

Estado Finalizado

Comenzado martes, 3 de febrero de 2026, 17:25

Completado martes, 3 de febrero de 2026, 19:55

Duración 2 horas 30 minutos

Calificación 61 de 100

Comentario - Aprobado

Pregunta 1

Sin contestar

Sin calificar

A continuación están listados los recursos a los que puede acceder durante el examen. Recuerde que **no está permitido acceder a otras aplicaciones o páginas**, ya que anularía el intento.

En la IDE encontrará el esqueleto del proyecto para su referencia, y que le permitirá compilar e implementar su código (su uso no es obligatorio, puede o no utilizarlo). El código que se evalúa es el que se encuentre en la respuesta a cada pregunta de este cuestionario.

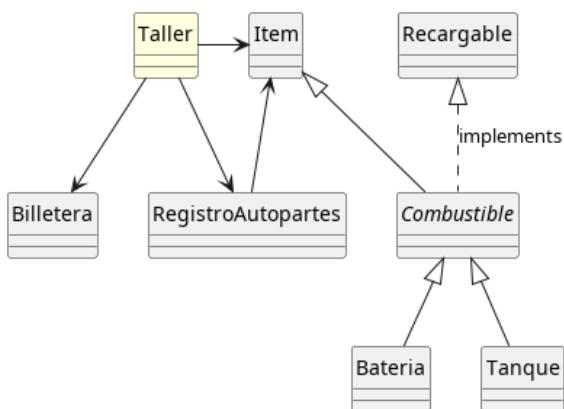
- [Proyecto TallerMecanico.zip](#)
- [JAVA API](#)

Enunciado

Este proyecto representa un **Taller Mecánico**.

Diagrama de clases

El diagrama de clases a continuación resume las principales relaciones. El alumno deberá completar la implementación de algunos métodos y clases solicitados, según la documentación.



Pregunta 2

Correcta

Se puntúa 18 sobre 20

ATENCIÓN

1. El botón "**Precheck**" prueba el programa para el ejemplo. Lo pueden utilizar todas las veces que necesiten.
2. El botón "**Comprobar**" aplica todas las pruebas. Tienen tres usos sin penalidad, a partir de la tercera se descontarán puntos en la nota final.
3. TODO debe estar dentro de las funciones predefinidas.

ENUNCIADO

Implemente los siguientes métodos de la clase Item según la documentación de la misma.

<i>«abstract»</i> Item
-String id
-String descripcion
-Double precio
+Item (String id)
+Item (String id, String descripcion)
+Item (String id, String descripcion, Double precio)
-generarId (String id) : String
+getPrecioConDescuento(Double porcentaje) : Double

Por ejemplo:

Prueba	Resultado
// Test constructor Item bat = new ElemTest("12v 55A", 75.0); try { System.out.println(bat.getId()); System.out.println(bat.getDescripcion()); System.out.println(bat.getPrecio()); System.out.println(bat); System.out.println("Ok"); } catch (Exception e) { System.out.println(e); }	bat_1 12v 55A 75.0 Item [id=bat_1, descripcion=12v 55A, precio=75.0] Ok

Respuesta: (sistema de penalización: 0, 0, 0, 5, 10, 15 %)

[Reiniciar respuesta](#)

```

1 abstract class Item
2 {
3     private static Integer contador = 0;
4
5     private final String id;
6     private String descripcion;
7     private Double precio;
8
9     /**
10      * Constructor simple que genera el id y setea la descripcion
11      * en el texto "Sin descripcion"
12      *
13      * @param id    El id del item
14      */
15     public Item(final String id)
16     {
17         // TODO - Implementar constructor
18         this.id = this.generarId(id);
19         this.descripcion = "Sin descripcion";
20     }
21
22     /**
23      * Constructor que genera el id y setea la descripcion
24      *
25      * @param id          El id del item
26      * @param descripcion La descripcion del elemento

```

```
27     */
28     public Item (final String id, String descripcion)
29     {
30         // TODO - Implementar constructor
31         this.id = this.generarId(id);
32         this.descripcion = descripcion;
33     }
34
35 /**
36 * Constructor que genera el id y setea la descripcion y el precio
37 *
38 * @param id El id del item
39 * @param descripcion La descripción del elemento
40 * @param precio El precio del producto
41 */
42     public Item (final String id, String descripcion, Double precio)
43     {
44         // TODO - Implementar constructor
45         this.id = this.generarId(id);
46         this.descripcion = descripcion;
47         this.precio = precio;
48     }
49
50 /**
51 * Genera el ID del producto uniendo el ID provisto con el valor
52 * del contador de elementos separados por un guion bajo " ".
```

Todas las pruebas superadas.

