¿Enviar la respuesta?

Solo puedes responder una vez este formulario. ¿Quieres continuar?

CAMBIAR DE CUENTA ENVIAR

2do. Parcial JAVA - FRRe B

Responda las respuestas y presione "enviar". Sólo se aceptará un sólo envío. Sólo se aceptarán los emails registrados en el CV-FRRe. No se admitirá CODIGO REPETIDO o ADAPTADO entre Alumnos. En el análisis de código, asuma que están importadas todas las dependencias necesarias. Escriba código que no dependa de funcionalidades ("inteligencia") de su IDE de preferencia, y funcione en cualquier Entorno (en particular BlueJ) .

joseluisvallejos1995@gmail.com Cambiar de cuenta

*Obligatorio

Correo *

joseluisvallejos1995@gmail.com

Dada la ***Clase*** en código adjunto en imagen, se pide (con libre criterio) : 25 puntos
- Sobreescriba el método toString().
- Sobreescriba el método equals(), asumiendo como "Criterio de comparación" la raza.
- "Normalizando" a minúsculas, ordene en forma alfabética por raza en forma descendente.

```
public class Gato implements Comparable<Gato> {
    private String nombre;
    private String color;
    private String raza;

    public Gato(String nombre, String color, String raza) {
        this.nombre = nombre;
        this.color = color;
        this.raza = raza;
    }

    public String getNombre() {
        return nombre;
    }

    public String getRaza() {
        return raza;
    }
```

```
import java.util.ArrayList;

public class Gato implements Comparable<Gato> {

    private String nombre;
    private String color;
    private String raza;

    //constructor
```

```
//constructor
public Gato(String nombre, String color, String raza) {
    this.nombre = nombre;
    this.color = color;
    this.raza = raza;
}

//getter
public String getNombre() {
    return nombre;
}

public String getRaza() {
    return raza;
}

//sobreescribimos el metodo toString
@Override
public String toString() {
    return "Gato{" + "nombre=" + nombre + ", color=" + color + ", raza=" + raza + '}';
}

//sobreescribimos el metodo equals
@Override
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Gato other = (Gato) obj;
    if ((this.nombre == null) ? (other.nombre != null) : !this.nombre.equals(other.nombre)) {
        return false;
    }
    if ((this.color == null) ? (other.color != null) : !this.color.equals(other.color)) {
        return false;
    }
    if ((this.raza == null) ? (other.raza != null) : !this.raza.equals(other.raza)) {
        return false;
    }
    return true;
}

//Normalizando a minusculas
public String normalizar(String cadena) {
    return cadena.toLowerCase();
}

@Override
public int compareTo(Gato o) {
    return 0;
}
```

```
    public static void main(String[] args) {

        //creamos un arraylist de gatos
        ArrayList<Gato> gatos = new ArrayList<Gato>();

        //creamos los gatos
        Gato gato1 = new Gato("Naranja", "naranja", "persa");
        Gato gato2 = new Gato("Gris", "gris", "siames");
        Gato gato3 = new Gato("Blanco", "blanco", "angora");

        //añadimos los gatos al arraylist
        gatos.add(gato1);
        gatos.add(gato2);
        gatos.add(gato3);

        //mostramos los gatos en minusculas y ordenados alfabeticamente por raza en forma
descendente
        for (Gato gato : gatos) {
            System.out.println(gato.getNombre().toLowerCase() + " " +
gato.getRaza().toLowerCase());
        }
    }

}
```

Escriba (con libre criterio) un programa JAVA que cree una Colección de Objetos numéricos ingresados por teclado, y luego la muestre por pantalla ordenada en forma descendente. 20 puntos

```
//scanner
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class ej8 {

    public static void main(String[] args) {
        int numero;
        List<Integer> lista = new ArrayList<Integer>();

        Scanner scanner = new Scanner(System.in);

        for (int i = 0; i < 10; i++) {
            System.out.println("Introduce un numero: ");
            numero = scanner.nextInt();
            lista.add(numero);

        }
        scanner.close();

        //Ordenar lista de mayor a menor
        for (int i = 0; i < lista.size(); i++) {
            for (int j = 0; j < lista.size(); j++) {
                if (lista.get(i) > lista.get(j)) {
                    int aux = lista.get(i);
                    lista.set(i, lista.get(j));
                    lista.set(j, aux);
                }
            }
        }
```

```
        System.out.println("\n\n\nLista ordenada de mayor a menor:\n\n\n");

        //Mostrar lista
        for (int i = 0; i < lista.size(); i++) {
            System.out.println(lista.get(i));
        }
    }


}
```

Seleccione según imagen adjunta, la opción CORRECTA. 5 puntos

Pregunta: En la definición de una interface en Java :

a. Es necesaria emplear la palabra clave abstract.
b. La signatura de los métodos de una interfaz tienen visibilidad public o private, pero no protected.
c. No se permiten campos constantes.
d. Aunque no se indique usando la palabra clave final, todos los campos son tratados como si así fuesen.

