Pregunta **1**
Sin responder aún
Puntúa como 0,39
Marcar pregunta

Indica cuál de las siguientes afirmaciones es falsa con respecto al uso de un objeto de tipo IMocksControl:

Seleccione una:

- no permite el uso de un partial mock
- permite chequear el orden de invocaciones entre varios dobles
- no permite el uso de verify()
- Dejo la pregunta en blanco
- permite chequear el orden de las invocaciones de un doble

Pregunta **2**
Sin responder aún
Puntúa como 0,39
Marcar pregunta

Al ejecutar una clase de test que contiene n drivers:

Seleccione una:

- el método anotado con @BeforeEach se ejecuta n-1 veces
- Dejo la pregunta en blanco
- el método anotado con @AfterAll se ejecuta n veces
- el método anotado con @BeforeAll se ejecuta una vez
- el método anotado con @AfterEach se ejecuta n+1 veces

Siguiente página

Pregunta **4**
Sin responder aún
Puntúa como 0,39
Marcar pregunta

De los siguientes comandos de maven, ¿cuáles no ejecutarán los tests unitarios?

Comando 1: mvn clean surefire:test
Comando 2: mvn clean test
Comando 3: mvn test
Comando 4: mvn clean compile surefire:test
Comando 5: mvn clean test-compile surefire:test

Seleccione una:

- los comandos 2 y 3
- los comandos 1, 2, 4 y 5
- los comandos 1, 4 y 5
- Dejo la pregunta en blanco
- los comandos 1 y 4

Siguiente página

Pregunta **8**
Sin responder aún
Puntúa como 0,39
Marcar pregunta

Si al aplicar el método de caja negra de particiones equivalentes, obtenemos las siguientes particiones de entrada, válidas y no válidas, teniendo en cuenta la siguiente codificación para identificar las particiones: 'E' denota entrada; 'V' denota válida; 'nV' denota no válida:

Entrada 1: E1V1, E1nV1
Entrada 2: E2V1, E2nV1, E2nV2

Indica cuál es la cardinalidad del conjunto de casos de prueba eficiente y efectivo obtenido al aplicar dicho método

Seleccione una:

- 4
- Dejo la pregunta en blanco
- 5
- No se puede obtener si no se conocen las particiones de salida válidas y no válidas
- 3

Siguiente página

Pregunta **9**

Sin responder aún

Puntúa como 0,39

Marcar pregunta

Con el método del camino básico de McCabe:

Seleccione una:

- debemos elegir el conjunto mínimo de caminos para conseguir que todas las sentencias se ejecuten al menos una vez en cada caso de prueba
- Dejo la pregunta en blanco
- todas las afirmaciones del resto de opciones son falsas
- debemos elegir el conjunto mínimo de caminos para conseguir ejecutar todas las condiciones al menos una vez en cada caso de prueba
- debemos elegir todos los caminos del grafo

Siguiente página

Para realizar pruebas de una SUT que contiene dependencias externas,

Seleccione una:

- usaremos una verificación basada en el estado para pruebas unitarias, y una verificación basada en el comportamiento para pruebas de integración
- todas las afirmaciones del resto de opciones son falsas
- el primer paso es identificar los seams que contiene la SUT
- no está permitido modificar la SUT de ninguna forma para realizar pruebas unitarias
- Dejo la pregunta en blanco

Para realizar pruebas unitarias utilizando verificación basada en el comportamiento de la SUT calculaPrecio(), indica los tipos y número de mocks necesarios:

```
public class AlquilaCoches {
  protected Calendario calendario = new Calendario();

  public Ticket calculaPrecio(TipoCoche tipo, LocalDate inicio, int ndias)
                                                    throws MensajeException {
    Ticket ticket = new Ticket();
    float precioDia,precioTotal =0.0f;
    float porcentaje = 0.25f;

   String observaciones = "";
   IService servicio = new Servicio();
   precioDia = servicio.consultaPrecio(tipo);
   for (int i=0; i<ndias;i++) {
      LocalDate otroDia = inicio.plusDays((long)i);
      try {
         if (calendario.es_festivo(otroDia)) {
           precioTotal += (1+ porcentaje)*precioDia;
         } else {
           precioTotal += (1- porcentaje)*precioDia;
         }
      } catch (CalendarioException ex) {
          observaciones += "Error en dia: "+otroDia+"; ";
      }
   }

   if (observaciones.length()>0) {
      throw new MensajeException(observaciones);
   }

   ticket.setPrecio_final(precioTotal);
   return ticket;
  }
}
```

```
public class Ticket {
  private float precio_final;
  //getters y setters
}
```

