Navegación por el cuestionario

1 2 3 4 5 6 7 8 9 10

Finalizar revisión

Comenzado el	domingo, 10 de marzo de 2019, 20:47
Estado	Finalizado
Finalizado en	martes, 12 de marzo de 2019, 20:08
Tiempo empleado	1 día 23 horas
Calificación	**7,75** de 10,00 (**78**%)

Pregunta **1**

Parcialmente correcta

Puntúa 0,75 sobre 1,00

Marcar pregunta

Tenemos las siguientes clases:

```
public class Vehiculo {...}
public class Coche extends Vehiculo {...}
public class Todoterreno extends Coche {...}
```

Indica cuál de las siguientes declaraciones son correctas:

Seleccione una o más de una:

- [x] a. Vehiculo v = new Coche(); ✓
- [ ] b. Coche c = new Vehiculo();
- [ ] c. Todoterreno t = new Car();
- [ ] d. Coche c = new Todoterreno();
- [x] e. Vehiculo v = new Todoterreno(); ✓
- [x] f. Todoterreno t = new Todoterreno(); ✓

Respuesta parcialmente correcta.

Ha seleccionado correctamente 3.

La respuesta correcta es: Vehiculo v = new Coche();, Vehiculo v = new Todoterreno();, Coche c = new Todoterreno();, Todoterreno t = new Todoterreno();

Parcialmente correcta

Puntos para este envío: 0,75/1,00.

Pregunta **2**

Correcta

Puntúa 0,70 sobre 1,00

Marcar pregunta

Si dispones de una clase A que es subclase de B y declaras una variable como referencia un objeto de tipo B. Aunque más tarde esa variable haga referencia a un objeto de tipo A (subclase), ¿cuáles serán los miembros a los que podrás acceder sin que el compilador produzca un error?

Seleccione una:

- (•) a. Los miembros de A que hayan sido heredados de B (superclase). ✓
- ( ) b. Los miembros de A que sean específicos de A (subclase).
- ( ) c. Cualquier miembro de A.
- ( ) d. Aquellos miembros de A que no sean abstractos.

La respuesta correcta es: Los miembros de A que hayan sido heredados de B (superclase).

Correcta

Puntos para este envío: 1,00/1,00. Contando con los intentos anteriores, daría **0,70/1,00**.

Pregunta **3**

Parcialmente correcta

Puntúa 0,50 sobre 1,00

Marcar pregunta

Tenemos las siguientes clases e interfaces:

```
public class ClaseA {...}
public interface interfaz1 {...}
public class ClaseB extends ClaseA implements interfaz1 {...}
public class claseC extends claseB {...}
```

Indica cuál de las siguientes declaraciones son correctas:

Seleccione una o más de una:

- [x] a. ClaseA unaClase = new ClaseC(); ✓
- [ ] b. Intefaz1 unaInterfaz = new ClaseA();
- [ ] c. ClaseC unaClase = new ClaseB();
- [ ] d. ClaseC unaClase = new ClaseA();
- [ ] e. Intefaz1 unaInterfaz = new ClaseC();
- [x] f. ClaseA unaClase = new ClaseB(); ✓
- [x] g. Intefaz1 unaInterfaz = new ClaseB(); ✓
- [ ] h. ClaseB unaclase = new ClaseA();

Respuesta parcialmente correcta.

Ha seleccionado correctamente 3.

La respuesta correcta es: ClaseA unaClase = new ClaseB();, ClaseA unaClase = new ClaseC();, Intefaz1 unaInterfaz = new ClaseB();, Intefaz1 unaInterfaz = new ClaseC();

Parcialmente correcta

Puntos para este envío: 0,75/1,00. Contando con los intentos anteriores, daría **0,50/1,00**.

Pregunta **4**

Correcta

Puntúa 0,80 sobre 1,00

Marcar pregunta

Queremos ejecutar la siguiente línea:

```
ClaseA unObjeto = (ClaseA) otroObjeto;
```

Indica las afirmaciones que se deben cumplir:

Seleccione una o más de una:

- [ ] a. ClaseA debe ser de una superclase de la clase de otroObjeto
- [x] b. otroObjeto debe ser de una superclase de ClaseA ✓
- [x] c. otroObjeto debe contener una instancia de la clase ClaseA ✓
- [ ] d. otroObjeto debe ser de una subclase de ClaseA

La respuesta correcta es: otroObjeto debe ser de una superclase de ClaseA, otroObjeto debe contener una instancia de la clase ClaseA

Correcta

Puntos para este envío: 1,00/1,00. Contando con los intentos anteriores, daría **0,80/1,00**.