Correcta

Puntos para este envío: 20/20. Contando con los intentos anteriores, daría **18/20**.

Pregunta 3

Parcialmente correcta

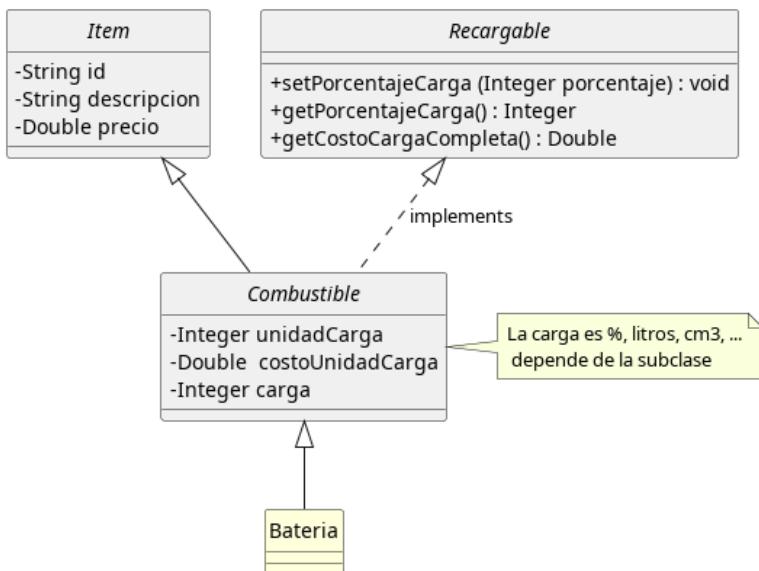
Se puntúa 14 sobre 20

ATENCIÓN

1. El botón "**Precheck**" prueba el programa para el ejemplo. Lo pueden utilizar todas las veces que necesiten.
2. El botón "**Comprobar**" aplica todas las pruebas. Tienen tres usos sin penalidad, a partir de la tercera se descontarán puntos en la nota final.
3. TODO debe estar dentro de las funciones predefinidas.

ENUNCIADO

Implemente los siguientes métodos de la clase Batería según la documentación de la misma.

**Por ejemplo:**

Prueba	Resultado
<pre>// Test constructor Bateria bat = new Bateria("bat", "12v-45A", 0.2); try { System.out.println(bat.getId()); System.out.println(bat.getDescripcion()); System.out.println(bat.getCostoUnidadCarga()); System.out.println("Ok"); } catch (Exception e) { System.out.println(e); }</pre>	bat_1 12v-45A 0.2 Ok

Respuesta: (sistema de penalización: 0, 0, 0, 5, 10, 15 %)**Reiniciar respuesta**

```

1 /**
2  * Representa una batería que puede cargarse y descargarse.
3  *
4  * La unidad de carga depende del modelo, pudiendo ser cualquier
5  * porcentaje como ser 1%, 2%, ..., 10%, etc.
6  */
7 class Bateria extends Combustible
8 {
9     /**
10      * Un constructor para una batería con 70% de carga inicial y unidad de carga de 5%.
11      *
12      * @param id             El código del producto
13      * @param descripcion    Detalle de la batería
14      * @param costoUnidadCarga Precio del costo de recarga por unidad (5%)
15      */
16     public Bateria(final String id, String descripcion, Double costoUnidadCarga)
17     {
18         ...
19     }
20 }
```

```
18     // TODO Implementar el constructor
19     super(id, descripcion, costoUnidadCarga, 5, 70);
20 }
21
22 /**
23 * Un constructor para una batería con 50% de carga inicial.
24 *
25 * @param id El código del producto
26 * @param descripcion Detalle de la batería
27 * @param unidadCarga Un valor en porcentaje
28 * @param costoUnidadCarga Precio del costo de recarga por unidad
29 */
30 public Bateria(final String id, String descripcion, Integer unidadCarga, Double costoUnidadCarga)
31 {
32     // TODO Implementar el constructor
33     super(id, descripcion, costoUnidadCarga, unidadCarga, 50);
34 }
35
36 // TODO Implementar los métodos faltantes de la interface Recargable
37 /**
38 * Nuevo porcentaje de carga. El valor final no puede ser
39 * menor a 0% ni mayor a 100%.
40 *
41 * @param porcentaje Nuevo porcentaje de carga
42 */
43 public void setPorcentajeCarga(Integer porcentaje)
44 {
45     if(porcentaje < 0 || porcentaje > 100)
46     {
47         return;
48     }
49     super.setCarga(porcentaje);
50 }
51
52 /**
53 *
```

