APELLIDOS:……

NOMBRE:…… ***TIEMPO: 60 MIN***

- Sitúe el DNI, o documento equivalente que permita su identificación, en lugar visible.
- Si lleva consigo un teléfono móvil o dispositivo electrónico, desconéctelo completamente.
- Ni es necesario ni se permite el uso de calculadora. Si lleva una consigo, guárdela.
- Debe entregar el examen completo.
- Para contestar a cada cuestión tipo test use esta hoja de respuestas (marque con una cruz la respuesta correcta). Cada cuestión presenta 3 opciones, y sólo 1 es correcta. Cada cuestión mal contestada resta media cuestión bien contestada, sobre el total del examen.
- Para contestar a cada cuestión de respuestas cortas use el hueco en esta hoja de respuestas.
- El examen consta de 20 cuestiones, todas igualmente ponderadas (0.5 puntos/cuestión).
- Se admiten preguntas, pero no se responden, si hay alguna errata se hará una aclaración en voz alta para todos los alumnos. La adecuada interpretación de las cuestiones del examen forma parte del mismo.

Test	1	2	3	4	5	6	7	8	9	10
a										
b										
c										

Respuesta corta	Hueco	Repuesta
11	h1	
12	h1	
	h2	
	h3	
13	h1	
	h2	
	h3	
	h4	
14	h1	
	h2	

1. Indica la forma de eliminar la cookie referenciada por la variable "cookie" antes de enviarla en la cabecera de respuesta correspondiente:

a. `cookie.setMaxAge(null);`
b. `cookie.setMaxAge(0);`
c. `cookie.delete();`
SOL: b

2. En una petición HTTP de tipo POST, para envío de datos de un formulario desde una página HTML, cuya acción dirige a un programa CGI, el valor de la variable "QUERY_STRING" pasada a dicho programa presenta el siguiente formato:

a. `nameX=valorX&nameY=valorY...`
b. `nameX=valorX?nameY=valorY...`
c. la variable QUERY_STRING está vacía.
SOL: c

3. Dentro de un servlet, una anotación permite:

a. Permite definir asociaciones de URLs con el servlet.
b. No permite definir parámetros de inicialización del servlet.
c. Permite definir parámetros de inicialización de la aplicación.
SOL: a

4. En un Servlet, para almacenar información de un atributo dentro del ámbito aplicación se emplea el siguiente método:

a. `request.getServletContext().setAttribute(String name, Object value)`
b. `application.setAttribute(String name, Object value)`
c. `request.getServletApplication().setAttribute(String name, Object value)`
SOL: a

5. Considere el siguiente trozo de código de un filtro que permite interceptar peticiones antes de llegar a los servlets:

```
...
@WebFilter(urlPatterns = { "*.asp" })
public class TestFilter implements Filter {
    @Override
    public void destroy() { }
    @Override
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
            throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        String url = request.getServletPath();
        if (url.endsWith(".asp")) {
            String nuevaUrl = url.substring(0, url.length() - 3) + "jsp";
            System.out.println("Nueva URL: "+nuevaUrl);
        } else {
            chain.doFilter(req, resp);
        }
    }
    @Override
    public void init(FilterConfig arg0) throws ServletException {
        ...
    }
}
```

El código del filtro:

a. Intercepta las páginas que contienen la extensión ".asp" en la URL y las reenvía a la nueva URL (misma página pero terminada en "jsp" en lugar de "asp").

b. Intercepta las páginas que contienen la extensión ".asp" en la URL e imprime la URL cambiada por la salida estándar.

c. No es correcto, en la llamada a chain.doFilter falta el argumento de la URL a donde se reenvía la petición.

SOL: b

6. Considere el código de la siguiente página JSP

```
<html><body>
  <%! String cad="a"; %>
  <% for( int i=1; i < 2; i++ ) {
       cad = cad + "b";
   } %>
   <p>Texto: <%= cad + "c"%></p>
</body></html>
```

