Asumiento que los métodos agregarElemento y obtenerElemento están correctamente implementados, el siguiente código Da error en tiempo de compilación en la línea 11Da error en tiempo de compilación en la línea 6Está perfectoDa error en tiempo de compilación en la línea 5Da error en tiempo de compilación en la línea 2Da error en tiempo de compilación en la línea 10Da error en tiempo de ejecución

```
1 public class Coleccion {

2    private final int TAMANIO;

3    private int ultimo;

4    private Object[] elementos;



5    public Coleccion(int tamanio) {

6        this.TAMANIO = tamanio;

7        ultimo = 0;

8        this.elementos = new Object[TAMANIO];

9    }
```

```
10   public void cambiarTamanio(int nuevoTamanio){

11       this.TAMANIO = nuevoTamanio;

12   }

   public void agregarElemento(Object elem){

       ...

   }

   public Object obtenerElemento(int pos){

       ...

   }

}
```

Retroalimentación

La respuesta correcta es:

Asumiento que los métodos agregarElemento y obtenerElemento están correctamente implementados, el siguiente código [Da error en tiempo de compilación en la línea 11]

```
1 public class Coleccion {

2    private final int TAMANIO;

3    private int ultimo;

4    private Object[] elementos;



5    public Coleccion(int tamanio) {

6        this.TAMANIO = tamanio;

7        ultimo = 0;

8        this.elementos = new Object[TAMANIO];

9    }



10   public void cambiarTamanio(int nuevoTamanio){
```

```
11        this.TAMANIO = nuevoTamanio;

12    }

    public void agregarElemento(Object elem){

        ...

    }

    public Object obtenerElemento(int pos){

        ...

    }

}
```

Pregunta **2**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Las variables de instancia Son variables estáticas dentro de una clase, pero fuera de cualquier método.Ninguna opción es correctaSon variables definidas dentro de métodos y constructores.Son variables dentro de una clase, pero fuera de cualquier método.

Retroalimentación

La respuesta correcta es:

Las variables de instancia [Son variables dentro de una clase, pero fuera de cualquier método.]

Pregunta **3**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

El siguiente código Da error en tiempo de compilaciónEstá perfectoDa error en tiempo de ejecuciónTiene un error conceptual

```
public class MathUtils {

   public static int sumar(int a, int b){

       return a + b;

   }



   public int multiplicar(int a, int b){

       return a * b;

   }



   public static void main(String[] args) {

       MathUtils mathUtils = new MathUtils();

       int result1 = mathUtils.sumar(5,10);

       System.out.println("La suma da: "+result1);
```

```
        int result2 = mathUtils.multiplicar(2,3);

        System.out.println("La multiplicacion da: "+result2);

    }



}
```

Retroalimentación

La respuesta correcta es:

El siguiente código [Tiene un error conceptual]

```
public class MathUtils {

    public static int sumar(int a, int b){

        return a + b;

    }



    public int multiplicar(int a, int b){
```

```
        return a * b;

    }



    public static void main(String[] args) {

        MathUtils mathUtils = new MathUtils();

        int result1 = mathUtils.sumar(5,10);

        System.out.println("La suma da: "+result1);

        int result2 = mathUtils.multiplicar(2,3);

        System.out.println("La multiplicacion da: "+result2);

    }



}
```

Pregunta **4**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Cuál de las siguiente líneas de código define un atributo de clase?

a.

private int elemento = 0;

b.

public final int elemento = 0;

c.

protected static int elemento = 0;

Retroalimentación

La respuesta correcta es:

```
protected static int elemento = 0;
```

Pregunta 5

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Cuál es la forma de declarar una Constante en Java para que esté accesible para todas las clases? Por ejemplo, EDAD_MIN = 18

a.

public static final int EDAD_MIN = 18;

b.

public static final int EDAD_MIN; //y en el constructor del objeto inicializo la variable con 18

c.

Cualquiera de las dos anteriores es correcta

d.

public static int EDAD_MIN = 18;

e.

protected static final int EDAD_MIN = 18;

Retroalimentación

La respuesta correcta es:

public static final int EDAD_MIN = 18;

Pregunta **6**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Cómo cambiaría el valor de un atributo llamado "ultimoCampeonato" declarado como "static" en una clase Argentina como se muestra a continuación?

```
public class Argentina {

   private static int ultimoCampleonato = 1993;

...

}
```

Elegir cuál de las opciones que se enuncian es la correcta

a.