Pregunta **5**

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;
    }
}
```

Ordena el método equals:

@Override
public boolean equals(Object objeto) {
if (objeto instanceof Triangulo) {
Triangulo unTriangulo = (Triangulo) objeto;
if (base == unTriangulo.base && altura == unTriangulo.altura && id.equals(unTriangulo.id)) {
return true;
}
}
return false;
}

para que al añadírselo a Triangulo, el siguiente código funcione correctamente:

```
public static void buscarTriangulo(Triangulo trianguloBuscar, ArrayList<Triangulo> listaTriangulos) {
   if (listaTriangulos.contains(trianguloBuscar)) {
      System.out.println("Ese triangulo esta en la lista");
   } else {
      System.out.println("Ese triangulo NO esta en la lista");
   }
}
```

}

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;
    }
}
```

Ordena el método equals:

```
[@Override]
[public boolean equals(Object objeto) {]
[if (objeto instanceof Triangulo) {]
[Triangulo unTriangulo = (Triangulo) objeto;]
[if (base == unTriangulo.base && altura == unTriangulo.altura && id.equals(unTriangulo.id)) {]
[return true;]
[}]
[}]
[return false;]
[}]
```

para que al añadírselo a Triangulo, el siguiente código funcione correctamente:

```
public static void buscarTriangulo(Triangulo trianguloBuscar, ArrayList<Triangulo> listaTriangulos) {
   if (listaTriangulos.contains(trianguloBuscar)) {
      System.out.println("Ese triangulo esta en la lista");
   } else {
      System.out.println("Ese triangulo NO esta en la lista");
   }
}
```

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

Pregunta **6**

Correcta

Puntúa 1,00 sobre 1,00

Marcar pregunta

Tenemos la clase Coche :

```
public class Coche {
    private String marca;
    private String modelo;
    private int year;

    public Coche(String marca, String modelo, int year) {
        this.marca = marca;
        this.modelo = modelo;
        this.year = year;
    }

    public String toString() {
        return marca + " " + modelo + " (" + year + ")";
    }
}
```

Corríge la siguiente versión para que implemente correctamente el método equals y cuando se ejecute el siguiente código identifique los coches de la misma marca y modelo como iguales sin tener en cuenta el año:

```
Coche coche1 = new Coche("Fiat", "Panda", 2010);
Coche coche2 = new Coche("Opel", "Meriva", 2010);
Coche coche3 = new Coche("Fiat", "Panda", 2015);
if (!coche1.equals(coche2)) System.out.println("DIFERENTES");
if (coche1.equals(coche3)) System.out.println("IGUALES");
```

El resultado será:

```
DIFERENTES
IGUALES
```

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

Reiniciar respuesta

```java
public class Coche {
    private String marca;
    private String modelo;
    private int year;

    public Coche(String marca, String modelo, int year) {
        this.marca = marca;
        this.modelo = modelo;
        this.year = year;
    }

    public String toString() {
        return marca + " " + modelo + " (" + year + ")";
    }

   public boolean equals(Coche unCoche) {
      if (marca.equals(unCoche.marca) && modelo.equals(unCoche.modelo)) {
         return true;
      }
      return false;
```

	Test	Expected	Got	

✓	`Coche coche1 = new Coche("Fiat", "Panda", 2010);`
	`Coche coche2 = new Coche("Opel", "Meriva", 2010);`
	`Coche coche3 = new Coche("Fiat", "Panda", 2015);`
	`if (!coche1.equals(coche2)) System.out.println("DIFERENTES");`
	`if (coche1.equals(coche3)) System.out.println("IGUALES");`
		DIFERENTES
		IGUALES
			DIFERENTES
			IGUALES
				✓

✓	`Coche coche1 = new Coche("Fiat", "Panda", 2010);`
	`Coche coche2 = new Coche("Opel", "Meriva", 2010);`
	`Coche coche3 = new Coche("Fiat", "Panda", 2015);`
	`if (!coche3.equals(coche2)) System.out.println("DIFERENTES");`
	`if (coche3.equals(coche1)) System.out.println("IGUALES");`
		DIFERENTES
		IGUALES
			DIFERENTES
			IGUALES
				✓

Todas las pruebas superadas. ✓

Question author's solution:

```java
public class Coche {
    private String marca;
    private String modelo;
    private int year;

    public Coche(String marca, String modelo, int year) {
        this.marca = marca;
        this.modelo = modelo;
        this.year = year;
    }

    public String toString() {
        return marca + " " + modelo + " (" + year + ")";
    }

    public boolean equals(Object objeto) {
      if (objeto instanceof Coche) {
         Coche unCoche = (Coche) objeto;
         if (marca.equals(unCoche.marca) && modelo.equals(unCoche.modelo)) {
            return true;
         }
      }
      return false;
    }
}
```

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **7**

Correcta

Puntúa 1,00 sobre 1,00

Marcar pregunta

Tenemos las siguiente clases.

