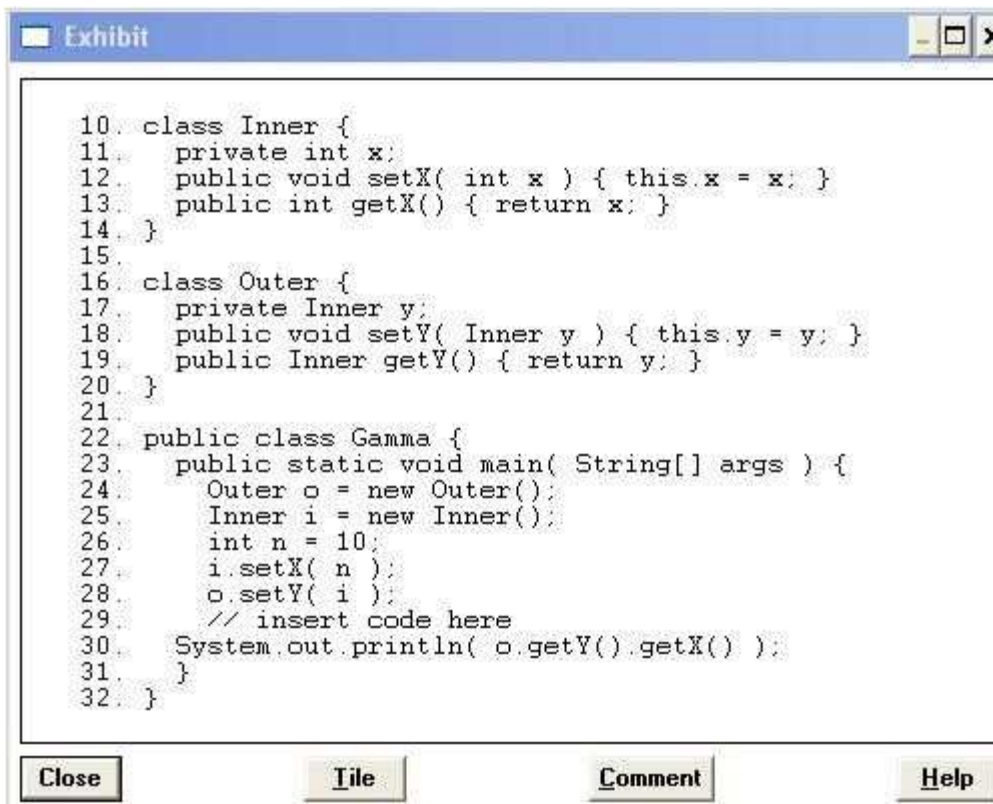
**Explanation**

**Explanation/Reference:**

La compilación falla en la línea 19 pues para implementar un método de una interfaz es necesario que el método que sobrescriba tenga un modificador de acceso public ya que los métodos de las interfaces son public. la interfaz debería quedar: public void doStuff(int s).

#### QUESTION 206

Click the Exhibit button. 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 = new 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 );

**Correct Answer:** BCF

**Section:** (none)

**Explanation**

**Explanation/Reference:**

De las formas **B, C y F** se hace referencia a la variable "x" de la clase Inner y se le da un valor de 100.

**A** es incorrecto solo le da el valor de 100 a la variable local "n".

**D y E** son incorrectas porque se crea una nueva instancia de la clase Inner la cual el objeto o de la clase "Outer" no conoce.



### QUESTION 207

Given:

```
11. public class Commander {
12. public static void main(String[] args) {
13. String myProp = /* insert code here */
14. System.out.println(myProp);
15. }
16. }
```

and the command line:

java -Dprop.custom=gobstopper Commander Which two, placed on line 13, will produce the output gobstopper? (Choose two.)

- A. System.load("prop.custom");
- B. System.getenv("prop.custom");
- C. System.property("prop.custom");
- D. System.getProperty("prop.custom");
- E. System.getProperties().getProperty("prop.custom");

**Correct Answer:** DE

**Section:** (none)

**Explanation**

#### **Explanation/Reference:**

De las dos formas **D** y **E** esta bien declarada la propiedad en **System.getProperty("prop.custom")** se llama el método `getProperty()` de la clase `System`, y en la otra primero se llama a **System.getProperties()** que devuelve un objeto tipo "Properties" que pertenece a `java.util` y almacena las propiedades completas, dentro de el tiene un metodo llamado `getProperty()` el cual funciona para devolver el valor de la propiedad nombrada.

#### **Reading System Properties**

The `System` class has two methods used to read system properties: `getProperty` and `getProperties`.

La clase `System` tiene dos versiones diferentes de `getProperty()`. Ambas versiones devuelven el valor de la propiedad nombrada en la lista de argumentos. La más simple de las dos `getProperty()` toma un sólo argumento: la clave de la propiedad que quiere buscar. Por ejemplo, para obtener el valor de `path.separator`, utilizamos la siguiente sentencia:

```
System.getProperty("path.separator");
```

Este método devuelve una cadena que contiene el valor de la propiedad. Si la propiedad no existe, esta versión de `getProperty()` devuelve `null`.

El último método proporcionado por la clase `System` para acceder a los valores de las propiedades es el método **`getProperties()`** que devuelve un objeto que contiene el conjunto completo de las propiedades del sistema. Se pueden utilizar varios métodos de la clase `Properties` para consultar valores específicos o para listar el conjunto completo de propiedades. Para más información sobre la clase `Properties`, puedes ver [Seleccionar y utilizar Propiedades](#).

### QUESTION 208

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.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La sintaxis esta bien y el programa compila sin problema.

#### QUESTION 209

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.print("f-p ");
13. if(w instanceof Fish) System.out.print("w-f ");

```

14. if(b instanceof Fish) System.out.print("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.

**Correct Answer:** B

**Section:** (none)

**Explanation**

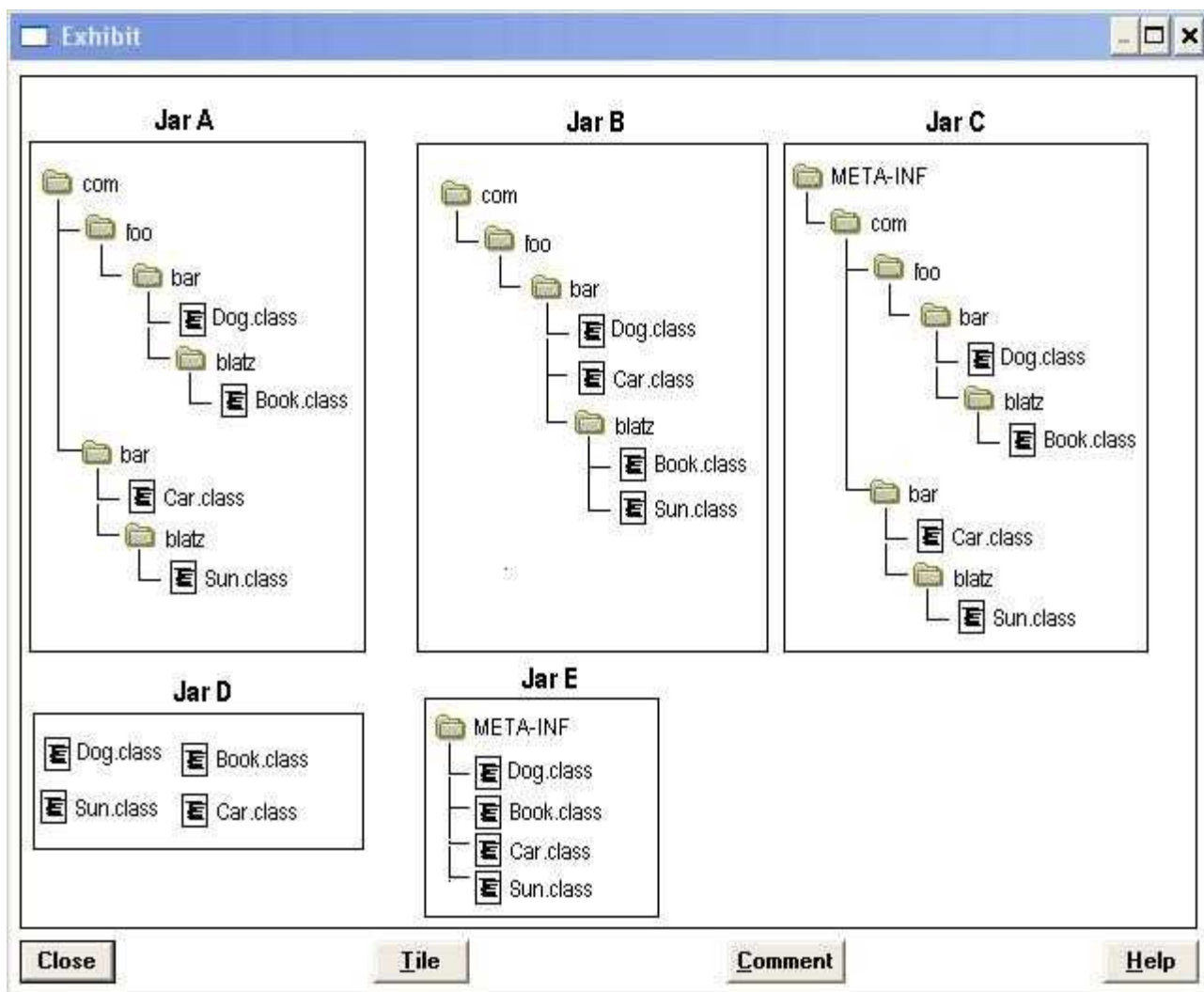
**Explanation/Reference:**

Para que una subclase sea instancia de "instanceof" otra superclase o de interface debe o bien extenderla o implementar la interfaz.

El objetivo del operador instanceof es conocer si un objeto es de un tipo determinado. Por tipo nos referimos a clase o interfaz (interface), es decir si el objeto pasaría el test ES-UN para esa clase o ese interfaz, especificado a la derecha del operador.

#### **QUESTION 210**

Click the Exhibit button. Given the fully-qualified class names: com.foo.bar.Dog com.foo.bar.blatz.Book com.bar.Car com.bar.blatz.Sun Which graph represents the correct directory structure for a JAR file from which those classes can be used by the compiler and JVM?



- A. Jar A
- B. Jar B
- C. Jar C
- D. Jar D
- E. Jar E

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Nos dan los nombres de clase completos para las clases de este modo:

`com.foo.bar.Dog`  
`com.foo.bar.blatz.Book`  
`com.bar.Car`  
`com.bar.blatz.Sun`

Siendo de este modo observamos que todas las clases estan alojadas en el paquete "com", la clase "sun" esta dentro de bar/blatz/ y la clase "book" esta alojada en el paquete foo/bar/blatz y dentro del paquete "bar" estan la clase "Dog", el unico diagrama que corresponde es el Jar A. Respuesta correcta "A".

#### QUESTION 211

Given:

1. `package com.company.application;`

2.

3. public class MainClass {

4.     public static void main(String[] args) {}

5. }

And MainClass exists in the /apps/com/company/application directory. Assume the CLASSPATH environment variable is set to "." (current directory). Which two java commands entered at the command line will run MainClass? (Choose two.)

A. java MainClass if run from the /apps directory

B. java com.company.application.MainClass if run from the /apps directory

C. java -classpath /apps com.company.application.MainClass if run from any directory

D. java -classpath . MainClass if run from the /apps/com/company/application directory

E. java -classpath /apps/com/company/application:. MainClass if run from the /apps directory

F. java com.company.application.MainClass if run from the /apps/com/company/application directory

**Correct Answer:** BC

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**A** es incorrecto. el compilador asume que la clase a ejecutar esta alojada en el directorio /apps y no es asi.

**D** es incorrecto no se puede reconocer la ruta con el identificador ".".

**E** es incorrecto el simbolo ":" no se reconoce en esta declaración.

**F** es incorrecto el compilador intentara encontrar la ruta de la clase en: /apps/com/company/application/com/company/application/MainClass

### Classpath (Java)

En el lenguaje de programación Java se entiende por Classpath una opción admitida en la línea de órdenes o mediante variable de entorno que indica a la Máquina Virtual de Java dónde buscar paquetes y clases definidas por el usuario a la hora de ejecutar programas.

### QUESTION 212

Given:

12. import java.util.\*;

13. public class Explorer2 {

14.     public static void main(String[] args) {

15.         TreeSet<Integer> s = new TreeSet<Integer>();

16.         TreeSet<Integer> subs = new TreeSet<Integer>();

17.         for(int i = 606; i < 613; i++)

18.             if(i%2 == 0) s.add(i);

19.         subs = (TreeSet)s.subSet(608, true, 611, true);

20.         s.add(629);

21.         System.out.println(s + " " + subs);

22.     }

23. }

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. [608, 610, 612, 629] [608, 610]
- D. [608, 610, 612, 629] [608, 610, 629]
- E. [606, 608, 610, 612, 629] [608, 610]
- F. [606, 608, 610, 612, 629] [608, 610, 629]

**Correct Answer:** E

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En el código anterior el TreeSet "s" toma los valores pares del 606 al 613 o sea : [606,608,610,612], El TreeSet "subs" esta limitado a valores entre 608 y 611 <inclusive> extraido de "s" quedando [608,610]. La linea 20 adiciona el número 629 a el TreeSet "s",quedando [606, 608, 610, 612, 629] En la linea 21 se imprimen los dos TreeSets así: [606, 608,609, 610, 612] [608, 610]

### QUESTION 213

Given that the elements of a PriorityQueue are ordered according to natural ordering, and:

- ```
2. import java.util.*;

3. public class GetInLine {

4.     public static void main(String[] args) {

5.         PriorityQueue<String> pq = new PriorityQueue<String>();

6.         pq.add("banana");

7.         pq.add("pear");

8.         pq.add("apple");

9.         System.out.println(pq.poll() + " " + pq.peek());

10.    }

11. }
```

What is the result?

- A. apple pear
- B. banana pear
- C. apple apple
- D. apple banana
- E. banana banana

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

En este caso se adicionan 3 items a la cola "pq", en a linea 9 se invoca el metodo poll() el cual devuelve el primer item en orden natural osea "apple" y lo elimina, despues se invoca el metodo peek(); que retorna el primer item en orden natural siendo "banana" pero no lo elimina.

Queue

La colección PriorityQueue ordena sus elementos utilizando una prioridad definida por el usuario. La prioridad puede ser el orden natural, pero también se puede ordenar utilizando un objeto Comparator. La

interface Queue posee métodos que no se encuentran en ninguna otra colección:

peek(): Devuelve el primer elemento de la cola (o el que tenga mayor prioridad).

poll(): Devuelve el primer elemento de la cola (o el que tenga mayor prioridad), y luego lo elimina de la cola.

offer(T o): Agrega un elemento al final de la cola (o en el lugar donde indique su prioridad).

PriorityQueue

```
public class PriorityQueue<E>  
    extends AbstractQueue<E>  
    implements Serializable
```

An unbounded priority queue based on a priority heap. The elements of the priority queue are ordered according to their natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used. A priority queue does not permit null elements. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in `ClassCastException`).

QUESTION 214

Given a pre-generics implementation of a method:

```
11. public static int sum(List list) {  
12.     int sum = 0;  
13.     for ( Iterator iter = list.iterator(); iter.hasNext(); ) {  
14.         int i = ((Integer)iter.next()).intValue();  
15.         sum += i;  
16.     }  
17.     return sum;  
18. }
```

What three changes allow the class to be used with generics and avoid an unchecked warning? (Choose three.)

- A. Remove line 14.
- B. Replace line 14 with "int i = iter.next();".
- C. Replace line 13 with "for (int i : intList) {".
- D. Replace line 13 with "for (Iterator iter : intList) {".
- E. Replace the method declaration with "sum(List<int> intList)".
- F. Replace the method declaration with "sum(List<Integer> intList)".

Correct Answer: ACF

Section: (none)

Explanation

Explanation/Reference:

Explanation: En este caso se elimina la línea 14 porque al hacer genérica la lista con `sum(List<Integer> intList)` se puede recorrer de esta forma la lista: `for (int i : intList)` y nos evita dar casting en la línea 14 para asignar el valor a la variable "i".

QUESTION 215

Given:

```
34. HashMap props = new HashMap();  
35. props.put("key45", "some value");
```

```
36. props.put("key12", "some other value");  
37. props.put("key39", "yet another value");  
38. Set s = props.keySet();  
39. // insert code here
```

What, inserted at line 39, will sort the keys in the props HashMap?

- A. Arrays.sort(s);
- B. s = new TreeSet(s);
- C. Collections.sort(s);
- D. s = new SortedSet(s);

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Para ordenar las claves de un HashMap una buena opción es pasar estas claves a un TreeSet que ordena naturalmente los ítems usando el método de los "Map" KeySet().

Un HashMap es una clase que implementa la interfaz "Map", es una colección de objetos, (como los Arrays), pero estos no tienen orden.

Cada objeto se identifica mediante algún identificador apropiado, por ejemplo un "uuid".

El nombre HASH, hace referencia a una técnica de organización de archivos llamada hashing o "dispersion" en el cual se almacenan registros en una dirección del archivo que es generada por una función que se aplica sobre la llave del registro.

Un mapa es una estructura de Java que nos permite almacenar pares clave/valor. De tal manera que para una clave solamente tenemos un valor.

Si añadimos un nuevo elemento clave/valor cuando la clave ya existe, se sobrescribe el valor almacenado anteriormente.

La estructura a listar que utilizamos como mapa es un HashMap. Lo primero que tenemos que hacer es crear el mapa y añadirle elementos:

```
Map mapa = new HashMap();  
  
mapa.put("String","String");
```

Métodos:

void clear()
Removes all mappings from this map.

boolean containsKey(Object key)
Returns true if this map contains a mapping for the specified key.

boolean containsValue(Object value)
Returns true if this map maps one or more keys to the specified value.

Set entrySet()
Returns a collection view of the mappings contained in this map.

Object get(Object key)
Returns the value to which the specified key is mapped in this identity hash map, or null if the map contains no mapping for this key.

boolean isEmpty()

Returns true if this map contains no key-value mappings.

Set keySet()

Returns a set view of the keys contained in this map.

Object put(Object key, Object value)

Associates the specified value with the specified key in this map.

Object remove(Object key)

Removes the mapping for this key from this map if present.

int size()

Returns the number of key-value mappings in this map.

Collection values()

Returns a collection view of the values contained in this map.

QUESTION 216

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) ) return 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 overridden.
- 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 overridden.
- 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.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Los HashSet no aceptan valores repetidos , sin embargo al crear varias instancias de la clase "Person" asi sean con su variable "name" igual son dos objetos diferentes y si se almacenarían en el HashSet.

QUESTION 217

Given:

```

3. import java.util.*;

4. public class Hancock {

5.     // insert code here

6.     list.add("foo");

7. }

8. }

```

Which two code fragments, inserted independently at line 5, will compile without warnings? (Choose two.)

- A. public void addStrings(List list) {
- B. public void addStrings(List<String> list) {
- C. public void addStrings(List<? super String> list) {
- D. public void addStrings(List<? extends String> list) {

Correct Answer: BC

Section: (none)

Explanation

Explanation/Reference:

<? super String> means we can only add Strings.

In the case of String, you can only add String objects because String is a final class, and can't have subclasses.

But in a situation where a class can have subclasses, you can use the lowerbound to make additions to the collection type safe. The compiler only allows you to add classes that are the lower bound or have an "is a" relationship with the lower bound.

For example in the method

```
public static void add(List<? super Animal> list) { ... },
```

you can add an Animal instance or an instance of a subclass of Animal because the compiler knows that the only parameterized Lists you can send are List<Animal> or List<Object>, assuming Animal doesn't extend another class.

But if you have the method

```
public static void add(List<? extends Animal> list) { ... },
```

you are not allowed to add anything to the list because the compiler can't guarantee type safety.

Animal could have many subclasses and each one of them could have subclasses.

QUESTION 218

Given:

```

1. public class Threads4 {
2.     public static void main (String[] args) {

3.         new Threads4().go();

4.     }

5.     public void go() {

6.         Runnable r = new Runnable() {

```

```
7.      public void run() {  
8.          System.out.print("foo");  
9.      }  
10. };  
11.  Thread t = new Thread(r);  
12.  t.start();  
13.  t.start();  
14.  }  
15. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes normally and prints "foo".
- D. The code executes normally, but nothing is printed.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

Se genera una excepción en tiempo de ejecución pues no se puede llamar dos veces al método start(); consecutivamente esto genera un IllegalStateException que es un error que se genera cuando un hilo no está en un estado apropiado para recibir la solicitud, en este caso no puede recibir la segunda solicitud "start()". hasta que termine de ejecutarse.

QUESTION 219

Given:

```
1. public class TestOne {  
2.     public static void main (String[] args) throws Exception {  
3.         Thread.sleep(3000);  
4.         System.out.println("sleep");  
5.     }  
6. }
```

What is the result?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. The code executes normally and prints "sleep".
- D. The code executes normally, but nothing is printed.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

La sintaxis del código es la correcta, el hilo principal (main) duerme 3 segundos antes de continuar.