Debería existir un método en la clase Argentina:

```
public void setUltimoCampeonato(int valor) {
    ultimoCampeonato = valor;

}
```

y llamarlo

Argentina.setUltimoCampeonato(2021)

b.

Debería existir un método en la clase Argentina:

```
public static void setUltimoCampeonato(int valor) {
    ultimoCampeonato = valor;

}
```

y llamarlo

Argentina.setUltimoCampeonato(2021)

c.

Argentina.ultimoCampeonato = 2021

d.

```
Argentina temp = new Argentina();
temp.ultimoCampeonato = 2021
```

Retroalimentación

La respuesta correcta es: Debería existir un método en la clase Argentina:

```
public static void setUltimoCampeonato(int valor) {
    ultimoCampeonato = valor;

}
```

y llamarlo

```
Argentina.setUltimoCampeonato(2021)
```

Pregunta 7

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Existe algún mecanismo en Java para acceder a un método de una clase sin crear un objeto de dicha clase?

a.

No

b.

Si

Retroalimentación

La respuesta correcta es:

Si

Pregunta **8**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dadas las siguientes clases A y B

```
public final class A {

   int valor;

   public A(int valor) {

       this.valor = valor;

   }

   public int getValor() {

       return valor;

   }

}

public class B extends A{
```

```
   public B(int valor) {

       super(valor*2);

   }

   public int getValor() {

       return super.getValor()*2;

   }

}
```

¿Qué imprime el siguiente código?

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

   A b1 = (A) new B(12);
```

```
   System.out.println(b1.getValor());


}
```

a.

36

b.

48

c.

No imprime nada, hay un error en tiempo de compilación en la clase B

d.

No imprime nada hay un error en tiempo de ejecución

e.

No imprime nada, hay un error en tiempo de compilación en el main

f.

No imprime nada, hay un error en tiempo de compilación en la clase A

g.

24

h.

6

Retroalimentación

La respuesta correcta es:

No imprime nada, hay un error en tiempo de compilación en la clase B

Pregunta **9**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dado el siguiente código

```
public class Alumno {

   private int cantidadAlumnos;

   private String nombre;

   private int legajo;



   public Alumno(String nombre) {

       this.nombre = nombre;

       legajo = cantidadAlumnos;

       cantidadAlumnos++;
```

```
    }




    public int getLegajo() {


        return legajo;


    }




    public void setLegajo(int legajo) {


        this.legajo = legajo;


    }


}
```

Indique qué imprime por consola el siguiente main

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

```
   Alumno a1 = new Alumno("Marcelo");



   Alumno a2 = new Alumno("Luis");



   Alumno a3 = new Alumno("Ariel");



   System.out.println(a3.getLegajo());



}
```

Respuesta:

Retroalimentación

La respuesta correcta es: 0

Pregunta **10**

Parcialmente correcta

Se puntúa 1,00 sobre 3,00

Marcar pregunta

Enunciado de la pregunta

Considere el siguiente código:

```
public abstract class Empleado {

   String nombre;



   public Empleado(String nombre) {

       this.nombre = nombre;

   }



   public abstract double getSueldo();



   public String getNombre() {

       return nombre;

   }



   public void setNombre(String nombre) {

       this.nombre = nombre;
```

```
    }

    public String toString(){

        return getNombre()+"-"+getSueldo();

    }

}



public class EmpleadoSueldo extends Empleado{

    private double sueldo;



    public EmpleadoSueldo(double sueldo) {

        super("SueldoFijo");

        this.sueldo = sueldo;

    }



    public double getSueldo() {

        return sueldo;

    }
```

```
}

public class EmpleadoComision extends Empleado{

   private double totalVentas;

   private double comision;



   public EmpleadoComision(double totalVentas, double
comision) {

       super("Comision");

       this.totalVentas = totalVentas;

       this.comision = comision;

   }



   public double getSueldo(){

       return totalVentas * comision;

   }

}
```

Dado el main que se muestra a continuación, seleccione de la lista desplegable el código faltante para que, como resultado, el programa imprima "Comision-349.0"?

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

   EmpleadoSueldo miEmpleado = new EmpleadoSueldo(349.0);
```

Respuesta

```
myEmpleado.setNombre("Comision");myEmpleado = new
EmpleadoComision(349.0, 1);myEmpleado = new
Empleado("Comision - 349");
```

```
   System.out.println(miEmpleado);

}
```

Dado el siguiente main, seleccione de la lista desplegable el código faltante para que, como resultado, el programa imprima "SueldoFijo-200.0"?