Clase Cosa

```java
public class Cosa {
    public void metodo1() {
        System.out.println("cosa 1");
    }

    public void metodo2() {
        System.out.println("cosa 2");
    }

    public String toString() {
        return "cosa";
    }
}
```

Clase Mueble

```java
public class Mueble extends Cosa {

    public void metodo1() {
        System.out.println("mueble 1");
        super.metodo2();
    }
}
```

Clase Mesa

```java
public class Mesa extends Mueble {

    public void metodo2() {
        System.out.println("mesa 2");
        super.metodo1();
    }

    public String toString() {
        return "mesa";
    }
}
```

Clase MesaJardin

```java
public class MesaJardin extends Mesa {

    public String toString() {
        return "mesa de jardin" + super.toString();
    }
}
```

Escribe lo que mostraría en consola el siguiente código:

```java
ArrayList<Cosa> elementos = new ArrayList<Cosa>();
elementos.add(new MesaJardin());
elementos.add(new Cosa());
elementos.add(1, new Mueble());
elementos.add(new Mesa());

for (Cosa unElemento : elementos) {
    unElemento.metodo1();
    System.out.println(unElemento);
    unElemento.metodo2();
}
```

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

```
mueble 1
cosa 2
mesa de jardinmesa
mesa 2
mueble 1
cosa 2
mueble 1
cosa 2
cosa
cosa 2
cosa 1
cosa
cosa 2
mueble 1
cosa 2
mesa
mesa 2
mueble 1
cosa 2
```

	Expected	Got	Comment	Mark	
✓	mueble 1 cosa 2 mesa de jardinmesa mesa 2 mueble 1 cosa 2 mueble 1 cosa 2 cosa cosa 2 cosa 1 cosa cosa 2 mueble 1 cosa 2 mesa mesa 2 mueble 1 cosa 2	mueble 1 cosa 2 mesa de jardinmesa mesa 2 mueble 1 cosa 2 mueble 1 cosa 2 cosa cosa 2 cosa 1 cosa cosa 2 mueble 1 cosa 2 mesa mesa 2 mueble 1 cosa 2	Line 0 right Line 1 right Line 2 right Line 3 right Line 4 right Line 5 right Line 6 right Line 7 right Line 8 right Line 9 right Line 10 right Line 11 right Line 12 right Line 13 right Line 14 right Line 15 right Line 16 right Line 17 right Line 18 right	1	✓

Todas las pruebas superadas. ✓

Question author's solution:

```
mueble 1
cosa 2
mesa de jardinmesa
mesa 2
mueble 1
cosa 2
mueble 1
cosa 2
cosa
cosa 2
cosa 1
cosa
cosa 2
mueble 1
cosa 2
mesa
mesa 2
mueble 1
cosa 2
```

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **8**

Correcta

Puntúa 1,00 sobre 1,00

Marcar pregunta

Tenemos la clase Cuenta :

```
public class Cuenta {
    public int numero;
    public double saldo;
    public String titular;

    public Cuenta(int numero, double saldo, String titular) {
        this.numero = numero;
        this.saldo = saldo;
        this.titular = titular;
    }

    public String toString() {
        return titular + "(" + numero + "): " + saldo + " euros";
    }
}
```

Desarrolla el método equals para que cuando se ejecute el siguiente código identifique las cuentas con el mismo número y titular sin tener en cuenta el saldo:

```
Cuenta cuenta1 = new Cuenta(233, 111.11, "Silvia");
Cuenta cuenta2 = new Cuenta(322, 555, "Aitor");
Cuenta cuenta3 = new Cuenta(233, 222.22, "Silvia");
if (!cuenta1.equals(cuenta2)) System.out.println("DIFERENTES");
if (cuenta1.equals(cuenta3)) System.out.println("IGUALES");
```