QUESTION 220

Given:

```
1. public class TestSeven extends Thread {  
2.     private static int x;  
3.     public synchronized void doThings() {  
4.         int current = x;  
5.         current++;  
6.         x = current;  
7.     }  
8.     public void run() {  
9.         doThings();  
10.    }  
11.}
```

Which statement is true?

- A. Compilation fails.
- B. An exception is thrown at runtime.
- C. Synchronizing the run() method would make the class thread-safe.
- D. The data in variable "x" are protected from concurrent access problems.
- E. Declaring the doThings() method as static would make the class thread-safe.
- F. Wrapping the statements within doThings() in a synchronized(new Object()) { } block would make the class thread-safe.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

E es correcto. Haciendo el método doThings() estático se asegura que solo exista una copia de ese método, este solo está bloqueado para la instancia en que se cree pero al hacerlo estático se asegura que dos hilos o mas no puedan ejecutar el método simultáneamente.

QUESTION 221

Which two code fragments will execute the method doStuff() in a separate thread? (Choose two.)

- A.

```
new Thread() {  
    public void run() { doStuff(); }  
};
```
- B.

```
new Thread() {  
    public void start() { doStuff(); }  
};
```
- C.

```
new Thread() {  
    public void start() { doStuff(); }  
}.run();
```
- D.

```
new Thread() {  
    public void run() { doStuff(); }  
}.start();
```
- E.

```
new Thread(new Runnable() {
```

```
        public void run() { doStuff(); }
    }).run();
F.    new Thread(new Runnable() {
        public void run() { doStuff(); }
    }).start();
```

Correct Answer: DF

Section: (none)

Explanation

Explanation/Reference:

A y B son incorrectos el hilo no se ha inicializado con ".run()" o ".start()".

C y E son incorrectos se llama al metodo(run) y no al inicializador start() que permite ejecutar el hilo de forma separada.

QUESTION 222

Given:

```
11. public static void main(String[] args) {
12.     Object obj = new int[] { 1, 2, 3 };
13.     int[] someArray = (int[])obj;
14.     for (int i : someArray) System.out.print(i + " ");
15. }
```

What is the result?

- A. 1 2 3
- B. Compilation fails because of an error in line 12.
- C. Compilation fails because of an error in line 13.
- D. Compilation fails because of an error in line 14.
- E. A ClassCastException is thrown at runtime.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es correcta y el casting de la linea 13 si se puede realizar puesto que obj si conoce a "int[]" con la anterior declaraci3n de la linea 12.

QUESTION 223

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*/ }
    }

```

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

B es incorrecto no se puede implementar una clase ella se extiende.

C es incorrecto no se puede referir al metodo load() de esa forma: Info.load

D y E son ncorrectos porque las clases no se implementan se extienden.

F es incorrecto se implementa de forma erronea el metodo load().

QUESTION 224

Given:

```

11. public static void parse(String str) {
12.     try {
13.         float f = Float.parseFloat(str);
14.     } catch (NumberFormatException nfe) {
15.         f = 0;
16.     } finally {
17.         System.out.println(f);
18.     }
19. }
20. public static void main(String[] args) {
21.     parse("invalid");
22. }

```

What is the result?

- A. 0.0
- B. Compilation fails.
- C. A ParseException is thrown by the parse method at runtime.
- D. A NumberFormatException is thrown by the parse method at runtime.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

La linea 21 llama al método parse que intenta convertir el String "invalid" en un Float es to da error. Pero antes que esto la compilación falla por que en el bloque catch se intenta acceder a la variable "f" la cual pertecece a "try" y solo es vista por el mismo.

QUESTION 225

Given

```
11. public interface Status {  
  
12     . /* insert code here */ int MY_VALUE = 10;  
  
13. }
```

Which three are valid on line 12? (Choose three.)

- A. final
- B. static
- C. native
- D. public
- E. private
- F. abstract
- G. protected

Correct Answer: ABD

Section: (none)

Explanation

Explanation/Reference:

A B y D las variables de las interfaces unicamente son public,static y final por ello se les puede agregar esos modificadores de acceso.

C es incorrecto.Unicamente los métodos son "native".

E y G son incorrectos las variables unicamente pueden ser public , static y final.

F las variables no pueden ser abstractas.

QUESTION 226

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.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La implementación es la correcta y se imprime "test"

QUESTION 227

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 succeeds.
- B. Exactly one class does NOT compile.
- C. Exactly two classes do NOT compile.
- D. Exactly four classes do NOT compile.
- E. Exactly three classes do NOT compile.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Las clases que no compilan son:

La clase B línea 13 genera error porque el método m1() no se implementa.

La clase D línea 15 no compila, el método m1 no se implementa bien debe estar sin parámetros.

Las clases abstractas **E, F y G** sí compilan ya que no es necesario que declaren o implementen métodos de la interfaz puesto que son abstractas.

QUESTION 228

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[] a) { 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.

Correct Answer: D

Section: (none)

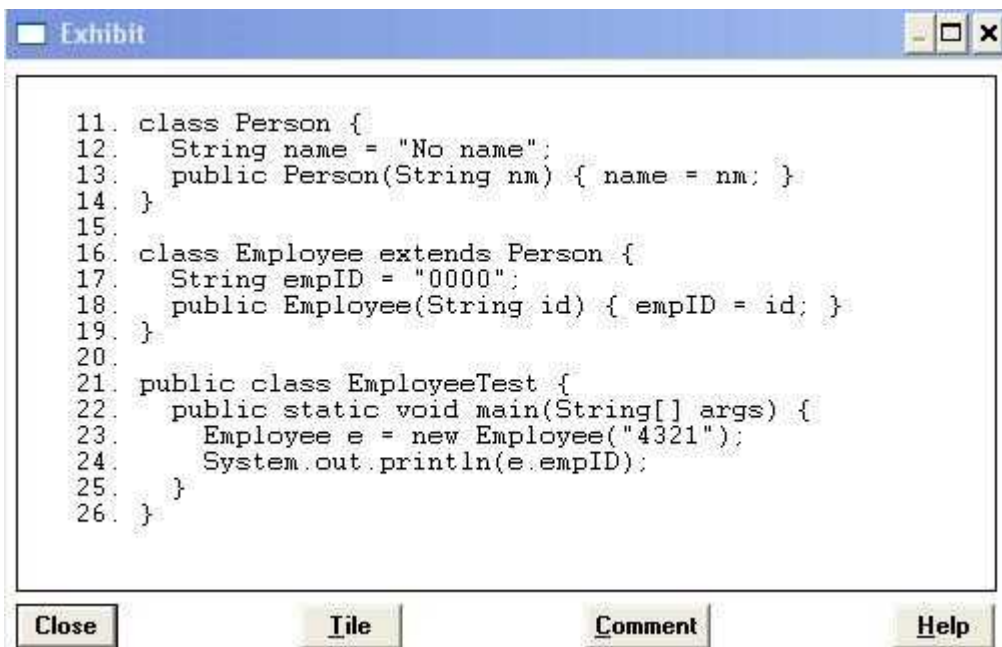
Explanation

Explanation/Reference:

Al invocar una nueva instancia abstracta en la línea 31 ya que son subclases se ejecutan primero los constructores de las superclases y después las subclases, se ejecuta el constructor de la línea 22 luego 25 y 28 imprimiendo "123".

QUESTION 229

Click the Exhibit button. 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 exception is thrown at runtime.
- D. Compilation fails because of an error in line 18.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La compilación falla pues internamente dentro de el constructor de la clase Employee en la línea 18 se está

llamando al super() constructor por defecto y este no existe en la clase person.

QUESTION 230

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; }

Correct Answer: CD

Section: (none)

Explanation

Explanation/Reference:

Las dos opciones **Cy D** hacen referencia a un a superclase que dentro de su metodo foo() devuelven una clase padre.

QUESTION 231

DRAG DROP

Click the Task button.

Chain these constructors to create objects to read from a file named "in" and to write to a file named "out"

reader = "in");

writer = "out");

Constructors

| | | |
|----------------------|----------------------|----------------------|
| <input type="text"/> | <input type="text"/> | <input type="text"/> |
| <input type="text"/> | <input type="text"/> | <input type="text"/> |

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Drag and Drop

Chain these constructors to create objects to read from a file named "in" and to write to a file named "out"

```
reader = new BufferedReader( new FileReader( "in" ) );
```

```
writer = new PrintWriter( new BufferedWriter( new FileWriter( "out" ) ) );
```

Constructors

- new FileReader()
- new FileWriter()
- new BufferedReader()
- new BufferedWriter()
- new PrintWriter()

Done

QUESTION 232

DRAG DROP

Click the Task button.

Place each Collection Type on its function. Note: Not all functions will be used.

| Function | Collection Type |
|--|----------------------|
| provides array manipulation utilities | java.util.SortedSet |
| provides collection manipulation utilities | java.util.Arrays |
| defines base methods for all array objects | java.util.Iterator |
| defines base methods for all collection objects | java.util.TreeSet |
| provides a concrete implementation of an ordered set | java.util.Collection |
| defines base methods for an ordered set | |
| defines methods for linear access to a collection | |
| defines methods for random access to a collection | |

Done

- A.
- B.
- C.
- D.

Correct Answer: A
Section: (none)
Explanation

Explanation/Reference:

Place each Collection Type on its function. Note: Not all functions will be used.

| Function | Collection Type |
|---------------------------------|---------------------------------|
| <div>java.util.Collection</div> | <div>java.util.SortedSet</div> |
| <div>java.util.TreeSet</div> | <div>java.util.Arrays</div> |
| <div>java.util.Arrays</div> | <div>java.util.Iterator</div> |
| <div>java.util.SortedSet</div> | <div>java.util.TreeSet</div> |
| <div>java.util.Arrays</div> | <div>java.util.Collection</div> |
| <div>java.util.Collection</div> | |
| <div>java.util.SortedSet</div> | |
| <div>java.util.Iterator</div> | <div>Done</div> |

QUESTION 233
DRAG DROP
Click the Task button.

Drag and Drop

Given:

```

10. Runnable r = new Runnable() {
11.     public void run() {
12.         try {
13.             Thread.sleep(1000);
14.         } catch (InterruptedException e) {
15.             System.out.println("interrupted");
16.         }
17.         System.out.println("ran");
18.     }
19. };
20. Thread t = new Thread(r);
21. t.start();
22. System.out.println('started');
23. t.sleep(2000);
24. System.out.println('interrupting');
25. t.interrupt();
26. System.out.println('ended');

```

Assume that sleep(n) executes in exactly n milliseconds, and all other code executes in an insignificant amount of time.

Place the fragments in the output area to show the result of running this code.

| Output | Fragments |
|------------|-----------------------|
| Place here | interrupted |
| Place here | ran |
| Place here | started |
| Place here | interrupting |
| Place here | ended |
| Place here | InterruptedException: |
| | (no more output) |

Done

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

El método interrupt() como lo indica el nombre interrumpe el hilo seleccionado invocando primero el método checkAccess para verificar el estado del hilo dependiendo de este metodo se genera una SecurityException, si el hilo esta bloqueado por algun wait, join, sleep, se lanza una InterruptedException.

Drag and Drop

Given:

```

10. Runnable r = new Runnable() {
11.     public void run() {
12.         try {
13.             Thread.sleep(1000);
14.         } catch (InterruptedException e) {
15.             System.out.println("interrupted");
16.         }
17.         System.out.println("ran");
18.     }
19. };
20. Thread t = new Thread(r);
21. t.start();
22. System.out.println('started");
23. t.sleep(2000);
24. System.out.println('interrupting");
25. t.interrupt();
26. System.out.println('ended");

```

Assume that sleep(n) executes in exactly n milliseconds, and all other code executes in an insignificant amount of time.

Place the fragments in the output area to show the result of running this code.

| Output | Fragments |
|------------------|----------------------|
| started | interrupted |
| ran | ran |
| interrupting | started |
| ended | interrupting |
| (no more output) | ended |
| | InterruptedException |
| | (no more output) |

Done

QUESTION 234

DRAG DROP

Click the Task button.

Add methods to the Beta class to make it compile correctly.