Seleccione una:

- ○ un partialMock y dos StrictMock
- ○ un StrictControl, un PartialMock y un Mock
- ○ un StrictControl, un PartialMock y dos Mock
- ○ Dejo la pregunta en blanco
- ✓ un StrictMock, un PartialMock y un Mock

Indica la línea o líneas en las que tenemos puntos de inyección de seams para la SUT calculaConsumo():

```
1.//paquete ppss.ejercicio2
2.public class GestorLlamadas {
3.   static double TARIFA_NOCTURNA=10.5;
4.   static double TARIFA_DIURNA=20.8;
5.
6.   public Calendario getCalendario() {
7.     Calendario c = new Calendario();
8.     return c;
9.   }
10.
11.  public double calculaConsumo(int minutos) {
12.    Calendario c = getCalendario();
13.    int hora = c.getHoraActual();
14.    if(hora < 8 || hora > 20) {
15.      return minutos * TARIFA_NOCTURNA;
16.    } else {
17.       return minutos * TARIFA_DIURNA;
18.    }
19.  }
20.}
```

Seleccione una:

- ✔ las líneas 12 y 13
- ○ la línea 6
- ○ Dejo la pregunta en blanco
- ○ no hay ningún punto de inyección
- ○ la línea 7

Pregunta **20**

Sin responder aún

Puntúa como 0,39

Marcar pregunta

Utilizando el método de caja negra de particiones equivalentes, si tenemos una entrada asociada a un tipo enumerado con 3 valores, indica cuál de las siguientes afirmaciones es falsa:

Seleccione una:

- ○ Dejo la pregunta en blanco
- ○ podemos tener una sola partición válida de dicha entrada
- ○ podemos tener tres particiones válidas de dicha entrada
- ✔ podemos tener tres particiones no válidas de dicha entrada
- ○ podemos tener dos particiones válidas de dicha entrada

Siguiente página

y tiene la siguiente clase para los tests con 3 drivers:

```
class MiClaseTest {
      @Tag("misTests")
      @Test void test1() {
            // aquí vendría el código del test
       }

      @Tag("otroTest")
      @Test void test2() {
            // aquí vendría el código del test
       }

      @Test void test3() {
            // aquí vendría el código del test
       }
}
```

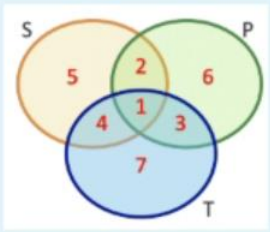
Si desde línea de comandos ejecutamos la orden

```
mnv test
```

Seleccione una:

- se ejecutan los 3 drivers
- se ejecuta sólo test3()
- Dejo la pregunta en blanco
- se ejecuta sólo test1()
- no se ejecuta ningún driver porque en la orden no se indica ninguna etiqueta

Dado el siguiente diagrama de Venn que hemos trabajado en clase:



Indica cuál de las siguientes afirmaciones es cierta:

Seleccione una:

- Dejo la pregunta en blanco
- Un tester debe intentar que el subconjunto 7 sea lo más grande posible
- Con métodos de caja negra y de caja blanca, se pueden alcanzar comportamientos de los subconjuntos 1 y 2
- Con un método de caja negra se pueden alcanzar comportamientos del subconjunto 3
- Con un método de caja blanca se pueden alcanzar comportamientos del subconjunto 4