Navegación por el cuestionario

1 2 3 4 5 6 7 8 9 10

Finalizar revisión

Comenzado el	jueves, 7 de marzo de 2019, 20:38
Estado	Finalizado
Finalizado en	viernes, 8 de marzo de 2019, 16:27
Tiempo empleado	19 horas 48 minutos
Calificación	**8,57** de 10,00 (**86**%)

Pregunta **1**
Correcta
Puntúa 1,00 sobre 1,00
Marcar pregunta

¿Cuál de las siguientes afirmaciones es correcta?

Seleccione una:
- a. Una clase no puede implementar más de dos `interfaces`.
- b. Una clase abstracta puede incluir implementaciones de métodos y una `interface` no. ✓
- c. Una `interface` puede contener varias declaraciones de métodos (sin implementar) y una clase abstracta no.
- d. Una clase puede heredar el comportamiento de dos clases abstractas disjuntas (que no heredan una de la otra).

La respuesta correcta es: Una clase abstracta puede incluir implementaciones de métodos y una `interface` no.

Correcta
Puntos para este envío: 1,00/1,00.

Pregunta **2**
Correcta
Puntúa 1,00 sobre 1,00
Marcar pregunta

¿Con qué nombre son conocidas aquellas clases cuya única función es la de ser superclase en una jerarquía, sin que llegue a haber nunca instancias de ellas?

Seleccione una:
- a. Clases jerárquicas.
- b. Clases básicas.
- c. Ese tipo de clases no tienen sentido y no existen en Java.
- d. Clases abstractas. ✓

La respuesta correcta es: Clases abstractas.

Correcta
Puntos para este envío: 1,00/1,00.

Pregunta **3**
Correcta
Puntúa 1,00 sobre 1,00
Marcar pregunta

Un método abstracto no puede ser estático. ¿Verdadero o falso?

Seleccione una:
- Verdadero ✓
- Falso

La respuesta correcta es 'Verdadero'

Correcta
Puntos para este envío: 1,00/1,00.

Pregunta **4**
Correcta
Puntúa 1,00 sobre 1,00
Marcar pregunta

Los modificadores de clase `final` y `abstract` son excluyentes. ¿Verdadero o falso?

Seleccione una:
- Verdadero ✓
- Falso

La respuesta correcta es 'Verdadero'

Correcta
Puntos para este envío: 1,00/1,00.

Pregunta **5**
Correcta
Puntúa 0,67 sobre 1,00
Marcar pregunta

Tenemos la clase Triangulo:

```
public class Triangulo {
    private String id;
    private double base;
    private double altura;

    public Triangulo(String id, double base, double altura) {
        this.id = id;
        this.base = base;
        this.altura = altura;
    }

    public String toString() {
        return "Triangulo(" + id + "): " + base + " x " + altura;
    }

    public double getArea() {
        return base * altura;
    }

    public double getBase() {
        return base;
    }
}
```

Ordena el código de la clase ComparadorBase para que implemente correctamente la interfaz Comparator:

```
import java.util.*;

public class ComparadorBase implements Comparator<Triangulo> {
    public int compare(Triangulo triangulo1, Triangulo triangulo2) {
        double base1 = triangulo1.getBase();
        double base2 = triangulo2.getBase();
        if (base1 == base2)
            return triangulo1.getId().compareTo(triangulo2.getId());
        if (base1 > base2)
            return -1;
        return 1;
    }
}
```

Cuando se ejecute el siguiente código el array se ordene según la base, de mayor a menor, y si las bases son iguales en función del id de los tríangulos, de la A a la Z:

```
Triangulo[] arrayTriangulo = {new Triangulo("A321", 4, 3), new Triangulo("C321", 4, 2), new Triangulo("A123", 2, 5), new Triangulo("B123", 3, 4)};
```

```
System.out.println(Arrays.toString(arrayTriangulo));
Arrays.sort(arrayTriangulo, new ComparadorBase());
System.out.println(Arrays.toString(arrayTriangulo));
```

