EXAMEN INGENIERÍA DEL SOFTWARE 2 (27-NOVIEMBRE-2014) DCCIA

Apellidos y Nombre: _____

1. Explica la diferencia entre una planificación adaptativa y otra predictiva y pon ejemplos concretos que ilustren tu respuesta. ¿Qué tipo de planificación hemos utilizado en prácticas? Justifica claramente por qué e indica sus ventajas e inconvenientes (1,5 p)

2. En una aplicación de Juegos a través de internet, tenemos una clase Proxy que se encarga de la comunicación de los juegos entre los clientes y un servidor. Queremos probar el método Proxy.enviaMensaje():

```
void enviaMensaje(Mensaje msg, int idJuego, String urlServidor)
            throws MalformedURLException, MensajeException, JuegoInvalidoException;
```

   El tipo mensaje tiene 3 campos: tipo, destinatario y datos. El tipo puede ser TipoMensaje.DATOS, o TipoMensaje.BROADCAST (TipoMensaje es una enumeración, por lo que cualquier tipo diferente a los mencionados produciría un error de compilación). Tanto en los mensajes de tipo DATOS como de tipo BROADCAST es obligatorio proporcionar en el campo datos un array de bytes diferente de null, con los datos que se le vayan a enviar a uno de los jugadores en el primer caso, o a todos ellos en el segundo. De no ser así se produciría una excepción de tipo MensajeException. De la misma forma, en el caso del tipo DATOS será obligatorio indicar el destinatario del mensaje (que es una cadena con el identificador del jugador al que va dirigido), y si no se hace, o el jugador indicado no se encuentra dentro de la partida, también se lanzará un MensajeException. Antes de enviar el mensaje, el método comprueba el estado del juego (si el juego sigue activo continuará con el envío del mensaje, si el juego no está activo en el servidor, se devolverá la excepción de tipo JuegoInvalidoException).
   Si la url del juego no está bien formada se producirá la excepción MalformedURLException.
   El método no envía el mensaje si se produce alguna excepción. Enviará el mensaje en cualquier otro caso.
   Dada la especificación anterior, indica el número de casos de prueba eficientes y efectivos para probar el método Proxy,enviaMensaje() utilizando el método que creas más conveniente de los vistos en clase. Justifica tu respuesta y explica los pasos que realizas, e indica claramente por qué dicho conjunto es efectivo y eficiente. (2,5p)

3. Dado el siguiente código, estamos interesados en probar el método WebClient.getContent(), de forma que no utilicemos una conexión de red real. Implementa un test JUnit utilizando **verificación basada en el comportamiento** en el que el método getContent() devuelva la cadena de caracteres "Funciona" cuando accedemos a la url: http://examen.ua.es. Implementa cualquier artefacto adicional que necesites para poder ejecutar el test, justificando claramente la necesidad de dichos artefactos. Es importante que expliques claramente los pasos que vas siguiendo. (3p)

```
1. public class WebClient {
2.   public String getContent(URL url) {
3.     StringBuilder content = new StringBuilder();
4.     try {
5.        HttpURLConnection connection = url.openConnection;
6.        InputStream is = connection.getInputStream();
7.        byte[] buffer = new byte[2048];
8.        int count;
9.        while (-1 != (count = is.read(buffer))) {
10.            content.append(new String(buffer, 0, count));
11.        }
12.    }
13.      catch (IOException e) {
14.       return null;
15.    }
16.    return content.toString();
17.  }
18.}
```

4. Explica la diferencia entre un *target* y un *commitment*. Pon ejemplos concretos de *targets* y *commitments*. Indica si en prácticas hemos hecho uso de *targets* y/o *commitments.* Justifica claramente tu respuesta con ejemplos concretos. Indica en qué deberían basarse tanto los *targets* como los *commitments* y justifica claramente por qué (1,5 p)

5. Indica si son ciertas o falsas las siguientes afirmaciones y justifica claramente tu respuesta con ejemplos concretos:
   (a) Un 100% de cobertura de condiciones no garantiza un 100% de cobertura de líneas. (0,5p)
   (b) La velocidad es una métrica de seguimiento que nos permite calcular la duración restante de un proyecto. (0,5p)
   (c) Un servidor de integraciones continuas planifica las construcciones del sistema por nosotros, las 24 horas del día (0,5p)