- [ ] a
- [ ] b
- [ ] c
- [x] d

Dado el código adjunto, cual sería la salida por Consola? 5 puntos

```
2. public class Tolt {
3.   public static void checkIt(int a) {
4.     if(a == 1)  throw new IllegalArgumentException();
5.   }
6.   public static void main(String[] args) {
7.     for(int x=0; x<2; x++)
8.       try {
9.         System.out.print("t ");
10.        checkIt(x);
11.        System.out.print("t2 ");
12.      }
13.      finally { System.out.print("f "); }
14. } }
```

- [x] Compilación falla.

- [ ] "t t2 f t "
- [ ] "t t2 f t f "
- [ ] "t t2 f t t2 f "
- [ ] "t t2 f t ", seguido de una Excepción no controlada.
- [ ] "t t2 f t f ", seguido de una Excepción no controlada.
- [ ] "t t2 f t t2 f ", seguido de una Excepción no controlada.

Dado el código en imagen, cual sería la salida por Consola? 5 puntos

```
 2. public class Checkout2 implements Runnable {
 3.   void doStuff() { }
 4.   synchronized void doSynch() {
 5.     try { Thread.sleep(1000); }
 6.     catch (Exception e) { System.out.print("e "); }
 7.   }
 8.   public static void main(String[] args) {
 9.     long start = System.currentTimeMillis();
10.     new Thread(new Checkout2()).start();
11.     Thread t1 = new Thread(new Checkout2());
12.     t1.start();
13.     try { t1.join(); }
14.     catch (Exception e) { System.out.print("e "); }
15.     System.out.println("elapsed: "
                          + (System.currentTimeMillis() - start));
16.   }
17.   public void run() {
18.     for(int j = 0; j < 4; j++) {
19.       doStuff();
20.       try { Thread.sleep(1000); }
21.       catch (Exception e) { System.out.print("e "); }
22.       doSynch();
23. } } }
```

- [ ] Compilación falla.
- [x] El tiempo transcurrido sería de alrededor de 8 segundos.
- [ ] El tiempo transcurrido sería de alrededor de 9 segundos.
- [ ] El tiempo transcurrido sería de alrededor de 12 segundos.
- [ ] Ninguna de las anteriores.

Dado el código en imagen, cual sería el resultado? 2 puntos

```
2. public class Maize {
3.   public static void main(String[] args) {
4.     String s = "12";
5.     s.concat("ab");
6.     s = go(s);
7.     System.out.println(s);
8.   }
9.   static String go(String s) {
10.    s.concat("56");
11.    return s;
12. } }
```

- [ ] ab
- [x] 12
- [ ] ab56
- [ ] 12ab
- [ ] 1256
- [ ] 12ab56
- [ ] Compilación falla.

Dado el código en imagen adjunta, cual sería el resultado? 5 puntos

```
2. public class Tshirt extends Thread {
3.   public static void main(String[] args) {
4.     System.out.print(Thread.currentThread().getId() + " ");
5.     Thread t1 = new Thread(new Tshirt());
6.     Thread t2 = new Thread(new Tshirt());
7.     t1.start();
8.     t2.run();
9.   }
10.   public void run() {
11.     for(int i = 0; i < 2; i++)
12.       System.out.print(Thread.currentThread().getId() + " ");
13. } }
```

- [x] Ninguna de las sgtes.
- [ ] 1 1 9 9 1
- [ ] 1 2 9 9 2
- [ ] 1 9 9 9 9
- [ ] Se lanza una Excepción durante la ejecución.
- [ ] Compilación falla debido a un error en línea 4.
- [ ] Compilación falla debido a un error en línea 8.

Dado el código adjunto, cual sería el resultado? 5 puntos

```
3. public static void main(String[] args) {
4.   try {
5.     throw new Error();
6.   }
7.   catch (Error e) {
8.     try { throw new RuntimeException(); }
9.     catch (Throwable t) { }
10.  }
11.  System.out.println("phew");
12. }
```

- [x] "phew"
- [ ] No se genera salida.
- [ ] Compilación falla en línea 5.
- [ ] Compilación falla en línea 7.
- [ ] Compilación falla en línea 8.
- [ ] Compilación falla en línea 9.