Si la página ha sido llamada dos veces, la salida en el navegador en la segunda llamada es:

a. Texto: abc

b. Texto: abbc

c. Texto: abbbbc

SOL: b

7. Para incluir recursos dentro una página JSP de forma dinámica y con paso de parámetros, se emplea:

a. la directiva "include": `<%@ include ... %>`

b. la etiqueta de acción "include": `<jsp:include ... />`

c. la directiva "page": `<%@ page import=... %>`

SOL: b

8. Indique cuál de las siguientes sentencias de Expression Language es equivalente a la ejecución "request.getQueryString()" para acceder a la cadena de petición:

a. `${pageContext.request.queryString}`

b. `${requestScope.queryString}`

c. `${pageContext.queryString}`

SOL: a

9. Dentro del intérprete de comandos psql, la sentencia `dit=> \copy Z FROM 'z.txt'` realiza lo siguiente:

a. copia los datos de la tabla 'dit' desde el fichero z.txt (Z = compresión zip)

b. rellena la tabla Z con los datos almacenados en el fichero z.txt

c. ejecuta las órdenes SQL del fichero 'z.txt' y el resultado se almacena en Z

SOL: b

10. A la vista de estos dos bloques de código, que realizan una función similar, indique la opción verdadera:

*/********** Bloque A **********/*

```
Connection conn = ds.getConnection();
Statement sentencia = conn.createStatement();
String sql = "SELECT * FROM p WHERE ciudad='"+request.getParameter("ciudad")+"'";
ResultSet rs = sentencia.executeQuery(sql);
```

```
/************** Bloque B **************/
Connection conn = ds.getConnection();
String sql = "SELECT * FROM p WHERE ciudad=?";
PreparedStatement sentencia = conn.prepareStatement(sql);
sentencia.setString(1, request.getParameter("ciudad"));
ResultSet rs = sentencia.executeQuery();
```

a. en el segundo bloque es necesario añadir la variable 'sql' como argumento del método 'executeQuery'.
b. el segundo bloque permite evitar la inyección de SQL.
c. el método 'setString' del segundo bloque debe contener solo un argumento (sobra el '1').
SOL: b

11. Rellene el hueco correspondiente para asignar, en una sola sentencia, todas las propiedades del bean cuyos nombres coincidan con los nombres de los parámetros de la petición HTTP:

```
...
<jsp:useBean id="client" class="paquete.Cliente" />
<jsp:setProperty property=______h1______ name="client"/>
...
```
SOL: "*"

12. Considere el siguiente trozo de código de una página JSP que hace uso del JavaBean "BeanExamen":

```
<%@page import="fast.BeanExamen"%>
...
<jsp:useBean id="bean" class="fast.BeanExamen" />
<jsp:setProperty name="bean" property="propiedad" value="Examen"/>
...
```
Escriba el código correspondiente al scriptlet que realiza la misma funcionalidad (sin etiquetas de acción de JavaBean y suponiendo que no existe ningún atributo):

```
<%@page import="fast.BeanExamen"%>
...
<%
_______h1______ bean = new _____h2_______;
bean._____h3_____("Examen");
pageContext.setAttribute("bean", bean);
%>
...
```
SOL:
BeanExamen bean = new BeanExamen();
bean.setPropiedad("Examen");

13. Rellene los huecos para que en el navegador se muestre el incremento del número de visitas cada vez que se recarga la página:

```
<%
   Integer visitCount = new Integer(0);
   String visitCountKey = "visitCount";
      if (session.isNew()){
      session._______h1________(visitCountKey,  visitCount);
   }
   visitCount = (Integer)session._______h2_______(_______h3______);
   visitCount = visitCount + 1;
   session._______h4________(visitCountKey,  visitCount);
%>
<html>
```

```
<body>
Numero de visitas: <%=visitCount %>
</body>
</html>
```

SOL:

```
<%
   Integer visitCount = new Integer(0);
   String visitCountKey = new String("visitCount");
   if (session.isNew()){
      session.setAttribute(visitCountKey,  visitCount);
   }
   visitCount = (Integer)session.getAttribute(visitCountKey);
   visitCount = visitCount + 1;
   session.setAttribute(visitCountKey,  visitCount);
%>
<html>
<body>
Numero de visitas: <%=visitCount %>
</body>
</html>
```

14. Tenga en cuenta los siguientes datos almacenados en la tabla de nombre T:

id	nombre	ciudad	edad
1	Rosa	Sevilla	25
2	Pedro	Madrid	31
3	Celia	Sevilla	28
4	Juan	Badajoz	45

Indique la sentencia SQL que obtiene el nombre (y solo el nombre) de las personas que viven en Sevilla y tienen más de 30 años:

_____h1_____ WHERE _____h2_____;

SOL: SELECT nombre FROM T WHERE ciudad = 'Sevilla' AND edad > 30;