```

class Alpha {
    public void bar( int... x ) { }
    public void bar( int x ) { }
}

```

```

public class Beta extends Alpha {

```

Place here

Place here

Place here

```

}

```

Done

Methods

private void bar(int x) { }

public void bar(int x) { }

public int bar(String x) { return 1; }

public Alpha bar(int x) { }

public void bar(int x, int y) { }

public int bar(int x) { return x; }

- A.
- B.
- C.

D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Add methods to the Beta class to make it compile correctly.

```
class Alpha {  
    public void bar( int... x ) { }  
    public void bar( int x ) { }  
}
```

```
public class Beta extends Alpha {
```

```
    public void bar( int x, int y ) { }
```

```
    public int bar( String x ) { return 1; }
```

```
    public void bar( int x ) { }
```

```
}
```

Done

Methods

```
private void bar( int x ) { }
```

```
public void bar( int x ) { }
```

```
public int bar( String x ) { return 1; }
```

```
public Alpha bar( int x ) { }
```

```
public void bar( int x, int y ) { }
```

```
public int bar( int x ) { return x; }
```

QUESTION 235

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.         Payload 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);`

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Al llamar al método `setWeight` se modifica la variable `weight` a 420, ya que la clase tiene implementado el método `toString()` esta llama al metodo `toString` del wrap "Integer" y devuelve la cadena "420" .

QUESTION 236

Given:

```
11. public void genNumbers() {  
12.     ArrayList numbers = new ArrayList();  
13.     for (int i=0; i<10; i++) {  
14.         int value = i * ((int) Math.random());  
15.         Integer intObj = new Integer(value);  
16.         numbers.add(intObj);  
17.     }  
18.     System.out.println(numbers);  
19. }
```

Which line of code marks the earliest point that an object referenced by `intObj` becomes a candidate for garbage collection?

- A. Line 16
- B. Line 17
- C. Line 18
- D. Line 19
- E. The object is NOT a candidate for garbage collection.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

El objeto que referencia la variable `intObj` queda libre para eliminar por el gc al finalizar el método pues se hace referencia a el en todo el bloque.

QUESTION 237

Given a correctly compiled class whose source code is:

```
1. package com.sun.sjcp;  
2. public class Commander {  
3.     public static void main(String[] args) {  
4.         // more code here  
5.     }
```


6. }

Assume that the class file is located in `/foo/com/sun/sjcp/`, the current directory is `/foo/`, and that the classpath contains `"."` (current directory). Which command line correctly runs `Commander`?

- A. `java Commander`
- B. `java com.sun.sjcp.Commander`
- C. `java com/sun/sjcp/Commander`
- D. `java -cp com.sun.sjcp Commander`
- E. `java -cp com/sun/sjcp Commander`

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

la sintaxis de la opción **B** es la correcta se refiere a la ruta definida en el package a través de puntos.

QUESTION 238

Given:

```
11. public static void test(String str) {  
12.     int check = 4;  
13.     if (check = str.length()) {  
14.         System.out.print(str.charAt(check -= 1) + ", ");  
15.     } else {  
16.         System.out.print(str.charAt(0) + ", ");  
17.     }  
18. } and the invocation:  
21. test("four");  
22. test("tee");  
23. test("to");
```

What is the result?

- A. `r, t, t,`
- B. `r, e, o,`
- C. Compilation fails.
- D. An exception is thrown at runtime.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

La compilación falla en la línea 13, se debe utilizar el operador relacional `==` para que nos retorne un booleano válido para el `"if"`. Por lo demás la sintaxis es correcta.

QUESTION 239

A developer is creating a class `Book`, that needs to access class `Paper`. The `Paper` class is deployed in a JAR named `myLib.jar`. Which three, taken independently, will allow the developer to use the `Paper` class while compiling the `Book` class? (Choose three.)

- A. The JAR file is located at \$JAVA_HOME/jre/classes/myLib.jar.
- B. The JAR file is located at \$JAVA_HOME/jre/lib/ext/myLib.jar..
- C. The JAR file is located at /foo/myLib.jar and a classpath environment variable is set that includes /foo/myLib.jar/Paper.class.
- D. The JAR file is located at /foo/myLib.jar and a classpath environment variable is set that includes /foo/myLib.jar.
- E. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -cp /foo/myLib.jar/ Paper Book.java.
- F. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -d /foo/myLib.jar Book.java
- G. The JAR file is located at /foo/myLib.jar and the Book class is compiled using javac -classpath /foo/myLib.jar Book.java

Correct Answer: BDG

Section: (none)

Explanation

Explanation/Reference:

B es correcto la carpeta \$JAVA_HOME/jre/lib/ext/myLib.jar. es donde se alojan los archivos externos y donde java busca por defecto las librerías requeridas.

G es correcto el -classpath se declara bien incluyendo el .jar donde se aloja la clase Paper.

D es correcto si se incluye la línea /foo/myLib.jar. dentro de la variable classpath y se mete dentro de esta ruta el .jar se podrá compilar correctamente.

A es incorrecto la ruta de las librerías externas no es esa.

C es incorrecto se debe hacer la declaración sin el slash al final.

E es incorrecto se desea acceder a la clase Paper dentro de myLib.jar por ello debe aparecer de último dentro de la declaración del classpath.

F es incorrecto el parámetro -d solo se aplica para decir donde tienen que ir los archivos .class compilados.

QUESTION 240

Given:

1. package com.company.application;
- 2.
3. public class MainClass {
4. public static void main(String[] args) {}
5. }

And MainClass exists in the /apps/com/company/application directory. Assume the CLASSPATH environment variable is set to "." (current directory).

Which two java commands entered at the command line will run MainClass? (Choose two.)

- A. java MainClass if run from the /apps directory
- B. java com.company.application.MainClass if run from the /apps directory
- C. java -classpath /apps com.company.application.MainClass if run from any directory
- D. java -classpath . MainClass if run from the /apps/com/company/application directory
- E. java -classpath /apps/com/company/application:. MainClass if run from the /apps directory
- F. java com.company.application.MainClass if run from the /apps/com/company/application directory

Correct Answer: BC

Section: (none)

Explanation

Explanation/Reference:

A es incorrecto. el compilador asume que la clase a ejecutar está alojada en el directorio /apps y no es así.

D es incorrecto no se puede reconocer la ruta con el identificador "."

E es incorrecto el simbolo ":" no se reconoce en esta declaración.

F es incorrecto el compilador intentara encontrar la ruta de la clase en: /apps/com/company/application/com/company/application/MainClass

Classpath (Java)

En el lenguaje de programación Java se entiende por Classpath una opción admitida en la línea de órdenes o mediante variable de entorno que indica a la Máquina Virtual de Java dónde buscar paquetes y clases definidas por el usuario a la hora de ejecutar programas.

QUESTION 241

Given:

```
3. public class Batman {  
4.     int squares = 81;  
5.     public static void main(String[] args) {  
6.         new Batman().go();  
7.     }  
8.     void go() {  
9.         incr(++squares);  
10.        System.out.println(squares);  
11.    }  
12.    void incr(int squares) { squares += 10; }  
13. }
```

What is the result?

- A. 81
- B. 82
- C. 91
- D. 92
- E. Compilation fails.
- F. An exception is thrown at runtime.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

El resultado de la variable de instancia squares es 82 valor que toma despues del incremento en la linea 9. La variable local squares es independiente de la de instancia y almacena el valor 92. el valor que se imprime es el de la variable de instancia por ello es 82.

QUESTION 242

Given a class Repetition:

```
1. package utils;  
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;

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

F es la respuesta correcta pues cumple con la verdadera sintaxis para importar metodos estaticos de otra clase.

Importacion Metodos Estaticos

Una innovación de java 5.0 son las llamadas importaciones estáticas que permiten llamar a un método o propiedad estática sin necesidad de hacer referencia al nombre de su clase.

La sintaxis general, es:

```
import static paquete.Clase.metodo_o_propiedad_static; //Para un sólo método o propiedad.
```

QUESTION 243

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.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

La sintaxis esta bien y el programa compila sin problema.

QUESTION 244

Given that Triangle implements Runnable, and:

31. void go() throws Exception {

32. Thread t = new Thread(new Triangle());

33. t.start();

34. for(int x = 1; x < 100000; x++) {

35. //insert code here

36. if(x%100 == 0) System.out.print("g");

37. } }

38. public void run() {

39. try {

40. for(int x = 1; x < 100000; x++) {

41. // insert the same code here

42. if(x%100 == 0) System.out.print("t");

43. }

44. } catch (Exception e) { }

45. }

Which two statements, inserted independently at both lines 35 and 41, tend to allow both threads to temporarily pause and allow the other thread to execute? (Choose two.)

- A. Thread.wait();
- B. Thread.join();
- C. Thread.yield();
- D. Thread.sleep(1);
- E. Thread.notify();

Correct Answer: CD

Section: (none)

Explanation

Explanation/Reference:

Las respuestas correctas son al **C** y **D**, Los métodos yield() y sleep() son métodos que pausan los hilos que se están ejecutando.

El método join deja que se termine la ejecución de un hilo y luego el se ejecuta.