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

```
   Empleado porComision = null;
```

Respuesta

```
porComision.setNombre("SueldoFijo-200");porComision = new
EmpleadoSueldo(200);porComision = new
EmpleadoComision(200, 1);porComision = "SueldoFijo-349";
```

```
   System.out.println(porComision);
}
```

Si incorporamos una nueva clase de empleado que cobra su sueldo en base a las horas extra mediante el siguiente código Java:

```
public class EmpleadoHorasExtra {

   private double cantidadHoras;

   private double sueldoHora;
```

```
   public EmpleadoHorasExtra(double cantidadHoras, double
sueldoHora) {

       this.cantidadHoras = cantidadHoras;

       this.sueldoHora = sueldoHora;

   }



   public double getSueldo(){

       return cantidadHoras*sueldoHora;

   }

}
```

¿qué sucede con la ejecución del siguiente método main?

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

   Empleado porHoras = new EmpleadoHorasExtra(10,25);

   System.out.println(porHoras.getSueldo());

}
```

Imprime por pantalla "null-250"

Imprime por pantalla "PorHoras-250"

Genera un error de Null Pointer Exception

No compila

Imprime por pantalla "250"

La respuesta correcta es: No compila

Pregunta **11**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dado el siguiente enunciado:

Una persona asiste a un congreso si TODOS los temas que considera importantes (por ejemplo "inteligencia artificial",

"programación", "java", etc.), son temáticas características del congreso.

considere el siguiente código Java

```
public class Asistente {

   private String nombre;

   private ArrayList<String> intereses;



   public Asistente(String nombre) {

       this.nombre = nombre;

       this.intereses = new ArrayList<>();

   }
```

```
   public ArrayList<String> getIntereses() {

       return new ArrayList<>(intereses);

   }

   public boolean cumple(Congreso congreso){

       int i = 0;

       while (i<intereses.size() &&
congreso.tieneCaracteristica(intereses.get(i)))

               i++;

       return i==intereses.size();

   }

}
```

```
public class Congreso {

   private ArrayList<String> caracteristicas = new
ArrayList<>();

   public boolean tieneCaracteristica(String
caracteristica){

       return caracteristicas.contains(caracteristica);

   }

   public boolean cumple(Asistente asistente){

       ArrayList<String> caractAsistente =
asistente.getIntereses();

       int i=0;
```

```
        while (i<caractAsistente.size() &&
tieneCaracteristica(caractAsistente.get(i)))



            i++;



        return i==caractAsistente.size();



    }


}
```

¿Cuál de los siguientes métodos cree que delega correctamente las responsabilidades?

a.

public boolean cumple(Congreso congreso) de la clase Asistente

b.

public boolean cumple(Asistente asistente) de la clase Congreso

c.

Ambos métodos delegan correctamente las responsabilidades

d.

Ninguno de los metodos delega correctamente las responsabilidades

Retroalimentación

La respuesta correcta es: public boolean cumple(Congreso congreso) de la clase Asistente

Pregunta 12

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dado el siguiente enunciado:

Una prestigiosa academia de magos esta organizando a sus Magos. Algunos magos tienen un "poder de magia" total definido. Existen otros magos peculiares cuyo poder de magia depende de la letra R y del día de la semana. Por ejemplo, si el día contiene una R el poder

del mago merma un 50%, por lo tanto, los lunes su poder no merma, pero los martes si.

Considere la siguiente solución al problema

```
public abstract class Mago {

   private String nombreMagico;



   public Mago(String nombreMagico) {

       this.nombreMagico = nombreMagico;

   }



   public abstract double getPoderMagico();
```

```
   public String getNombre(){

       return nombreMagico;