El resultado será:

```
DIFERENTES
IGUALES
```

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

Reiniciar respuesta

```
1  public class Cuenta {
2      public int numero;
3      public double saldo;
4      public String titular;
5
6      public Cuenta(int numero, double saldo, String titular) {
7          this.numero = numero;
8          this.saldo = saldo;
9          this.titular = titular;
10     }
11
12     public String toString() {
13         return titular + "(" + numero + "): " + saldo + " euros";
14     }
15       public boolean equals(Cuenta unaCuenta) {
16       if(numero == unaCuenta.numero && titular.equals(unaCuenta.titular))
17           return true;
18
19       return false;
20     }
```

	Test	Expected	Got	

✓	Cuenta cuenta1 = new Cuenta(233, 111.11, "Silvia");
	Cuenta cuenta2 = new Cuenta(322, 555, "Aitor");
	Cuenta cuenta3 = new Cuenta(233, 222.22, "Silvia");
	if (!cuenta1.equals(cuenta2)) System.out.println("DIFERENTES");
	if (cuenta1.equals(cuenta3)) System.out.println("IGUALES");
		DIFERENTES
		IGUALES
			DIFERENTES
			IGUALES
				✓

✓	Cuenta cuenta1 = new Cuenta(233, 111.11, "Silvia");
	Cuenta cuenta2 = new Cuenta(322, 555, "Aitor");
	Cuenta cuenta3 = new Cuenta(233, 222.22, "Silvia");
	if (cuenta3.equals(cuenta1)) System.out.println("IGUALES");
	if (!cuenta3.equals(cuenta2)) System.out.println("DIFERENTES");
		IGUALES
		DIFERENTES
			IGUALES
			DIFERENTES
				✓

Todas las pruebas superadas. ✓

Question author's solution:

```
public class Cuenta {
    public int numero;
    public double saldo;
    public String titular;

    public Cuenta(int numero, double saldo, String titular) {
        this.numero = numero;
        this.saldo = saldo;
        this.titular = titular;
    }

    public String toString() {
```

```
        return titular + "(" + numero + "): " + saldo + " euros";
    }

    public boolean equals(Object objeto) {
      if (objeto instanceof Cuenta) {
         Cuenta unCuenta = (Cuenta) objeto;
         if (titular.equals(unCuenta.titular) && numero == unCuenta.numero) {
            return true;
         }
      }
      return false;
    }
}
```

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **9**

Correcta

Puntúa 1,00 sobre 1,00

Marcar pregunta

Tenemos la clase Empleado :

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

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

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

Desarrolla el método equals para que cuando se ejecute el siguiente código identifique 2 objetos con los datos del mismo empleado:

```
Empleado ana1 = new Empleado("Ana Salas", "111111A");
Empleado xabi = new Empleado("Xabier Arana", "222222B");
Empleado ana2 = new Empleado("Ana Salas", "111111A");
if (!ana1.equals(xabi)) System.out.println("DIFERENTES");
if (ana1.equals(ana2)) System.out.println("IGUALES");
```

El resultado será:

```
DIFERENTES
IGUALES
```

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

Reiniciar respuesta

```
1  public class Empleado {
2      private String nombre;
3      private String numSeguridadSocial;
4
5      public Empleado(String nombre, String numero) {
6          this.nombre = nombre;
7          numSeguridadSocial = numero;
8      }
9
10     public String toString() {
11         return nombre + " (" + numSeguridadSocial + ")";
12     }
13
14       public boolean equals(Object objeto) {
15       if (objeto instanceof Empleado) {
16          Empleado unEmpleado = (Empleado) objeto;
17          if (nombre.equals(unEmpleado.nombre) && numSeguridadSocial.equals(unEmpleado.numSeguridadSocial)) {
18             return true;
19          }
20       }
```

	Test	Expected	Got	

✓	`Empleado ana1 = new Empleado("Ana Salas", "111111A");`
	`Empleado xabi = new Empleado("Xabier Arana", "222222B");`
	`Empleado ana2 = new Empleado("Ana Salas", "111111A");`
	`if (!ana1.equals(xabi)) System.out.println("DIFERENTES");`
	`if (ana1.equals(ana2)) System.out.println("IGUALES");`
		DIFERENTES
		IGUALES
			DIFERENTES
			IGUALES
				✓

✓	`Empleado ana1 = new Empleado("Ana Salas", "111111A");`
	`Empleado xabi = new Empleado("Xabier Arana", "222222B");`
	`Empleado ana2 = new Empleado("Ana Salas", "111111A");`
	`if (ana2.equals(ana1)) System.out.println("IGUALES");`
	`if (!ana2.equals(xabi)) System.out.println("DIFERENTES");`
		IGUALES
		DIFERENTES
			IGUALES
			DIFERENTES
				✓