El método wait() y notify() trabajan en conjunto para pausar un hilo(wait()) mientras otro realiza una tarea y luego notificarlo (notify()) a continuar.

QUESTION 245

Which two code fragments will execute the method doStuff() in a separate thread? (Choose two.)

- A. new Thread() {
 public void run() { doStuff(); }
};
- B. new Thread() {
 public void start() { doStuff(); }
};
- C. new Thread() {
 public void start() { doStuff(); }
}.run();
- D. new Thread() {
 public void run() { doStuff(); }
}.start();
- E. new Thread(new Runnable() {
 public void run() { doStuff(); }
}).run();
- F. new Thread(new Runnable() {
 public void run() { doStuff(); }
}).start();

Correct Answer: DF

Section: (none)

Explanation

Explanation/Reference:

A y B son incorrectos el hilo no se ha inicializado con ".run()" o ".start()".

C y E son incorrectos se llama al metodo(run) y no al inicializador start() que permite ejecutar el hilo de forma separada.

QUESTION 246

Given: public class NamedCounter {

 private final String name;

 private int count;

 public NamedCounter(String name) { this.name = name; }

```

public String getName() { return name; }

public void increment() { count++; }

public int getCount() { return count; }

public void reset() { count = 0; }
}

```

Which three changes should be made to adapt this class to be used safely by multiple threads? (Choose three.)

- A. declare reset() using the synchronized keyword
- B. declare getName() using the synchronized keyword
- C. declare getCount() using the synchronized keyword
- D. declare the constructor using the synchronized keyword
- E. declare increment() using the synchronized keyword

Correct Answer: ACE

Section: (none)

Explanation

Explanation/Reference:

Para usar de forma segura hilos en la clase NameCounter es necesario declarar sinchronized los metodos que contiene :

reset(), getCount() e increment();

QUESTION 247

Given that t1 is a reference to a live thread, which is true?

- A. The Thread.sleep() method can take t1 as an argument.
- B. The Object.notify() method can take t1 as an argument.
- C. The Thread.yield() method can take t1 as an argument.
- D. The Thread.setPriority() method can take t1 as an argument.
- E. The Object.notify() method arbitrarily chooses which thread to notify.

Correct Answer: E

Section: (none)

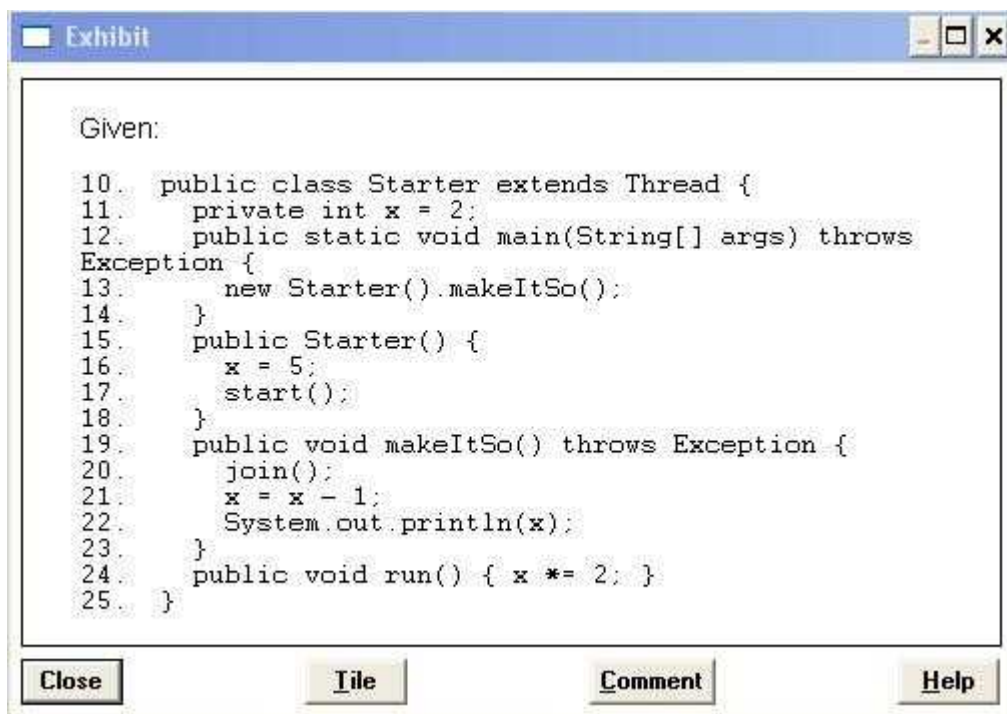
Explanation

Explanation/Reference:

Dado que todas las clases son hijas de Oobject, Object.notify() notifica arbitrariamente cualquier hilo que se este ejecutando.

QUESTION 248

Click the Exhibit button. What is the output if the main() method is run?



- A. 4
- B. 5
- C. 8
- D. 9
- E. Compilation fails.
- F. An exception is thrown at runtime.
- G. It is impossible to determine for certain.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La sintaxis es correcta, en este código se ejecuta primero el constructor Starter() asignando a x un valor de 5, posteriormente se ejecuta el método run() o la declaración de makeItSo, en este ultimo método el hilo principal "main" es detenido en la línea 20 hasta tanto no termine el método "run" que pertenece a un segundo hilo "Thread-0" por lo tanto run hace que "x" sea igual a "10", por ultimo se le resta 1 en la línea 21 y da como resultado 9.

QUESTION 249

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.

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

En el código anterior se crea una clase anónima que es de tipo TestB por ello el metodo que se ejecuta es el perteneciente a TestB imprimiendo "TestB".

QUESTION 250

Which two code fragments correctly create and initialize a static array of int elements? (Choose two.)

- A. `static final int[] a = { 100,200 };`
- B. `static final int[] a;`
`static { a=new int[2]; a[0]=100; a[1]=200; }`
- C. `static final int[] a = new int[2]{ 100,200 };`
- D. `static final int[] a;`
`static void init() { a = new int[3]; a[0]=100; a[1]=200; }`

Correct Answer: AB

Section: (none)

Explanation

Explanation/Reference:

La sintaxis de la opción A es la correcta para declarar instanciar e inicializar un arreglo.

B es correcto pues con el metodo `static{}` se pueden instanciar e inicializar las variables estaticas.

C es incorrecta pues la sintaxis no esta bien escrita en: `new int[2]{ 100,200 }` deberia ser igual que la opcion A.

D es incorrecta pues la inicializacion de una variable `static final` solo es posible en la declaración, en el constructor o en el bloque de inicializacion `static{}`.

Bloques de Inicializacion `{}` `static{}`

Son bloques pero sin nombre. Si se dice que un constructor es como un método sin tipo, un inicializador es un bloque sin nombre

Los bloques de inicialización se ejecutan cuando la clase es cargada por primera vez (un bloque de inicialización estático) o cuando se crea una instancia (un bloque de inicialización estático)

Primero se ejecutarán los inicializadores de clase , los de instancia y al final el constructor.

QUESTION 251

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 */ }

Correct Answer: BE

Section: (none)

Explanation

Explanation/Reference:

A es inválido una Clase extiende no implementa otra clase.

C es inválido ya que siendo Circle clase concreta se necesita implementar el método draw.

D y F son incorrectos , las clases se extienden no se implementan.

QUESTION 252

Given:

```

10. class Nav{
11.     public enum Direction { NORTH, SOUTH, EAST, WEST }
12. }
13. public class Sprite{
14.     // insert code here
15. }

```

Which code, inserted at line 14, allows the Sprite class to compile?

- A. Direction d = NORTH;
- B. Nav.Direction d = NORTH;
- C. Direction d = Direction.NORTH;
- D. Nav.Direction d = Nav.Direction.NORTH;

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La sintaxis de la opción D es la correcta para crear un enum.

QUESTION 253

Given:

```
5. class Atom {  
6.     Atom() { System.out.print("atom "); }  
7. }  
8. class Rock extends Atom {  
9.     Rock(String type) { System.out.print(type); }  
10. }  
11. public class Mountain extends Rock {  
12.     Mountain() {  
13.         super("granite ");  
14.         new Rock("granite ");  
15. }  
16. public static void main(String[] a) { new Mountain(); }  
17. }
```

What is the result?

- A. Compilation fails.
- B. atom granite
- C. granite granite
- D. atom granite granite
- E. An exception is thrown at runtime.
- F. atom granite atom granite

Correct Answer: F

Section: (none)

Explanation

Explanation/Reference:

A. es incorrecta la sintaxis esta bien.

F. es correcta la clase mountain llama en la línea 13 a su super constructor Rock(String type) el cual lo que hace es ocultamente llamar primero a su superconstructor en atom que es Atom() el cual genera la cadena "atom " y posteriormente ejecuta System.out.print(type) generando "granite " y este proceso se repite en la llamada de la clase de la línea 14;

Constructores y herencia en Java

Sean las clases "Base" y "Derivada".

TODAS las clases de Java tienen AL MENOS un constructor. Siempre.

Si no ponemos ninguno, Java les pone automáticamente uno de este estilo: Base(){}

Si ponemos aunque sólo sea uno (sea como sea), Java ya no añadirá el suyo.

La primera línea de un constructor es SIEMPRE una llamada al constructor de la clase base.

Si no ponemos ninguna llamada, Java pondrá automáticamente una de este estilo: `super()`.
Si ponemos una nuestra, Java ya no añadirá la suya.

En una clase derivada, se ejecuta antes el constructor de la clase base que el suyo (recuérdese que la llamada a "super" es la primera instrucción del constructor).

QUESTION 254

Given:

```
1. public class A {  
2.     public void doit() {  
3.     }  
4.     public String doit() {  
5.         return "a";  
6.     }  
7.     public double doit(int x) {  
8.         return 1.0;  
9.     }  
10. }
```

What is the result?

- A. An exception is thrown at runtime.
- B. Compilation fails because of an error in line 7.
- C. Compilation fails because of an error in line 4.
- D. Compilation succeeds and no runtime errors with class A occur.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Así en los métodos `public void doit()` y `public String doit()` se retornen valores de diferente tipo lo que hace distinto un método para sobrecargar es el tipo y cantidad de parámetros.

QUESTION 255

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[] a) { 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.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Al invocar una nueva instancia abstracta en la linea 31 ya que son subclases se ejecutan primero los constructores de las superclases y despues las subclases, se ejecuta el constructor de la linea 22 luego 25 y 28 imprimiendo "123".

QUESTION 256

Given:

```

11. public class Rainbow {

12.     public enum MyColor {

13.         RED(0xff0000), GREEN(0x00ff00), BLUE(0x0000ff);

14.         private final int rgb;

15.         MyColor(int rgb) { this.rgb = rgb; }

16.         public int getRGB() { return rgb; }

17.     };

18.     public static void main(String[] args) {

19.         // insert code here

20.     }

21. }

```

Which code fragment, inserted at line 19, allows the Rainbow class to compile?