   }

}

public class MagoSupremo extends Mago{

   private double poderMagico;



   public MagoSupremo(String nombre, double poder) {

       super(nombre);

       this.poderMagico = poder;

   }
```

```
   public double getPoderMagico() {

       return poderMagico;

   }

}

public abstract class MagoPeculiar extends Mago{

   private double poderMagico;

   private double merma;

   private String letra;



   public MagoPeculiar(String nombre, double poder, double
merma, String letra) {

       super(nombre);
```

```
        this.poderMagico = poder;


        this.merma = merma;


        this.letra = letra;


    }




    public double getPoderMagico() {


        LocalDate hoy = LocalDate.now();


        String nombreDia = hoy.getDayOfWeek().getDisplayName(


                TextStyle.FULL, Locale.getDefault()


        );


        if (nombreDia.contains(letra))


            return poderMagico - poderMagico*merma;
```

```
        else


            return poderMagico;


    }


}
```

Nota: LocalDate y DayOfWeek se utilizan solo para obtener el nombre del día de la semana y no hay errores en el uso de estas clases

Indique cual de las siguientes afirmaciones es correcta

NOTA: se puede seleccionar más de una, y las respuestas incorrectas RESTAN puntaje

a.

La clase Mago por ser abstracta no deberia tener constructor

b.

El código está perfecto

c.

Se debería abstraer en la clase Mago el atributo poderMagico. La clase mago Supremo no es más necesaria y la clase Mago debería ser concreta

d.

La clase MagoPeculiar deberia ser concreta

e.

EL código no compila

f.

Se deberia abstraer en la clase Mago el atributo poderMagico. El atributo deberia ser Protected para que pueda ser usado por los hijos MagoSupremo y Mago Peculiar. La clase Mago debe seguir siendo abstracta.

g.

Falla en tiempo de ejecución porque la clase MagoPeculiar es abstracta y nadie heredo de ella

Retroalimentación

Las respuestas correctas son:

Se debería abstraer en la clase Mago el atributo poderMagico. La clase mago Supremo no es más necesaria y la clase Mago debería ser concreta,

La clase MagoPeculiar deberia ser concreta

Asumiento que los métodos agregarElemento y obtenerElemento están correctamente implementados, el siguiente código Da error en tiempo de compilación en la línea 11Da error en tiempo de compilación en la línea 6Está perfectoDa error en tiempo de compilación en la línea 5Da error en tiempo de compilación en la

línea 2Da error en tiempo de compilación en la línea 10Da error en tiempo de ejecución

```
1 public class Coleccion {

2    private final int TAMANIO;

3    private int ultimo;

4    private Object[] elementos;



5    public Coleccion(int tamanio) {

6        this.TAMANIO = tamanio;

7        ultimo = 0;

8        this.elementos = new Object[TAMANIO];

9    }



10   public void cambiarTamanio(int nuevoTamanio){

11       this.TAMANIO = nuevoTamanio;
```

```
12    }

    public void agregarElemento(Object elem){

        ...

    }

    public Object obtenerElemento(int pos){

        ...

    }

}
```

Retroalimentación

La respuesta correcta es:

Asumiento que los métodos agregarElemento y obtenerElemento están correctamente implementados, el siguiente código [Da error en tiempo de compilación en la línea 11]

```
1 public class Coleccion {
```

```
2    private final int TAMANIO;
3    private int ultimo;
4    private Object[] elementos;


5    public Coleccion(int tamanio) {
6        this.TAMANIO = tamanio;
7        ultimo = 0;
8        this.elementos = new Object[TAMANIO];
9    }


10   public void cambiarTamanio(int nuevoTamanio){
11       this.TAMANIO = nuevoTamanio;
12   }
   public void agregarElemento(Object elem){
       ...
   }
   public Object obtenerElemento(int pos){
       ...
```

```
    }


}
```

Pregunta **2**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Las variables de instancia Son variables estáticas dentro de una clase, pero fuera de cualquier método.Ninguna opción es correctaSon variables definidas dentro de métodos y constructores.Son variables dentro de una clase, pero fuera de cualquier método.

Retroalimentación

La respuesta correcta es:

Las variables de instancia [Son variables dentro de una clase, pero fuera de cualquier método.]

Pregunta 3

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

El siguiente código Da error en tiempo de compilaciónEstá perfectoDa error en tiempo de ejecuciónTiene un error conceptual