Todas las pruebas superadas. ✓

Question author's solution:

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

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

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

   public boolean equals(Object objeto) {
     if (objeto instanceof Empleado) {
        Empleado unEmpleado = (Empleado) objeto;
        if (nombre.equals(unEmpleado.nombre) && numSeguridadSocial.equals(unEmpleado.numSeguridadSocial)) {
           return true;
        }
     }
     return false;
    }
}
```

Correcta

Puntos para este envío: 1,00/1,00.

Pregunta **10**

Incorrecta

Puntúa 0,00 sobre 1,00

Marcar pregunta

Tenemos las siguiente clases.

Clase Coche

```
public class Coche {
    protected String marca;
    protected String modelo;

    public Coche(String marca, String modelo
) {
        this.marca = marca;
        this.modelo = modelo;
    }

    public String toString() {
        return this.getClass().getName() + "
```

Clase CocheElectrico

```
public class CocheElectrico extends Coche {
   private int autonomiaKm;
   public CocheElectrico(String marca, String
modelo, int autonomiaKm) {
      super(marca, modelo);
      this.autonomiaKm = autonomiaKm;
   }

   public int getAutonomia() {
      return autonomiaKm;
   }
```

Clase CocheGasolina

```
public class CocheGasolina extends Coche {

   private int kilometros;

   public CocheGasolina(String marca, String
modelo, int kilometros) {
      super(marca, modelo);
      this.kilometros = kilometros;
   }

   public int getKilometros() {
      return kilometros;
```

```
" + marca + " " + modelo;
    }
}
```

```
    public int viajar(int km) {
        autonomiaKm -= km;
        return autonomiaKm;
    }

    public void repostar(int autonomiaKm) {
        this.autonomiaKm = autonomiaKm;
    }
}
```

```
    }

    public boolean tocaRevision() {
        if (kilometros > 10000) {
            return true;
        }
        return false;
    }
```

Mejora el siguiente programa para que muestre qué coche de eléctrico tiene la mayor autonomía.

Si el array tiene los siguientes elementos:

```
[Coche Opel Meriva, CocheElectrico Renault ZOE, CocheGasolina Fiat Punto, Coche Ford Fiest, CocheElectrico Hyundai Ioniq, CocheGasolina Renault Clio]
```

El resultado será:

```
CocheElectrico Renault ZOE 300
```

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

Reiniciar respuesta

```
1  import java.util.*;
2
3  public class CochesMain {
4     public static void main(String[] args) {
5        Coche[] arrayCoches = {new Coche("Opel", "Meriva"), new CocheElectrico("Renault", "ZOE", 300),
6           new CocheGasolina("Fiat", "Punto", 15000), new Coche("Ford", "Fiest"),
7           new CocheElectrico("Hyundai", "Ioniq", 200), new CocheGasolina("Renault", "Clio", 1500)};
8
9        System.out.println(Arrays.toString(arrayCoches));
10       System.out.println();
11
12       // Detectamos el coche electrico con mayor autonomia
13
14 }
```

Syntax Error(s)

```
CochesMain.java:14: error: reached end of file while parsing
}
 ^
1 error
```

Question author's solution:

```
import java.util.*;

public class CochesMain {
   public static void main(String[] args) {
      Coche[] arrayCoches = {new Coche("Opel", "Meriva"), new CocheElectrico("Renault", "ZOE", 300),
         new CocheGasolina("Fiat", "Punto", 15000), new Coche("Ford", "Fiest"),
         new CocheElectrico("Hyundai", "Ioniq", 200), new CocheGasolina("Renault", "Clio", 1500)};

      System.out.println(Arrays.toString(arrayCoches));
      System.out.println();

      // Coche electrico con mayor autonomia
      CocheElectrico mayorAutonomia = null;
      for (Coche unCoche : arrayCoches) {
         if (unCoche instanceof CocheElectrico) {
            CocheElectrico electrico = (CocheElectrico) unCoche;
            if (mayorAutonomia == null || (electrico.getAutonomia() > mayorAutonomia.getAutonomia())) {
               mayorAutonomia = electrico;
            }
         }
      }
      System.out.println(mayorAutonomia + " " + mayorAutonomia.getAutonomia());
   }
}
```

Incorrecta

Puntos para este envío: 0,00/1,00.

Finalizar revisión