DEPARTAMENTO
LENGUAJES Y CIENCIAS
DE LA COMPUTACIÓN

MANTENIMIENTO Y PRUEBAS DEL SOFTWARE
(Prueba realizada el 11 de Junio de 2015)

APELLIDOS

Nombre

3 ptos

1. El siguiente método muestra el máximo y la media de los números almacenados en un fichero de números.

```
public static void maxMedia(DataInputStream f)
    throws IOException {
  double suma = 0.0, max = 0.0, valor;
  int num = 0;
  boolean EOF = false;
  while (!EOF) {
    try {
      valor = f.readDouble();
      if (valor > max)
        max = valor;
      suma += valor;
      num++;
    } catch (EOFException e) {
      EOF = true;
    }
  }
  System.out.println("Max: " + max);
  System.out.println("Media: " + suma / num);
}
```

a) Define el mínimo número de casos de prueba para conseguir cobertura de instrucciones.
b) Define el mínimo número de casos de prueba para conseguir cobertura de ramas.
c) Utilizando la técnica basada en flujo de datos, proporciona un conjunto de caminos de prueba para alcanzar el 100% según el criterio de definiciones.
d) Define casos de prueba para los caminos de prueba del apartado anterior.
e) Implementa uno de los casos de prueba del apartado d) utilizando JUnit.
f) ¿Cuál es el principal beneficio de la técnica basada en el flujo de datos?

2. Considera la siguiente condición

```
if (X < 6) and (Y >= 19) or (Z = 3) then ...
```

1 ptos

a) ¿Cuántos caminos de prueba serían necesarios para conseguir 100% de cobertura de condiciones compuestas?
b) ¿Cuántos caminos de prueba serían necesarios para conseguir 100% de cobertura con el criterio MC/DC?
c) ¿Cuál es el principal beneficio del criterio MC/DC?

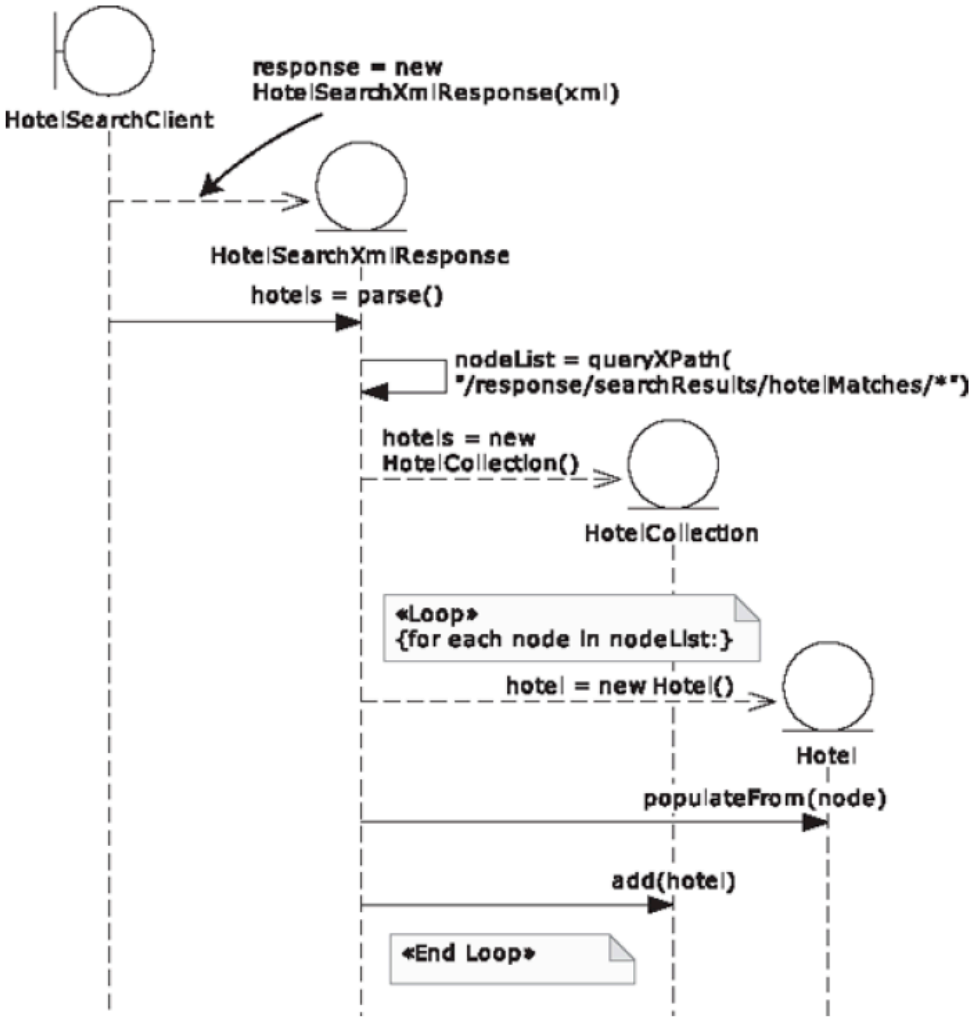
MapLet es un sistema web de localización que nos permite visualizar mapas, fotografías y vídeos de puntos de interés, así como realizar búsquedas de hoteles y actividades en los alrededores de estos puntos. Para hacer manejables los conjuntos de hoteles devueltos es posible filtrar los resultados utilizando diferentes criterios, como precios, facilidades que ofrece, etc.