Dada la imagen adjunta, seleccione la afirmación INCORRECTA. 5 puntos

a. Únicamente las clases que implementan la interfaz List permiten el uso de iteradores.
b. Un iterador es un objeto que proporciona funcionalidad para recorrer todos los elementos de una colección.
c. Un iterador permite recorrer cualquier tipo de colección hacia adelante utilizando el método next() combinado con el método hasNext() para comprobar si se ha alcanzado el final de la colección.
d. Una colección puede recorrerse tanto con un iterador como con un ciclo for-each. Ambas formas son equivalentes.

- [x] a

- [ ] b
- [ ] c
- [ ] d

Dado el código main() en imagen adjunta, seleccione el fragmento de código que insertado en línea 8 , produciría la salida por consola "2-4-7-" 5 puntos

```
1. import java.util.*;
2. public class Ps {
3.   public static void main(String[] args) {
4.     PriorityQueue<String> pq = new PriorityQueue<String>();
5.     pq.add("4");
6.     pq.add("7");
7.     pq.add("2");
8.     // insert code here
9. } }
```

A.
```
Iterator it2 = pq.iterator();
while(it2.hasNext()) System.out.print(it2.next() + "-");
System.out.println();
```

- [ ] a

B.
```
Arrays.sort(pq.toArray());
Iterator it3 = pq.iterator();
while(it3.hasNext()) System.out.print(it3.next() + "-");
System.out.println();
```

- [x] b

C.
```
Object[] pqa = pq.toArray();
Arrays.sort(pqa);
for(Object o: pqa) System.out.print(o + "-");
System.out.println();
```

- [ ] c

Dado el código en imagen adjunta, cual sería el resultado por Consola? 5 puntos

```
2. class Noodle {
3.   String name;
4.   Noodle(String n) { name = n; }
5. }
6. class AsianNoodle extends Noodle {
7.   public boolean equals(Object o) {
8.     AsianNoodle n = (AsianNoodle)o;
9.     if(name.equals(n.name))  return true;
10.    return false;
11.  }
12.  public int hashCode() { return name.length(); }
13.  AsianNoodle(String s) { super(s); }
14. }
15. public class Soba extends AsianNoodle {
16.   public static void main(String[] args) {
17.     Noodle n1 = new Noodle("bob");  Noodle n2 = new Noodle("bob");
18.     AsianNoodle a1 = new AsianNoodle("fred");
19.     AsianNoodle a2 = new AsianNoodle("fred");
20.     Soba s1 = new Soba("jill");  Soba s2 = new Soba("jill");
21.     System.out.print(n1.equals(n2) + " " + (n1 == n2) + " | ");
22.     System.out.print(a1.equals(a2) + " " + (a1 == a2) + " | ");
23.     System.out.println(s1.equals(s2) + " " + (s1 == s2));
24.   }
25.   Soba(String s) { super(s); }
26. }
```

- [ ] Compilación falla.
- [ ] tru e true | true true | true true
- [ ] true false | true false | true false
- [ ] false false | true false | true false
- [x] false false | true false | false false
- [ ] false false | false false | false false

Escriba (con libre criterio) un programa JAVA que diseñe una Clase que cumplimente el Patrón JAVABean, con 2 atributos, y sobrescriba (tambien con libre criterio) los métodos toString(), equals() y hashCode(). 13 puntos

```
/*

 */

public class ej7 {

    private String name;
    private int age;

    // Constructor
    public ej7(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }

    // constructor
    public ej7() {
        this.name = "John Doe";
        this.age = 0;
    }

    // getter
    public String getName() {
        return name;
    }

    // getter
    public int getAge() {
        return age;
    }
```

```
    // setter
    public void setName(String name) {
        this.name = name;
    }

    // setter
    public void setAge(int age) {
        this.age = age;
    }

    // toString
    @Override
    public String toString() {
        return "ej7{" + "name=" + name + ", age=" + age + '}';
    }

    //equals
    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final ej7 other = (ej7) obj;
        if ((this.name == null) ? (other.name != null) : !this.name.equals(other.name)) {
            return false;
        }
        if (this.age != other.age) {
            return false;
        }
        return true;
    }

    //hashCode
    @Override
    public int hashCode() {
        int hash = 7;
        hash = 97 * hash + (this.name != null ? this.name.hashCode() : 0);
        hash = 97 * hash + this.age;
        return hash;
    }

}
```