El resultado será:

```
[Triangulo(A321): 4.0 x 3.0, Triangulo(C321): 4.0 x 2.0, Triangulo(A123): 2.0 x 5.0, Triangulo(B123): 3.0 x 4.0]
[Triangulo(A321): 4.0 x 3.0, Triangulo(C321): 4.0 x 2.0, Triangulo(B123): 3.0 x 4.0, Triangulo(A123): 2.0 x 5.0]
```

Respuesta correcta

La respuesta correcta es:
Tenemos la clase Triangulo :

```
public class Triangulo {
    private String id;
    private double base;
    private double altura;

    public Triangulo(String id, double base, double altura) {
        this.id = id;
        this.base = base;
        this.altura = altura;
    }

    public String toString() {
        return "Triangulo(" + id + "): " + base + " x " + altura;
    }

    public double getArea() {
        return base * altura;
    }

    public double getBase() {
        return base;
    }
}
```

Ordena el código de la clase ComparadorBase para que implemente correctamente la interfaz Comparator:

```
import java.util.*;

[public class ComparadorBase implements Comparator<Triangulo> {]
   [public int compare(Triangulo triangulo1, Triangulo triangulo2) {]
      [double base1 = triangulo1.getBase();]
      [double base2 = triangulo2.getBase();]
      [if (base1 == base2)]
         [return triangulo1.getId().compareTo(triangulo2.getId());]
      [if (base1 > base2)]
         [return -1;]
      [return 1;]
}
```

Cuando se ejecute el siguiente código el array se ordene según la base, de mayor a menor, y si las bases son iguales en función del id de los tríangulos, de la A a la Z:

```
Triangulo[] arrayTriangulo = {new Triangulo("A321", 4, 3), new Triangulo("C321", 4, 2), new Triangulo("A123", 2, 5), new Triangulo("B123", 3, 4)};

System.out.println(Arrays.toString(arrayTriangulo));
Arrays.sort(arrayTriangulo, new ComparadorBase());
System.out.println(Arrays.toString(arrayTriangulo));
```

El resultado será:

```
[Triangulo(A321): 4.0 x 3.0, Triangulo(C321): 4.0 x 2.0, Triangulo(A123): 2.0 x 5.0, Triangulo(B123): 3.0 x 4.0]
[Triangulo(A321): 4.0 x 3.0, Triangulo(C321): 4.0 x 2.0, Triangulo(B123): 3.0 x 4.0, Triangulo(A123): 2.0 x 5.0]
```

Correcta

Puntos para este envío: 1,00/1,00. Contando con los intentos anteriores, daría **0,67/1,00**.

Pregunta **6**

Correcta

Puntúa 1,00 sobre 1,00

Marcar pregunta

Tenemos la clase Triangulo :

```
public class Triangulo {
    private String id;
    private double base;
    private double altura;

    public Triangulo(String id, double base, double altura) {
        this.id = id;
        this.base = base;
        this.altura = altura;
    }

    public String toString() {
        return "Triangulo(" + id + "): " + base + " x " + altura;
    }

    public double getArea() {
        return base * altura;
    }
}
```

Corríge la siguiente versión para que implemente correctamente la interfaz Comparable y cuando se ejecute el siguiente código el array se ordene según el id de los tríangulos, de la Z a la A:

```
Triangulo[] arrayTriangulo = {new Triangulo("A321", 4, 3), new Triangulo("C321", 5, 2), new Triangulo("A123", 2, 5), new Triangulo("B123", 3, 4)};

System.out.println(Arrays.toString(arrayTriangulo));
Arrays.sort(arrayTriangulo);
System.out.println(Arrays.toString(arrayTriangulo));
```