```
public class MathUtils {

   public static int sumar(int a, int b){

       return a + b;

   }



   public int multiplicar(int a, int b){

       return a * b;

   }
```

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

       MathUtils mathUtils = new MathUtils();

       int result1 = mathUtils.sumar(5,10);

       System.out.println("La suma da: "+result1);

       int result2 = mathUtils.multiplicar(2,3);

       System.out.println("La multiplicacion da: "+result2);

   }



}
```

Retroalimentación

La respuesta correcta es:

El siguiente código [Tiene un error conceptual]

```
public class MathUtils {
```

```
    public static int sumar(int a, int b){

        return a + b;

    }



    public int multiplicar(int a, int b){

        return a * b;

    }



    public static void main(String[] args) {

        MathUtils mathUtils = new MathUtils();

        int result1 = mathUtils.sumar(5,10);

        System.out.println("La suma da: "+result1);

        int result2 = mathUtils.multiplicar(2,3);

        System.out.println("La multiplicacion da: "+result2);

    }



}
```

Pregunta **4**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Cuál de las siguiente líneas de código define un atributo de clase?

a.

private int elemento = 0;

b.

public final int elemento = 0;

c.

protected static int elemento = 0;

Retroalimentación

La respuesta correcta es:

protected static int elemento = 0;

Pregunta **5**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Cuál es la forma de declarar una Constante en Java para que esté accesible para todas las clases? Por ejemplo, EDAD_MIN = 18

a.

public static final int EDAD_MIN = 18;

b.

public static final int EDAD_MIN; //y en el constructor del objeto inicializo la variable con 18

c.

Cualquiera de las dos anteriores es correcta

d.

public static int EDAD_MIN = 18;

e.

protected static final int EDAD_MIN = 18;

Retroalimentación

La respuesta correcta es:

public static final int EDAD_MIN = 18;

Pregunta **6**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Cómo cambiaría el valor de un atributo llamado "ultimoCampeonato" declarado como "static" en una clase Argentina como se muestra a continuación?

```
public class Argentina {

   private static int ultimoCampleonato = 1993;

...

}
```

Elegir cuál de las opciones que se enuncian es la correcta

a.

Debería existir un método en la clase Argentina:

```
public void setUltimoCampeonato(int valor) {
    ultimoCampeonato = valor;

}
```

y llamarlo

```
Argentina.setUltimoCampeonato(2021)
```

b.

Debería existir un método en la clase Argentina:

```
public static void setUltimoCampeonato(int valor) {
    ultimoCampeonato = valor;

}
```

y llamarlo

```
Argentina.setUltimoCampeonato(2021)
```

c.

```
Argentina.ultimoCampeonato = 2021
```

d.

```
Argentina temp = new Argentina();
temp.ultimoCampeonato = 2021
```

Retroalimentación

La respuesta correcta es: Debería existir un método en la clase Argentina:

```
public static void setUltimoCampeonato(int valor) {
    ultimoCampeonato = valor;


}
```

y llamarlo
Argentina.setUltimoCampeonato(2021)

Pregunta **7**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

¿Existe algún mecanismo en Java para acceder a un método de una clase sin crear un objeto de dicha clase?

a.

No

b.

Si

Retroalimentación

La respuesta correcta es:

Si

Pregunta **8**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dadas las siguientes clases A y B

```
public final class A {

   int valor;

   public A(int valor) {
```

```
        this.valor = valor;

    }



    public int getValor() {

        return valor;

    }

}



public class B extends A{

    public B(int valor) {

        super(valor*2);

    }



    public int getValor() {
```

```
        return super.getValor()*2;

    }

}
```

¿Qué imprime el siguiente código?