2 ptos

3. Diseña 5 casos de prueba obtenidos mediante un análisis de parámetros de la función de localización. En este ejercicio nos centramos en la localización de hoteles, y, para simplificar, el filtrado utilizaremos como criterio de búsqueda la ciudad en la que se encuentra y que el precio esté dentro de un rango de valores.

2 ptos

4. La figura siguiente muestra un fragmento de un diagrama de secuencia del servicio de búsqueda de hoteles. El objeto HotelSearchClient se encarga de analizar el XML devuelto por el servicio de búsqueda de hoteles y crear una HotelCollection, para ello crea un objeto HotelSearchXmlResponse y comienza el proceso enviándole un mensaje parse().



a) Diseña un caso de prueba de integración a partir de este diagrama.
b) Implementa el caso de prueba del apartado a) utilizando JUnit+Mockito.
c) Este diagrama de secuencia tiene varias variantes, relacionadas con el XML pasado (por ejemplo, para distinto número de hoteles, o si el XML no es válido) y el número de hoteles en la colección de hoteles finalmente devuelta como resultado. ¿Cómo se complementarán los casos de prueba para estos casos?
d) Este fragmento del sistema depende de funciones de bibliotecas predefinidas (por ejemplo, queryXPath para recorrer el árbol resultante del análisis del XML recibido) y de servicios externos (por ejemplo, el

servicio que nos proporciona la lista de hoteles en una ciudad determinada). ¿Cómo manejarías bibliotecas y servicios externos en tus pruebas?

1.5 ptos 5. Diseña un caso de prueba a partir del siguiente caso de uso del sistema MapLet. Impleméntalo usando JUnit+FEST.

Caso de Uso: Búsqueda simple.
Descripción: Se buscan hoteles en una ciudad con un precio dentro de un rango.
Ámbito: Caso de uso primario.
Versión: 1.0.
Precondición: La ciudad X dispone de 3 hoteles con precios entre 100 y 120 euros la noche.
Secuencia de acciones:

1. El usuario escribe X en el campo del nombre de la ciudad a buscar.
2. En cuanto hay un nombre en el campo de la ciudad se activa el botón "buscar" para realizar la búsqueda.
3. El usuario escribe 100 y 120 como límites inferior y superior del rango de precios.
4. El usuario pulsa el botón "buscar".
5. El sistema muestra una lista de hoteles que satisfacen el criterio de búsqueda.

Postcondición: El sistema queda en la pantalla en la que se muestran los hoteles que satisfacen el criterio de búsqueda.

0.5 ptos 6. ¿Qué uso podríamos hacer del siguiente matcher?

```
private ArgumentMatcher actionEvent(final String str) {
  return new ArgumentMatcher() {
    @Override
      public boolean matches(Object argument) {
        return argument instanceof ActionEvent &&
          ((ActionEvent) argument).getActionCommand() == str;
      }
    };
  }
```