El resultado será:

```
[Triangulo(A321): 4.0 x 3.0, Triangulo(C321): 5.0 x 2.0, Triangulo(A123): 2.0 x 5.0, Triangulo(B123): 3.0 x 4.0]
[Triangulo(C321): 5.0 x 2.0, Triangulo(B123): 3.0 x 4.0, Triangulo(A321): 4.0 x 3.0, Triangulo(A123): 2.0 x 5.0]
```

Respuesta: (penalty regime: 10, 20, ... %)

Reiniciar respuesta

```
4     private double altura;
5
6     public Triangulo(String id, double base, double altura) {
7         this.id = id;
8         this.base = base;
9         this.altura = altura;
10    }
11
12    public String toString() {
13        return "Triangulo(" + id + "): " + base + " x " + altura;
14    }
15
16    public double getArea() {
17        return base * altura;
18    }
```

```
19    @Override
20    public int compareTo(Triangulo unTriangulo) {
21        return unTriangulo.id.compareTo(id);
22    }
23 }
```

	Test

✓	`Triangulo[] arrayTriangulo = {new Triangulo("A321", 4, 3), new Triangulo("C321", 5, 2), new Triangulo("A123", 2, 5), new Triangulo("B123", 3, 4`
	`System.out.println(java.util.Arrays.toString(arrayTriangulo));`
	`java.util.Arrays.sort(arrayTriangulo);`
	`System.out.println(java.util.Arrays.toString(arrayTriangulo));`

✓	`Triangulo[] arrayTriangulo = {new Triangulo("B321", 2, 3), new Triangulo("C321", 5, 2), new Triangulo("A123", 3, 5), new Triangulo("D123", 6, 4`
	`System.out.println(java.util.Arrays.toString(arrayTriangulo));`
	`java.util.Arrays.sort(arrayTriangulo);`
	`System.out.println(java.util.Arrays.toString(arrayTriangulo));`

Todas las pruebas superadas. ✓

Question author's solution:

```
public class Triangulo implements Comparable<Triangulo> {
    private String id;
    private double base;
    private double altura;

    public Triangulo(String id, double base, double altura) {
        this.id = id;
        this.base = base;
        this.altura = altura;
    }

    public String toString() {
        return "Triangulo(" + id + "): " + base + " x " + altura;
    }

    public double getArea() {
        return base * altura;
    }

    public int compareTo(Triangulo unTriangulo) {
        return -id.compareTo(unTriangulo.id);
    }
}
```

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **7**

Incorrecta

Puntúa 0,00 sobre 1,00

Marcar pregunta

Tenemos la clase abstracta Empleado :

```
public abstract class Empleado {
    private String nombre;
    private String numSeguridadSocial;

    public Empleado(String nombre, String numero) {
        this.nombre = nombre;
        numSeguridadSocial = numero;
    }

    abstract double salario();

    public String toString() {
        return nombre + " (" + numSeguridadSocial + ")";
    }
}
```

Corrige la clase EmpleadoAsalariado, creada a partir de la clase Empleado, para que cumpla las siguientes características:

```
1. Tendrá un atributo más de tipo doble denominado salario. Se deberá inicializar en el constructor a partir del valor indicado como parámetro.
2. El método salario devolverá el valor del atributo salario.
3. El método toString devolverá el siguiente texto: nombre (numSeguridadSocial) Salario: salario. El salario se mostrará con 2 decimales.
```

Si ejecutamos el siguiente código :

```
EmpleadoAsalariado sara = new EmpleadoAsalariado("Sara Alkorta", "111111JJ", 25000);
System.out.println(sara);
System.out.println(sara.salario());
```

El resultado será:

```
Sara Alkorta (111111JJ) Salario: 25000,00
25000.0
```

Recuerda interactuar con el código de la superclase mediante la palabra reservada super.