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

    A b1 = (A) new B(12);

    System.out.println(b1.getValor());

}
```

a.

36

b.

48

c.

No imprime nada, hay un error en tiempo de compilación en la clase B

d.

No imprime nada hay un error en tiempo de ejecución

e.

No imprime nada, hay un error en tiempo de compilación en el main

f.

No imprime nada, hay un error en tiempo de compilación en la clase A

g.

24

h.

6

Retroalimentación

La respuesta correcta es:

No imprime nada, hay un error en tiempo de compilación en la clase B

Pregunta **9**

Correcta

Se puntúa 1,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dado el siguiente código

```
public class Alumno {

   private int cantidadAlumnos;

   private String nombre;

   private int legajo;



   public Alumno(String nombre) {

       this.nombre = nombre;

       legajo = cantidadAlumnos;

       cantidadAlumnos++;

   }
```

```
    public int getLegajo() {

        return legajo;

    }



    public void setLegajo(int legajo) {

        this.legajo = legajo;

    }

}
```

Indique qué imprime por consola el siguiente main

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

    Alumno a1 = new Alumno("Marcelo");

    Alumno a2 = new Alumno("Luis");

    Alumno a3 = new Alumno("Ariel");
```

```
    System.out.println(a3.getLegajo());


}
```

Respuesta:

Retroalimentación

La respuesta correcta es: 0

Pregunta **10**

Parcialmente correcta

Se puntúa 1,00 sobre 3,00

Marcar pregunta

Enunciado de la pregunta

Considere el siguiente código:

```
public abstract class Empleado {

    String nombre;



    public Empleado(String nombre) {
```

```
        this.nombre = nombre;
    }

    public abstract double getSueldo();

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public String toString(){
        return getNombre()+"-"+getSueldo();
    }
}

public class EmpleadoSueldo extends Empleado{
    private double sueldo;

    public EmpleadoSueldo(double sueldo) {
```

```
        super("SueldoFijo");

        this.sueldo = sueldo;

    }



    public double getSueldo() {

        return sueldo;

    }

}

public class EmpleadoComision extends Empleado{

    private double totalVentas;

    private double comision;



    public EmpleadoComision(double totalVentas, double
comision) {

        super("Comision");

        this.totalVentas = totalVentas;

        this.comision = comision;

    }



    public double getSueldo(){

        return totalVentas * comision;

    }
```

```
}
```

Dado el main que se muestra a continuación, seleccione de la lista desplegable el código faltante para que, como resultado, el programa imprima "Comision-349.0"?

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

   EmpleadoSueldo miEmpleado = new EmpleadoSueldo(349.0);
```

Respuesta

```
myEmpleado.setNombre("Comision");myEmpleado = new
EmpleadoComision(349.0, 1);myEmpleado = new
Empleado("Comision - 349");
```

```
   System.out.println(miEmpleado);

}
```

Dado el siguiente main, seleccione de la lista desplegable el código faltante para que, como resultado, el programa imprima "SueldoFijo-200.0"?

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

   Empleado porComision = null;
```

Respuesta

```
porComision.setNombre("SueldoFijo-200");porComision = new
EmpleadoSueldo(200);porComision = new
EmpleadoComision(200, 1);porComision = "SueldoFijo-349";
```

```
   System.out.println(porComision);

}
```

Si incorporamos una nueva clase de empleado que cobra su sueldo en base a las horas extra mediante el siguiente código Java:

```
public class EmpleadoHorasExtra {

   private double cantidadHoras;

   private double sueldoHora;



   public EmpleadoHorasExtra(double cantidadHoras, double
sueldoHora) {

       this.cantidadHoras = cantidadHoras;

       this.sueldoHora = sueldoHora;

   }



   public double getSueldo(){

       return cantidadHoras*sueldoHora;

   }
```

```
}
```

¿qué sucede con la ejecución del siguiente método main?

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

   Empleado porHoras = new EmpleadoHorasExtra(10,25);

   System.out.println(porHoras.getSueldo());

}
```

Imprime por pantalla "null-250"

Imprime por pantalla "PorHoras-250"

Genera un error de Null Pointer Exception

No compila

Imprime por pantalla "250"

La respuesta correcta es: No compila

Pregunta **11**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dado el siguiente enunciado:

Una persona asiste a un congreso si TODOS los temas que considera importantes (por ejemplo "inteligencia artificial", "programación", "java", etc.), son temáticas características del congreso.

considere el siguiente código Java

```
public class Asistente {

   private String nombre;

   private ArrayList<String> intereses;



   public Asistente(String nombre) {

       this.nombre = nombre;

       this.intereses = new ArrayList<>();