Tu programa ha fallado en uno o más de las pruebas ocultas

Parcialmente correcta

Puntos para este envío: 14/20.

Pregunta 4

Parcialmente correcta

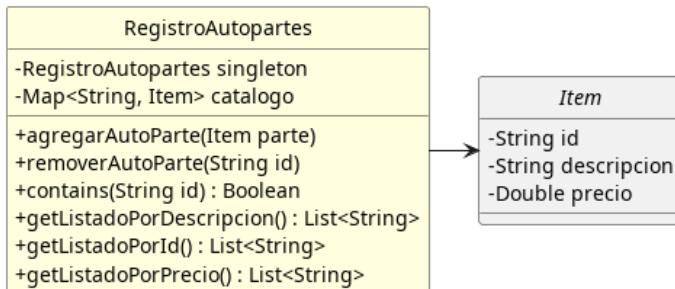
Se puntúa 17 sobre 30

ATENCIÓN

1. El botón "**Precheck**" prueba el programa para el ejemplo. Lo pueden utilizar todas las veces que necesiten.
2. El botón "**Comprobar**" aplica todas las pruebas. Tienen tres usos sin penalidad, a partir de la tercera se descontarán puntos en la nota final.
3. TODO debe estar dentro de las funciones predefinidas.

ENUNCIADO

Implemente los siguientes métodos de la clase RegistroAutopartes según la documentación de la misma.

**Por ejemplo:**

Prueba	Resultado
<pre> System.out.println("--> Registro AutoParte agregarAutoParte"); try { RegistroAutopartes registro = RegistroAutopartes.getRegistro(); Map<String, Item> m = registro.getCatalogo(); Bateria bateria = new Bateria("item1","Bateria comun",10.0); registro.agregarAutoParte(bateria); System.out.println ("Total items: " + m.size()); System.out.println ("Item agregado: " + (m.get(bateria.getId()) != null)); m.clear(); } catch (Exception e) { System.out.println(e); } </pre>	<pre> --> Registro AutoParte agregarAutoParte Total items: 1 Item agregado: true </pre>

Respuesta: (sistema de penalización: 0, 0, 0, 5, 10, 15 %)**Reiniciar respuesta**

```

1 import java.util.HashMap;
2 import java.util.Map;
3 import java.util.Iterator;
4 import java.util.List;
5 import java.util.TreeSet;
6 import java.util.ArrayList;
7
8 public class RegistroAutopartes
9 {
10     private static RegistroAutopartes singleton = null;
11     private Map<String, Item> catalogo;           // La key es el ID del Item
12
13     // Constructor Privado
14     private RegistroAutopartes()
15     {
16         this.catalogo = new HashMap<>();
17     }
18
19     public Map<String, Item> getCatalogo()
20     {
21         return catalogo;
22     }

```

```
22
23
24 /**
25 * Este metodo asegura que siempre se devuelva el mismo objeto registro desde
26 * cualquier lado que se llame. Ej:
27 *
28 * RegistroAutopartes miRegistro = RegistroAutopartes.getRegistro();
29 *
30 * @return El registro existente.
31 */
32 public static RegistroAutopartes getRegistro()
33 {
34     if(singleton == null)
35     {
36         singleton = new RegistroAutopartes();
37     }
38     return singleton;
39 }
40
41 /**
42 * Limpia el registro, dejandolo vacio
43 */
44 public void limpiarRegistro()
45 {
46     this.catalogo.clear();
47 }
48
49 /**
50 * Agrega una autoparte al registro.
51 *
52 * @param parte
53 *          La autoparte a agregar.
```

Tu programa ha fallado en uno o más de las pruebas ocultas

Parcialmente correcta

Puntos para este envío: 17/30.