Respuesta: (penalty regime: 10, 20, ... %)

Reiniciar respuesta

```
1  public class EmpleadoAsalariado extend Empleado {
2     private int salario;
3
4     public EmpleadoAsalariado(String nombre, String numero, double salario) {
5         Empleado(nombre, numero);
6         this.salario = salario;
7     }
8
9     public double salario() {
10        return salario;
11    }
12
13    public String toString(){
14        return super.toString() + " Salario: " + salario();
15    }
16 }
```

Syntax Error(s)

```
__Tester__.java:1: error: '{' expected
class EmpleadoAsalariado extend Empleado {
                                ^
1 error
```

Question author's solution:

```
public class EmpleadoAsalariado extends Empleado {
   private double salario;

 public EmpleadoAsalariado(String nombre, String numero, double salario) {
   super(nombre, numero);
      this.salario = salario;
        }

 public double salario() {
   return salario;
        }
```

```
 public String toString(){
  return super.toString() + String.format(" Salario: %.2f", salario());
       }
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Pregunta **8**

Correcta

Puntúa 1,00 sobre 1,00

Marcar pregunta

Tenemos la clase abstracta Ave :

```
public abstract class Ave {
    protected char sexo;
    protected int edad;

    Ave(char sexo, int edad){
        this.sexo = sexo;
        this.edad = edad;
    }

    public String toString(){
        return "Sexo: " + sexo + " Edad: " + edad;
    }

   public String trinar() {
      return "Piopio";
   }
}
```

Crea la clase Loro a partir de la clase Ave para que cumpla las siguientes características:

```
1. Tendrá un atributo más de tipo String denominado region. Se deberá inicializar en el constructor a partir del valor indicado como parámetro.
2. El método trinar devolverá el siguiente texto: "HOLA Piopio HOLA"
3. Tendrá el método deDondeEs. No necesita parámetros y devolverá un String indicando la región de la que es el loro:
    - Si region es igual a "N" la región será "Norte"
    - Si region es igual a "S" la región será "Sur"
    - Si region es igual a "E" la región será "Este"
    - Si region es igual a "O" la región será "Oeste"
    - Para el resto de valores devolverá "Desconocido"
```

Si ejecutamos el siguiente código :

```
Loro unLoro = new Loro('F', 10, "S");
System.out.println("LORO");
System.out.println(unLoro);
System.out.println(unLoro.trinar());
System.out.println("Region: " + unLoro.deDondeEs());
```

El resultado será:

```
LORO
Sexo: F Edad: 10
HOLA Piopio HOLA
Region: Sur
```

Respuesta: (penalty regime: 10, 20, ... %)

```
1  public class Loro extends Ave {
2     String region;
3
4     public Loro(char sexo, int edad, String region) {
5        super(sexo, edad);
6        this.region = region;
7     }
8     public String trinar() {
9        return "HOLA Piopio HOLA";
10    }
11    public String deDondeEs() {
12       region = region.toUpperCase();
13       switch(region) {
14          case "N":
15             return "Norte";
16
17          case "S":
18             return "Sur";
19
20          case "E":
```

	Test	Expected	Got	

✓	`Loro unLoro = new Loro('F', 10, "S");`
	`System.out.println(unLoro);`
	`System.out.println(unLoro.trinar());`
	`System.out.println("Region: " + unLoro.deDondeEs());`
		Sexo: F Edad: 10
		HOLA Piopio HOLA
		Region: Sur
			Sexo: F Edad: 10
			HOLA Piopio HOLA
			Region: Sur
				✓

✓	`Loro unLoro = new Loro('M', 2, "F");`
	`System.out.println(unLoro);`
	`System.out.println(unLoro.trinar());`
	`System.out.println("Region: " + unLoro.deDondeEs());`
		Sexo: M Edad: 2
		HOLA Piopio HOLA
		Region: Desconocido
			Sexo: M Edad: 2
			HOLA Piopio HOLA
			Region: Desconocido
				✓

✓	`Loro unLoro = new Loro('M', 4, "O");`
	`System.out.println(unLoro);`
	`System.out.println(unLoro.trinar());`
	`System.out.println("Region: " + unLoro.deDondeEs());`
		Sexo: M Edad: 4
		HOLA Piopio HOLA
		Region: Oeste
			Sexo: M Edad: 4
			HOLA Piopio HOLA
			Region: Oeste
				✓

Todas las pruebas superadas. ✓

Question author's solution:

```
public class Loro extends Ave {
 private String region;

 Loro(char sexo, int edad, String region) {
   super(sexo, edad);
   this.region = region;
       }

 public String deDondeEs() {
     switch (region) {
         case "N":
           return "Norte";
         case "E":
           return "Este";
        case "S":
           return "Sur";
        case "O":
           return "Oeste";
        default:
           return "Desconocido";
     }
       }

   public String trinar() {
      return "HOLA " + super.trinar() + " HOLA";
   }
}
```

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **9**

Correcta

Puntúa 1,00 sobre 1,00

Tenemos la clase abstracta Ave :

```
public abstract class Ave {
    protected char sexo;
    protected int edad;
```

Marcar pregunta

```
    Ave(char sexo, int edad){
        this.sexo = sexo;
        this.edad = edad;
    }

    public String toString(){
        return "Sexo: " + sexo + " Edad: " + edad;
    }

   public String trinar() {
      return "Piopio";
   }
}
```

Crea la clase Canario a partir de la clase Ave para que cumpla las siguientes características:

```
1. Tendrá un atributo más de tipo entero denominado size. Se deberá inicializar en el constructor a partir del valor indicado como parámetro.
2. El método trinar mostrará por consola el siguiente texto: "Piopio - Piopio"
3. Tendrá el método tipoCanario. No necesita parámetros y devolverá un String indicando el tipo de canario:
    - Si size es mayor que 30 el tipo será "Grande"
    - Si size es mayor o igual que 15 y menor a 30 el tipo será "Mediano"
    - Si size es menor que 15 el tipo será "Enano"
```

Si ejecutamos el siguiente código :

```
Canario unCanario = new Canario('M', 5, 23.0);
System.out.println("CANARIO");
System.out.println(unCanario);
System.out.println(unCanario.trinar());
System.out.println("Tipo: " + unCanario.tipoCanario());
```

El resultado será:

```
CANARIO
Sexo: M Edad: 5
Piopio - Piopio
Tipo: Mediano
```

Respuesta: (penalty regime: 10, 20, ... %)

```
1  public class Canario extends Ave{
2     double size;
3     public Canario(char sexo, int edad, double size) {
4        super(sexo, edad);
5        this.size = size;
6     }
7
8     @Override
9     public String trinar() {
10       return "Piopio - Piopio";
11    }
12
13    public String tipoCanario() {
14       String tipo = "Enano";
15       if(size > 30){
16          tipo = "Grande";
17       }else if(size >= 15) {
18          tipo = "Mediano";
19       }
20       return tipo;
```

	Test	Expected	Got	

✓	`Canario unCanario = new Canario('M', 5, 23.0);`
	`System.out.println(unCanario);`
	`System.out.println(unCanario.trinar());`
	`System.out.println("Tipo: " + unCanario.tipoCanario());`
		`Sexo: M Edad: 5`
		`Piopio - Piopio`
		`Tipo: Mediano`
			`Sexo: M Edad: 5`
			`Piopio - Piopio`
			`Tipo: Mediano`
				✓

✓	`Canario unCanario = new Canario('F', 7, 13.2);`
	`System.out.println(unCanario);`
	`System.out.println(unCanario.trinar());`
	`System.out.println("Tipo: " + unCanario.tipoCanario());`
		`Sexo: F Edad: 7`
		`Piopio - Piopio`
		`Tipo: Enano`
			`Sexo: F Edad: 7`
			`Piopio - Piopio`
			`Tipo: Enano`
				✓

Todas las pruebas superadas. ✓

Question author's solution:

```
public class Canario extends Ave {
 private double size;

 public Canario(char sexo, int edad, double size) {
   super(sexo, edad);
     this.size = size;
       }

   public String trinar() {
      return super.trinar() + " - " + super.trinar();
   }

 public String tipoCanario() {
    String tipo;
   if (size > 30){
     return "Grande";
             }
   if (size >= 15) {
     return "Mediano";
             }
     return "Enano";
       }
}
```

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **10**

Correcta

Puntúa 0,90 sobre 1,00

Marcar pregunta

Tenemos la clase abstracta Empleado :

```
public abstract class Empleado {
    private String nombre;
    private String numSeguridadSocial;

    public Empleado(String nombre, String numero) {
        this.nombre = nombre;
        numSeguridadSocial = numero;
    }

    abstract double salario();

    public String toString() {
        return nombre + " (" + numSeguridadSocial + ")";
    }
}
```

Crea la clase EmpleadoComision a partir de la clase Empleado para que cumpla las siguientes características:

```
1. Tendrá 2 atributos más de tipo doble denominados ventas y comision. Se deberán inicializar en el constructor a partir de los valores indicados como parámetro.
2. El método salario devolverá el resultado de la siguiente operación: ventas * comision / 100
3. El método toString devolverá el siguiente texto: nombre (numSeguridadSocial) Ventas: ventas Comision: comision. Tanto las ventas como la comisión se mostrarán con 2 decimales.
```

Si ejecutamos el siguiente código :

```
EmpleadoComision javier = new EmpleadoComision("Javier Salas", "111111JJ", 60000, 40);
System.out.println(javier);
```

```
System.out.println(javier.salario());
```

El resultado será:

```
Javier Salas (111111JJ) Ventas: 60000,00 Comision: 40,00
24000.0
```

Recuerda interactuar con el código de la superclase mediante la palabra reservada super.

Respuesta: (penalty regime: 10, 20, ... %)

```
1  public class EmpleadoComision extends Empleado {
2      double ventas;
3      double comision;
4      public EmpleadoComision(String nombre, String numero, double ventas, double comision ) {
5          super(nombre, numero);
6          this.ventas = ventas;
7          this.comision = comision;
8      }
9
10     public double salario() {
11         return ventas * comision / 100;
12     }
13
14     public String toString() {
15         //return super.toString() + String.format(" Paga/hora: %.2f Horas: %.2f", ventas, comision);
16         return super.toString() + String.format(" Ventas: %.2f Comision: %.2f", ventas, comision);
17             //Javier Salas (111111JJ) Ventas: 60000,00 Comision: 40,00
18
19     }
20 }
```

	Test	Expected

✓	`EmpleadoComision javier = new EmpleadoComision("Javier Salas", "111111JJ", 60000, 40);`
	`System.out.println(javier);`
	`System.out.println(javier.salario());`
		`Javier Salas (111111JJ) Ventas: 60000.00 Comision: 40.0`
		`24000.0`

✓	`EmpleadoComision laura = new EmpleadoComision("Laura Arana", "222222AA", 40000, 45);`
	`System.out.println(laura);`
	`System.out.println(laura.salario());`
		`Laura Arana (222222AA) Ventas: 40000.00 Comision: 45.00`
		`18000.0`

Todas las pruebas superadas. ✓

Question author's solution:

```
public class EmpleadoComision extends Empleado {
 private double comision;
  private double ventas;

 public EmpleadoComision(String nombre, String numero, double ventas, double comision) {
   super(nombre, numero);
     this.comision = comision;
     this.ventas = ventas;
       }

 public double salario() {
  return ventas * comision / 100;
      }

 @Override
 public String toString(){
  return super.toString() + String.format(" Ventas: %.2f Comision: %.2f", ventas, comision);
      }
}
```

Correcta

Puntos para este envío: 1,00/1,00. Contando con los intentos anteriores, daría **0,90/1,00**.

Finalizar revisión