- A. MyColor skyColor = BLUE;
- B. MyColor treeColor = MyColor.GREEN;
- C. if(RED.getRGB() < BLUE.getRGB()) { }
- D. Compilation fails due to other error(s) in the code.
- E. MyColor purple = new MyColor(0xff00ff);
- F. MyColor purple = MyColor.BLUE + MyColor.RED;

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

B es la forma correcta de establecer un enum MyColor.

QUESTION 257

A company that makes Computer Assisted Design (CAD) software has, within its application, some utility classes that are used to perform 3D rendering tasks. The company's chief scientist has just improved the performance of one of the utility classes' key rendering algorithms, and has assigned a programmer to replace the old algorithm with the new algorithm. When the programmer begins researching the utility classes, she is happy to discover that the algorithm to be replaced exists in only one class. The programmer reviews that class's API, and replaces the old algorithm with the new algorithm, being careful that her changes adhere strictly to the class's API. Once testing has begun, the programmer discovers that other classes that use the class she changed are no longer working properly. What design flaw is most likely the cause of these new bugs?

- A. Inheritance
- B. Tight coupling
- C. Low cohesion
- D. High cohesion
- E. Loose coupling
- F. Object immutability

Correct Answer: B

Section: (none)

Explanation

Explanation/Reference:

El problema anterior se debe al estrecho acoplamiento (Tight coupling) que la clase que se va a reemplazar tenía con otros miembros del programa o sea no era del todo independiente como debiera.

coupling (acoplamiento)

El acoplamiento es en qué medida esta clase se integra con las implementaciones de otras clases, que un cambio en cualquier otra clase se traducirá en un cambio en esta clase. Por lo tanto siempre es mejor programar con interfaces.

cohesion (cohesion)

Cohesión significa que el conjunto de una clase está integrada. Una clase debe ser responsable de sí misma, debe hacer una cosa y en la medida de lo posible, hacer todo lo posible para hacer funcionar esa cosa. Un elemento es cohesivo a medida que cambia el elemento entero cuando el sistema necesita cambiar.

QUESTION 258

Given:

- 11. `abstract class Vehicle { public int speed() { 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 RaceCar();`

```

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
- E. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

Los objetos racer, car y vehicle son de tipos distintos pero se les asigno a todos un objeto RaceCar por ello el método que se ejecuta es el de la linea 13 generando la salida 150,150,150.

QUESTION 259

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.

Correct Answer: ABCF

Section: (none)

Explanation

Explanation/Reference:

D es incorrecta se necesitaria que BabyRaccoon extendiera a Raccoon para ser verdad.

E es incorrecta dentro de la clase BabyRaccoon no existe ningun objeto de tipo Mammal.

QUESTION 260

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.

Correct Answer: E

Section: (none)

Explanation

Explanation/Reference:

El error ocurre porque no se puede llamar a super o this dentro de un método estático.

QUESTION 261

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 commission;
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.commission = commission;`
- 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);`

Correct Answer: AE

Section: (none)

Explanation

Explanation/Reference:

En el constructor de la clase SalesPerson se hace como siempre una llamada al superconstructor `super()` y esto genera error porque el constructor `super()` no lo posee, por ello cualquier código insertado debe tener un llamado al constructor que existe en la clase padre employee.

QUESTION 262

Given:

```
11. class A {  
12.     public void process() { System.out.print("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.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La compilación falla porque el metodo sobreescrito debe tambien implementar un llamado a throws ya que el método que lo sobreescribe lo declara.

QUESTION 263

Given a method that must ensure that its parameter is not null:

```
11. public void someMethod(Object value) {  
12. // check for null value ...  
20. System.out.println(value.getClass());  
21. }
```

What, inserted at line 12, is the appropriate way to handle a null value?

- A. assert value == null;
- B. assert value != null, "value is null";
- C. if (value == null) {
 throw new AssertionError("value is null");
}
- D. if (value == null) {
 throw new IllegalArgumentException("value is null"); }

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La sintaxis de la opcion **D** es la correcta para evaluar si es null la variable y lanzar una "exception".

QUESTION 264

Given:

```
11. public static void main(String[] args) {  
12.     try {  
13.         args = null;  
14.         args[0] = "test";  
15.         System.out.println(args[0]);  
16.     } catch (Exception ex) {  
17.         System.out.println("Exception");  
18.     } catch (NullPointerException npe) {  
19.         System.out.println("NullPointerException");  
20.     }  
21. }
```

What is the result?

- A. test
- B. Exception
- C. Compilation fails.
- D. NullPointerException

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Este código produce un error en compilación en la línea 16, se da porque la excepción en la línea 16 es capturada como "Exception" ya que es más global o general que el catch de la línea 18 :
NullPointerException.

QUESTION 265

Given:

```
11. public static Iterator reverse(List list) {  
12.     Collections.reverse(list);  
13.     return list.iterator();  
14. }  
  
15. public static void main(String[] args) {  
16.     List list = new ArrayList();  
17.     list.add("1"); list.add("2"); list.add("3");  
18.     for (Object obj: reverse(list))  
19.         System.out.print(obj + " ");  
20. }
```

What is the result?

- A. 3, 2, 1,
- B. 1, 2, 3,
- C. Compilation fails.
- D. The code runs with no output.
- E. An exception is thrown at runtime.

Correct Answer: C

Section: (none)

Explanation

Explanation/Reference:

Existen dos errores que no permitirían compilar correctamente, uno es que las colecciones están en java.util y la declaración del import para importar estas clases no se encuentra, el segundo error ocurre en la línea 18 porque la declaración del for está mal debería ser: for (Object obj: list).

QUESTION 266

Given:

```
11. public class Test {  
  
12.     public static void main(String [] args) {
```

```

13.     int x = 5;
14.     boolean b1 = true;
15.     boolean b2 = false;
16.
17.     if ((x == 4) && !b2 )
18.         System.out.print("1 ");
19.     System.out.print("2 ");
20.     if ((b2 = true) && b1 )
21.         System.out.print("3 ");
22. }
23. }

```

What is the result?

- A. 2
- B. 3
- C. 1 2
- D. 2 3
- E. 1 2 3
- F. Compilation fails.
- G. An exception is thrown at runtime.

Correct Answer: D

Section: (none)

Explanation

Explanation/Reference:

La sintaxis esta bien , en la linea 19 escribe la cadena "2" ya que no esta dentro de el "if" , la linea 21 imprime "3" ya que la condicion del if se cumple, dando como resultado "2 3 ".

QUESTION 267

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.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Se crea una clase SubX que extiende otra clase creada "X", en el main se crea una clase anónima de SubX y se llama su método foo(), el cual llama en la línea 15 el constructor de la clase "X" imprimiendo "X", luego en la línea 16 se evalúa si el if es verdadero y como por defecto es "true" se lanza una RuntimeException y termina el programa porque el método llamante "main" no trata la excepción.

QUESTION 268

Given:

- ```
1. public class Mule {
2. public static void main(String[] args) {
3. boolean assert = true;
4. if(assert) {
5. System.out.println("assert is true");
6. }
7. }
8. }
```

Which command-line invocations will compile?

- A. javac Mule.java
- B. javac -source 1.3 Mule.java
- C. javac -source 1.4 Mule.java
- D. javac -source 1.5 Mule.java

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Ya que el parámetro -source se usa para hacer compatible el código con una versión de Java, dado que los assert se crearon en Java 1.4 se debe compilar con una versión menor para que no genere error la palabra reservada assert de la línea 3.

El error es el siguiente: as of release 1.4, 'assert' is a keyword, and may not be used as an identifier (use -source 1.3 or lower to use 'assert' as an identifier)

En la ayuda de javac nos dice:

-source<release> Provide source compatibility with specified release

#### QUESTION 269

Given:

```
11. public static Collection get() {
12. Collection sorted = new LinkedList();
13. sorted.add("B"); sorted.add("C"); sorted.add("A");
14. return sorted;
15. }

16. public static void main(String[] args) {
17. for (Object obj: get()) {
18. System.out.print(obj + ", ");
19. }
20. }
```

What is the result?

- A. A, B, C,
- B. B, C, A,
- C. Compilation fails.
- D. The code runs with no output.
- E. An exception is thrown at runtime.

**Correct Answer:** B

**Section:** (none)

**Explanation**

#### Explanation/Reference:

Esta es una forma válida de recorrer una coleccion incluyendo un método dentro de un for -each como en la línea 17.

#### QUESTION 270

Given:

```
11. public void testIfA() {
12. if (testIfB("True")) {
13. System.out.println("True");
14. } else {
15. System.out.println("Not true");
16. }
17. }

18. public Boolean testIfB(String str) {
19. return Boolean.valueOf(str);
20. }
```

What is the result when method testIfA is invoked?

- A. True
- B. Not true
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error at line 12.
- E. Compilation fails because of an error at line 19.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La clase o el wrap : "Boolean" posee métodos para pasar de booleano a String : toString(boolean b) y para pasar un String a boolean: valueOf(). Por ello el código anterior en la línea 12 testIfB da "true" e imprime "True" en la siguiente línea.

Class Boolean

```
public final class Boolean
extends Object
implements Serializable
```

The Boolean class wraps a value of the primitive type boolean in an object. An object of type Boolean contains a single field whose type is boolean.

In addition, this class provides many methods for converting a boolean to a String and a String to a boolean, as well as other constants and methods useful when dealing with a bo

#### **QUESTION 271**

Click the Exhibit button. Given: ClassA a = new ClassA(); a.methodA(); What is the result?

```
10. public class ClassA {
11. public void methodA() {
12. ClassB classB = new ClassB();
13. classB.getValue();
14. }
15. }

And:

20. class ClassB {
21. public ClassC classC;
22.
23. public String getValue() {
24. return classC.getValue();
25. }
26. }

And:

30. class ClassC {
31. public String value;
32.
33. public String getValue() {
34. value = "ClassB";
35. return value;
36. }
37. }
```

- A. Compilation fails.
- B. ClassC is displayed.
- C. The code runs with no output.
- D. An exception is thrown at runtime.

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El error de el código anterior se encuentra en la línea 24 de la clase "ClassB" no se puede invocar el método getValue() de la clase "ClassC" puesto que no es un método estatico, hay que invocarlo a través de una instancia.

El bloque del método debería quedar así:

```
ClassC miClase= new ClassC();
return miClase.getValue();
```

**QUESTION 272**

Click the Exhibit button.