   }
```

```
   public ArrayList<String> getIntereses() {

       return new ArrayList<>(intereses);

   }

   public boolean cumple(Congreso congreso){

       int i = 0;

       while (i<intereses.size() &&
congreso.tieneCaracteristica(intereses.get(i)))

               i++;

       return i==intereses.size();

   }

}

public class Congreso {
```

```
   private ArrayList<String> caracteristicas = new
ArrayList<>();









   public boolean tieneCaracteristica(String
caracteristica){



       return caracteristicas.contains(caracteristica);



   }









   public boolean cumple(Asistente asistente){



       ArrayList<String> caractAsistente =
asistente.getIntereses();



       int i=0;



       while (i<caractAsistente.size() &&
tieneCaracteristica(caractAsistente.get(i)))



           i++;



       return i==caractAsistente.size();



   }
```

```
}
```

¿Cuál de los siguientes métodos cree que delega correctamente las responsabilidades?

a.

public boolean cumple(Congreso congreso) de la clase Asistente

b.

public boolean cumple(Asistente asistente) de la clase Congreso

c.

Ambos métodos delegan correctamente las responsabilidades

d.

Ninguno de los metodos delega correctamente las responsabilidades

Retroalimentación

La respuesta correcta es: public boolean cumple(Congreso congreso) de la clase Asistente

Pregunta **12**

Incorrecta

Se puntúa 0,00 sobre 1,00

Marcar pregunta

Enunciado de la pregunta

Dado el siguiente enunciado:

Una prestigiosa academia de magos esta organizando a sus Magos. Algunos magos tienen un "poder de magia" total definido. Existen otros magos peculiares cuyo poder de magia depende de la letra R y del día de la semana. Por ejemplo, si el día contiene una R el poder del mago merma un 50%, por lo tanto, los lunes su poder no merma, pero los martes si.

Considere la siguiente solución al problema

```
public abstract class Mago {

   private String nombreMagico;



   public Mago(String nombreMagico) {

       this.nombreMagico = nombreMagico;

   }
```

```
    public abstract double getPoderMagico();




    public String getNombre(){

        return nombreMagico;

    }

}

public class MagoSupremo extends Mago{

    private double poderMagico;




    public MagoSupremo(String nombre, double poder) {

        super(nombre);

        this.poderMagico = poder;

    }
```

```
   public double getPoderMagico() {

       return poderMagico;

   }

}

public abstract class MagoPeculiar extends Mago{

   private double poderMagico;

   private double merma;

   private String letra;



   public MagoPeculiar(String nombre, double poder, double
merma, String letra) {

       super(nombre);

       this.poderMagico = poder;

       this.merma = merma;

       this.letra = letra;
```

```
    }



    public double getPoderMagico() {

        LocalDate hoy = LocalDate.now();

        String nombreDia = hoy.getDayOfWeek().getDisplayName(

                TextStyle.FULL, Locale.getDefault()

        );

        if (nombreDia.contains(letra))

            return poderMagico - poderMagico*merma;

        else

            return poderMagico;

    }

}
```

Nota: LocalDate y DayOfWeek se utilizan solo para obtener el nombre del día de la semana y no hay errores en el uso de estas clases

Indique cual de las siguientes afirmaciones es correcta

NOTA: se puede seleccionar más de una, y las respuestas incorrectas RESTAN puntaje

a.

La clase Mago por ser abstracta no deberia tener constructor

b.

El código está perfecto

c.

Se debería abstraer en la clase Mago el atributo poderMagico. La clase mago Supremo no es más necesaria y la clase Mago debería ser concreta

d.

La clase MagoPeculiar deberia ser concreta

e.

EL código no compila

f.

Se deberia abstraer en la clase Mago el atributo poderMagico. El atributo deberia ser Protected para que pueda ser usado por los hijos MagoSupremo y Mago Peculiar. La clase Mago debe seguir siendo abstracta.

g.

Falla en tiempo de ejecución porque la clase MagoPeculiar es abstracta y nadie heredo de ella

Retroalimentación

Las respuestas correctas son:

Se debería abstraer en la clase Mago el atributo poderMagico. La clase mago Supremo no es más necesaria y la clase Mago debería ser concreta,

La clase MagoPeculiar deberia ser concreta