Pregunta 5

Parcialmente correcta

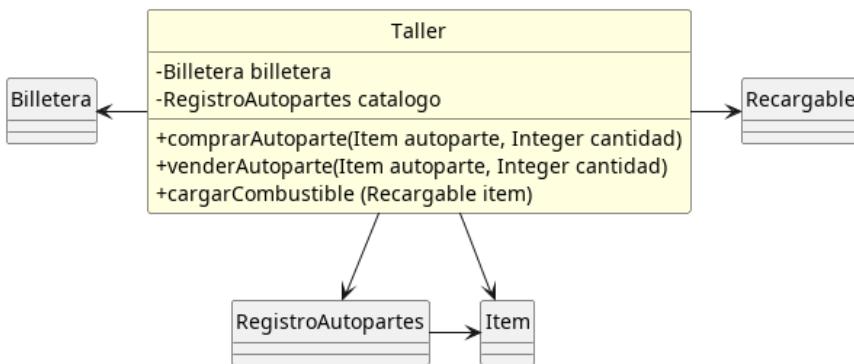
Se puntúa 12 sobre 30

ATENCIÓN

1. El botón "**Precheck**" prueba el programa para el ejemplo. Lo pueden utilizar todas las veces que necesiten.
2. El botón "**Comprobar**" aplica todas las pruebas. Tienen tres usos sin penalidad, a partir de la tercera se descontarán puntos en la nota final.
3. TODO debe estar dentro de las funciones predefinidas.

ENUNCIADO

Implemente los siguientes métodos de la clase Taller según la documentación de la misma.

**Por ejemplo:**

Prueba	Resultado
<pre>// Test constructor Taller t = new Taller(); try { System.out.println(t.getRegistroAutopartes()); System.out.println(t.getBilletera()); System.out.println("Ok"); } catch (Exception e) { System.out.println(e); }</pre>	<pre>RegistroAutopartes [size=0, catalogo={}] Billetera -> Fondos: \$100.0 Ok</pre>

Respuesta: (sistema de penalización: 0, 0, 0, 5, 10, 15 %)**Reiniciar respuesta**

```

1 /**
2  * La clase Taller modela las gestiones de compra y venta de autopartes.
3  *
4  * Se comienza con dinero en la billetera y se debe comprar y vender autopartes.
5  */
6 public class Taller
7 {
8     private Billetera billetera;
9     private RegistroAutopartes catalogo;
10
11 /**
12  * Crea el objeto Taller, desde el cual se gestiona al resto de los objetos.
13  *
14  * Debe instanciarse el RegistroAutopartes y la billetera con $100.
15  *
16  * Inicialmente no hay ninguna autoparte.
17  */
18 public Taller()
19 {
20     // TODO implementar metodo
21     this.billetera = new Billetera(100.0);
22     this.catalogo = RegistroAutopartes.getRegistro();
23 }
24
25 public void comprarAutoparte(Item item)
26 {
27     // Implementar metodo
28 }
  
```

```
27     comprarAutoparte(item, 1);  
28 }  
29  
30 /**  
31 * Se utiliza para comprar una autoparte y agregarla al catalogo.  
32 *  
33 * La operacion se concreta si hay dinero suficiente en la billetera.  
34 *  
35 * Notar que la billetera puede lanzar DineroInsuficienteException.  
36 *  
37 * Si no hay dinero suficiente, se imprime el mensaje "Compra fallida".  
38 *  
39 * @param autoparte La autoparte a comprar.  
40 * @param cantidad La cantidad a comprar.  
41 */  
42 public void comprarAutoparte(Item autoparte, Integer cantidad)  
43 {  
44     // TODO implementar metodo  
45     try  
46     {  
47         if(this.billetera.getPesos() >= autoparte.getPrecio()*cantidad)  
48         {  
49             this.billetera.gastar(autoparte.getPrecio()*cantidad);  
50             for(int i = 0; i < cantidad; i++)  
51             {  
52                 this.catalogo.agregarAutoParte(autoparte);  
53             }  
54         }  
55     }  
56 }
```

Tu programa ha fallado en uno o más de las pruebas ocultas

Parcialmente correcta

Puntos para este envío: 12/30.