Given:

```
31. public void method() {
32. A a = new A();
33. a.method1();
34. }
```

Which statement is true if a TestException is thrown on line 3 of class B?



```
1. public class A {
2. public void method1() {
3. try {
4. B b = new B();
5. b.method2();
6. // more code here
7. } catch (TestException te) {
8. throw new RuntimeException(te);
9. }
10. }
11. }

12.
13. public class B {
14. public void method2() throws TestException {
15. // more code here
16. }
17. }

18.
19. public class TestException extends Exception {
20. }
```

- A. Line 33 must be called within a try block.
- B. The exception thrown by method1 in class A is not required to be caught.
- C. The method declared on line 31 must be declared to throw a RuntimeException.
- D. On line 5 of class A, the call to method2 of class B does not need to be placed in a try/catch block.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

La excepción si se lanza en el método "method2" es reenviada a su método llamante en este caso method1() el cual la trata, por ello no se necesita capturar.

**QUESTION 273**

Given that the elements of a PriorityQueue are ordered according to natural ordering, and:

- 2. import java.util.\*;
- 3. public class GetInLine {
- 4. public static void main(String[] args) {
- 5. PriorityQueue<String> pq = new PriorityQueue<String>();
- 6. pq.add("banana");
- 7. pq.add("pear");
- 8. pq.add("apple");
- 9. System.out.println(pq.poll() + " " + pq.peek());
- 10. }
- 11. }

What is the result?

- A. apple pear
- B. banana pear
- C. apple apple
- D. apple banana
- E. banana banana

**Correct Answer:** D

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En este caso se adicionan 3 items a la cola "pq", en a linea 9 se invoca el metodo poll() el cual devuelve el primer item en orden natural osea "apple" y lo elomina, despues se invoca el metodo peek(); que retorna el priomer item en orden natural siendo "banana" pero no lo elimina.

**Queue**

La colección PriorityQueue ordena sus elementos utilizando una prioridad definida por el usuario. La prioridad puede ser el orden natural, pero también se puede ordenar utilizando un objeto Comparator. La interface Queue posee métodos que no se encuentran en ninguna otra colección:

peek(): Devuelve el primer elemento de la cola (o el que tenga mayor prioridad).

poll(): Devuelve el primer elemento de la cola (o el que tenga mayor prioridad), y luego lo elimina de la cola.

offer(T o): Agrega un elemento al final de la cola (o en el lugar donde indique su prioridad).

**PriorityQueue**

```
public class PriorityQueue<E>
extends AbstractQueue<E>
implements Serializable
```

An unbounded priority queue based on a priority heap. The elements of the priority queue are ordered according to their natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used. A priority queue does not permit null elements. A priority queue relying on natural ordering also does not permit insertion of non-comparable objects (doing so may result in ClassCastException).

**QUESTION 274**

Given:

```
11. public class Person {
12. private String name, comment;
13. private int age;
14. public Person(String n, int a, String c) {
15. name = n; age = a; comment = c;
16. }
17. public boolean equals(Object o) {
18. if (! (o instanceof Person)) return false;
19. Person p = (Person)o;
20. return age == p.age && name.equals(p.name);
21. }
```

22. }

What is the appropriate definition of the hashCode method in class Person?

- A. return super.hashCode();
- B. return name.hashCode() + age \* 7;
- C. return name.hashCode() + comment.hashCode() / 2;
- D. return name.hashCode() + comment.hashCode() / 2 - age \* 3;

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

En realidad lo que la pregunta quiere decir es como utilizar una apropiada definición o uso del método hashCode que es propio de todos los objetos para generar para la clase un objeto unico.

Como el hashCode podría ser igual si se ingresaran dos valores iguales para el String la forma de generar un único código es usando la combinación de el hashCode de la variable age y el de name o el de comment.

#### QUESTION 275

A programmer must create a generic class MinMax and the type parameter of MinMax must implement Comparable. Which implementation of MinMax will compile?

- A. 

```
class MinMax<E extends Comparable<E>> {
 E min = null;
 E max = null;
 public MinMax() {}
 public void put(E value) { /* store min or max */ }
```
- B. 

```
class MinMax<E implements Comparable<E>> {
 E min = null;
 E max = null;
 public MinMax() {}
 public void put(E value) { /* store min or max */ }
```
- C. 

```
class MinMax<E extends Comparable<E>> {
 <E> E min = null;
 <E> E max = null;
 public MinMax() {}
 public <E> void put(E value) { /* store min or max */ }
```
- D. 

```
class MinMax<E implements Comparable<E>> {
 <E> E min = null;
 <E> E max = null;
 public MinMax() {}
 public <E> void put(E value) { /* store min or max */ }
```

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**B y D** son incorrectos ya que no se puede usar la palabra implements en un generico.

**C** es incorrecto porque el uso de <E>E es inválido.

#### QUESTION 276

Given:

- 3. `import java.util.*;`
- 4. `public class G1 {`
- 5. `public void takeList(List<? extends String> list) {`

```

6. // insert code here
7. }
8. }

```

Which three code fragments, inserted independently at line 6, will compile? (Choose three.)

- A. list.add("foo");
- B. Object o = list;
- C. String s = list.get(0);
- D. list = new ArrayList<String>();
- E. list = new ArrayList<Object>();

**Correct Answer:** BCD

**Section:** (none)

**Explanation**

**Explanation/Reference:**

El método genérico takeList solo acepta listas de tipo String. por ello **E** es incorrecto.  
**A** es incorrecto .add no se puede usar porque la lista no se ha inicializado.

#### QUESTION 277

Given:

```

1. public class Drink implements Comparable {
2. public String name;
3. public int compareTo(Object o) {
4. return 0;
5. }
6. }

```

and:

```

20. Drink one = new Drink();
21. Drink two = new Drink();
22. one.name= "Coffee";
23. two.name= "Tea";
24. TreeSet set = new TreeSet();
25. set.add(one);
26. set.add(two);

```

A programmer iterates over the TreeSet and prints the name of each Drink object. What is the result?

- A. Tea
- B. Coffee
- C. Coffee  
Tea
- D. Compilation fails.
- E. The code runs with no output.

F. An exception is thrown at runtime.

**Correct Answer:** B

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Un TreeSet es un tipo de Set que no acepta objetos repetidos y los ordena naturalmente. Si una clase implementa el método compareTo() este retorna tres valores 1,-1 y 0 que quieren decir mayor que menor que e igual.

Como siempre el método compareTo en el ejemplo anterior retorna "0" le dice al compilador que todos los objetos que se crean son iguales pero como el TreeSet no acepta repetidos solo adiciona el primer objeto cuya variable name es "Coffe".

#### QUESTION 278

Which two scenarios are NOT safe to replace a StringBuffer object with a StringBuilder object? (Choose two.)

- A. When using versions of Java technology earlier than 5.0.
- B. When sharing a StringBuffer among multiple threads.
- C. When using the java.io class StringBufferInputStream.
- D. When you plan to reuse the StringBuffer to build more than one string.

**Correct Answer:** AB

**Section:** (none)

**Explanation**

**Explanation/Reference:**

StringBuilder se incorporó en la version 5 por ello no es bueno el remplazo en versiones mas antiguas.

No se debe remplazar StringBuffer por StringBuilder cuando se trabajan hilos pues StringBuilder no soporta ser sincronizado.

#### QUESTION 279

Given:

1. public class LineUp {
2.     public static void main(String[] args) {
3.         double d = 12.345;
4.         // insert code here
5.     }
6. }

Which code fragment, inserted at line 4, produces the output | 12.345|?

- A. System.out.printf("|%7d| \n", d);
- B. System.out.printf("|%7f| \n", d);
- C. System.out.printf("|%3.7d| \n", d);
- D. System.out.printf("|%3.7f| \n", d);
- E. System.out.printf("|%7.3d| \n", d);
- F. System.out.printf("|%7.3f| \n", d);

**Correct Answer:** F

**Section:** (none)

**Explanation**

**Explanation/Reference:**

**A, C y E** son invalidos pues el formateo se hace con la opción "d" que significa numero pero no es valido para tipos double.  
**B y D** tienen la sintaxis correcta pero nos retornarán 7 decimales.

#### QUESTION 280

Given that the current directory is empty, and that the user has read and write privileges to the current directory, and the following:

```
1. import java.io.*;
2. public class Maker {
3. public static void main(String[] args) {
4. File dir = new File("dir");
5. File f = new File(dir, "f");
6. }
7. }
```

Which statement is true?

- A. Compilation fails.
- B. Nothing is added to the file system.
- C. Only a new file is created on the file system.
- D. Only a new directory is created on the file system.
- E. Both a new file and a new directory are created on the file system.

**Correct Answer: B**

**Section: (none)**

**Explanation**

#### Explanation/Reference:

Para crear un directorio es necesario despues de instanciar el objeto "File" , añadir la declaración `dir.mkdir()`; y para crear un archivo es necesario añadir la declaración `f.createNewFile()`.  
La respuesta correcta es la **B**.

#### QUESTION 281

Given:

- 1. `d` is a valid, non-null Date object
- 2. `df` is a valid, non-null DateFormat object set to the current locale What outputs the current locale's country name and the appropriate version of `d`'s date?

- A. `Locale loc = Locale.getLocale();`  
`System.out.println(loc.getDisplayCountry()`  
`+ " " + df.format(d));`
- B. `Locale loc = Locale.getDefault();`  
`System.out.println(loc.getDisplayCountry()`  
`+ " " + df.format(d));`
- C. `Locale loc = Locale.getLocale();`  
`System.out.println(loc.getDisplayCountry()`  
`+ " " + df.setDateFormat(d));`
- D. `Locale loc = Locale.getDefault();`  
`System.out.println(loc.getDisplayCountry()`  
`+ " " + df.setDateFormat(d));`

**Correct Answer: B**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

**A** es incorrecto por que el método getLocale no existe para la clase Locale.

**C y D** son incorrectos el metodo setDateFormat() no existe para la clase "Date".

**QUESTION 282**

Given:

```
1. public class BuildStuff {
2. public static void main(String[] args) {
3. Boolean test = new Boolean(true);
4. Integer x = 343;
5. Integer y = new BuildStuff().go(test, x);
6. System.out.println(y);
7. }
8. int go(Boolean b, int i) {
9. if(b) return (i/7);
10. return (i/49);
11. }
12. }
```

What is the result?

- A. 7
- B. 49
- C. 343
- D. Compilation fails.
- E. An exception is thrown at runtime.

**Correct Answer: B**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

En este caso la linea 5 se iguala la variable "y" al valor resultante de la invocación al método BuilSttuff().go (Boolean b, int i)

El resultado se evalua en la línea 9 dando 49 ya que la variable local b es "true" y nunca llega al return de la linea 10.

**QUESTION 283**

DRAG DROP

Click the Task button.

Given: `NumberNames nn = new NumberNames();`  
`nn.put("one", 1);`  
`System.out.println(nn.getNames());`

Place the code into position to create a class that maps from Strings to integer values.  
 The result of execution must be [one]. Some options may be used more than once.

```
public class NumberNames {
 private HashMap<Place here , Place here > map =
 new HashMap<Place here , Place here Place here :
 public void put(String name, int value) {
 map.put(Place here , Place here);
 }
 public Place here getNames() {
 return map.keySet();
 }
}
```

Code

|                     |                  |                     |     |
|---------------------|------------------|---------------------|-----|
| Set<int>            | Set<Integer>     | HashSet             |     |
| Set<Integer,String> | Set<int, String> | Set<String,Integer> |     |
| Set<String, int>    | Set<String>      | NumberNames         |     |
| String              | Integer          | int                 | >   |
| >()                 | name             | value               | map |

Done

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:



Given: `NumberNames nn = new NumberNames();`  
`nn.put("one", 1);`  
`System.out.println(nn.getNames());`

Place the code into position to create a class that maps from Strings to integer values.  
The result of execution must be [one]. Some options may be used more than once.

```
public class NumberNames {
 private HashMap<String, Integer> map =
 new HashMap<String, Integer> >{};
 public void put(String name, int value) {
 map.put(name, value);
 }
 public Set<String> getNames() {
 return map.keySet();
 }
}
```

Code

|                      |                  |                      |
|----------------------|------------------|----------------------|
| Set<int>             | Set<Integer>     | HashSet              |
| Set<Integer, String> | Set<int, String> | Set<String, Integer> |
| Set<String, int>     | Set<String>      | NumberNames          |
| String               | Integer          | int                  |
| >()                  | name             | value                |
|                      |                  | map                  |

Done

#### QUESTION 284

DRAG DROP

Click the Task button.

Replace two of the Modifiers that appear in the `Single` class to make the code compile.  
Note: Three modifiers will not be used and four modifiers in the code will remain unchanged.

Code

```
public class Single {
 private static Single instance;
 public static Single getInstance() {
 if (instance == null) instance = create();
 return instance;
 }
 private Single() { }
 protected Single create() { return new Single(); }
}

class SingleSub extends Single {
}
```

Modifiers

final  
protected  
private  
abstract  
static

Done

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation

Explanation/Reference:

Replace two of the Modifiers that appear in the `Single` class to make the code compile.  
 Note: Three modifiers will not be used and four modifiers in the code will remain unchanged.

#### Code

```
public class Single {
 private static Single instance;
 public static Single getInstance() {
 if (instance == null) instance = create();
 return instance;
 }
 protected Single() { }
 static Single create() { return new Single(); }
}

class SingleSub extends Single {
}
```

#### Modifiers

final  
protected  
private  
abstract  
static

Done

#### QUESTION 285

##### DRAG DROP

Click the Task button.

Place the Relations on their corresponding Implementation Structures.  
 Note: Not all Implementation Structures will be used.

#### Implementation Structures

```
class A {
 List b;
}
```

```
class A
extends B,C { }
```

```
class A { }
```

```
class A {
 B b; C c;
}
```

```
class A {
 B b;
}
```

```
class A
implements B,C
{ }
```

```
class A
extends B { }
```

Done

#### Relations

Car is a Vehicle  
and  
Car is a Collectable

Car has a  
SteeringWheel

Car has Wheels

Mini is a Car

Car is an Object

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Answer: A

Place the Relations on their corresponding Implementation Structures.  
Note: Not all Implementation Structures will be used.

| Implementation Structures  |                                                 | Relations                                       |
|----------------------------|-------------------------------------------------|-------------------------------------------------|
| Car has Wheels             | <pre>class A<br/>extends B,C { }</pre>          | Car is a Vehicle<br>and<br>Car is a Collectable |
| Car is an Object           | <pre>class A {<br/>  B b; C c;<br/>}</pre>      | Car has a<br>SteeringWheel                      |
| Car has a<br>SteeringWheel | Car is a Vehicle<br>and<br>Car is a Collectable | Car has Wheels                                  |
| Mini is a Car              |                                                 | Mini is a Car                                   |
|                            | Done                                            | Car is an Object                                |

**QUESTION 286**

DRAG DROP

Click the Task button.



**Drag and Drop**

Insert six modifiers into the code such that it meets all of these requirements:

1. It must be possible to create instances of Alpha and Beta from outside the packages in which they are defined.
2. When an object of type Alpha (or any potential subclass of Alpha) has been created, the instance variable alpha may never be changed.
3. The value of the instance variable alpha must always be "A" for objects of type Alpha.

**Code**

```
package alpha;
 Placeholder class Alpha {
 Placeholder String alpha;
 Placeholder Alpha() { this("A"); }
 Placeholder Alpha(String a) { alpha = a; }
 }

package beta;
 Placeholder class Beta extends alpha.Alpha {
 Placeholder Beta(String a) { super(a); }
 }
```

**Modifiers**

private  
protected  
public

Done

- A.  
B.  
C.  
D.

**Correct Answer: A**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

**Drag and Drop**

Insert six modifiers into the code such that it meets all of these requirements:

1. It must be possible to create instances of Alpha and Beta from outside the packages in which they are defined.
2. When an object of type Alpha (or any potential subclass of Alpha) has been created, the instance variable alpha may never be changed.
3. The value of the instance variable alpha must always be "A" for objects of type Alpha.

**Code**

```
package alpha;
 public class Alpha {
 private String alpha;
 public Alpha() { this("A"); }
 protected Alpha(String a) { alpha = a; }
 }

package beta;
 public class Beta extends alpha.Alpha {
 public Beta(String a) { super(a); }
 }
```

**Modifiers**

private  
protected  
public

Done

### QUESTION 287

DRAG DROP

Click the Task button.

Given:

```
1. import java.util.*;
2. public class TestGenericConversion {
3. public static void main(String[] args) {
4. List list = new LinkedList();
5. list.add("one");
6. list.add("two");
7. System.out.print(((String)list.get(0)).length());
8. }
9. }
```

Refactor this class to use generics without changing the code's behavior.

```
1. import java.util.*;
2. public class TestGenericConversion {
3. public static void main(String[] args) {
4. Place here
5. list.add("one");
6. list.add("two");
7. Place here
8. }
9. }
```

#### Code

|                                               |                                                           |
|-----------------------------------------------|-----------------------------------------------------------|
| List list = new LinkedList();                 | System.out.print( list.get(0).length() );                 |
| List<String> list = new LinkedList<String>(); | System.out.print( list.get<String>(0).length() );         |
| List<String> list = new LinkedList();         | System.out.print( <String>list.get(0).length() );         |
| List list = new LinkedList<String>();         | System.out.print( ((List<String>)list.get(0)).length() ); |

- A.
- B.
- C.
- D.

**Correct Answer:** A

**Section:** (none)

**Explanation**

**Explanation/Reference:**

Given:

```
1. import java.util.*;
2. public class TestGenericConversion {
3. public static void main(String[] args) {
4. List list = new LinkedList();
5. list.add("one");
6. list.add("two");
7. System.out.print(((String)list.get(0)).length());
8. }
9. }
```

Refactor this class to use generics without changing the code's behavior.

```
1. import java.util.*;
2. public class TestGenericConversion {
3. public static void main(String[] args) {
4. List<String> list = new LinkedList<String>();
5. list.add("one");
6. list.add("two");
7. System.out.print(list.get(0).length());
8. }
9. }
```

Code

|                                               |                                                         |
|-----------------------------------------------|---------------------------------------------------------|
| List list = new LinkedList();                 | System.out.print(list.get(0).length());                 |
| List<String> list = new LinkedList<String>(); | System.out.print(list.get<String>(0).length());         |
| List<String> list = new LinkedList();         | System.out.print(<String>list.get(0).length());         |
| List list = new LinkedList<String>();         | System.out.print(((List<String>)list.get(0)).length()); |

### QUESTION 288

DRAG DROP

Click the Task button.

Drag and Drop

Place each Collection Type on the statement to which it applies.

| Statements                                                      | Collection Types |
|-----------------------------------------------------------------|------------------|
| allows access to elements by their integer index                | java.util.Map    |
| defines the method V get(Object key)                            | java.util.Set    |
| is designed for holding elements prior to processing            | java.util.List   |
| contains no pair of elements e1 and e2, such that e1.equals(e2) | java.util.Queue  |

Done

- A.
- B.
- C.
- D.

Correct Answer: A

Section: (none)

Explanation



**Explanation/Reference:**

Drag and Drop

Place each Collection Type on the statement to which it applies.

| Statements                   | Collection Types             |
|------------------------------|------------------------------|
| <code>java.util.List</code>  | <code>java.util.Map</code>   |
| <code>java.util.Map</code>   | <code>java.util.Set</code>   |
| <code>java.util.Queue</code> | <code>java.util.List</code>  |
| <code>java.util.Set</code>   | <code>java.util.Queue</code> |

Done

**QUESTION 289**

**DRAG DROP**

Click the Task button.

Drag and Drop

Place the code fragments into position to produce the output:  
true true false

**Code**

```
Scanner scanner = new Scanner("One 5 true 3 true 6 7 false");
scanner.useDelimiter(",");
while (Place here) {
 if (Place here) {
 System.out.print(Place here + " ");
 } else Place here ;
}
```

**Code Fragments**

|                                       |                                    |
|---------------------------------------|------------------------------------|
| <code>scanner.hasNextBoolean()</code> | <code>scanner.nextBoolean()</code> |
| <code>scanner.next()</code>           | <code>scanner.hasNext()</code>     |

Done

- A.
- B.
- C.
- D.

**Correct Answer: A**

**Section: (none)**

**Explanation**

**Explanation/Reference:**

**Drag and Drop**

Place the code fragments into position to produce the output:  
true true false

**Code**

```
Scanner scanner = new Scanner("One 5 true 3 true 6 7 false");
scanner.useDelimiter(",");
while () {
 if () {
 System.out.print(+ " ");
 } else ;
}
```

**Code Fragments**

**QUESTION 290**

**DRAG DROP**

Click the Task button.

**Drag and Drop**

Chain these constructors to create objects to read from a file named "in" and to write to a file named "out"

```
reader = "in");
writer = "out");
```

**Constructors**

- A.
- B.
- C.
- D.

**Correct Answer: A**

**Section: (none)**

**Explanation**

**Explanation/Reference:**



Drag and Drop

Chain these constructors to create objects to read from a file named "in" and to write to a file named "out"

reader = );

writer = );

### Constructors

Done