## RECOPILACIÓN PREGUNTAS UNIDAD 8 (desde 2014)

**[Examen final 2º bloque junio 2020 --> examen online]**

**Tema 8: generales comunicación + Sobre ROI y RMI Batería 20 preg --> selecc 8**

1. En la comunicación indirecta, el nodo emisor envía los mensajes a un intermediario que más tarde los enviará al nodo destino. \*Verdadera
2. Cuando se utilizan mecanismos de comunicación como invocaciones a objetos remotos, el contenido de los mensajes que envía el middleware es normalmente responsabilidad del programador de la aplicación. \*Falso
3. En la comunicación asincrónica, el emisor se bloquea hasta ser avisado de que el receptor ha recibido de forma correcta el mensaje. \*Falso
4. En algún tipo de comunicación sincrónica, el emisor se bloquea hasta ser avisado de que el receptor ha procesado el mensaje. \*Verdadero
5. En algún tipo de comunicación sincrónica, el emisor se bloquea hasta ser avisado de que el receptor ha recibido de forma correcta el mensaje. \*Verdadero
6. Cuando se emplean sockets se utilizan operaciones básicas de envío y recepción.

\*Verdadero

1. XML y JSON se utilizan para representar en binario el contenido de los mensajes

\*Falso

1. Se utiliza direccionamiento directo cuando los mensajes se envían a un bróker.

\*Falso

1. En la comunicación no persistente es necesario que el receptor esté activo cuando el emisor envía el mensaje para que éste se transmita. \*Verdadero
2. Una invocación remota tiene el mismo comportamiento que una invocación local, pero una sintaxis distinta. \*Falso
3. A diferencia de Java RMI, ROI no utiliza un servidor de nombres. \*Falso
4. En ROI, cuando se crea un objeto remoto el sistema instala el esqueleto correspondiente.\*Verdadero
5. En ROI, cuando un cliente referencia por vez primera a un objeto remoto, obtiene un proxy para dicho objeto. \*Verdadero
6. Java RMI ofrece persistencia. \*Falso
7. El proxy es el único elemento encargado de la serialización (empaquetado). \*Falso
8. Cuando pasamos un objeto por referencia, el receptor desempaqueta el mensaje y crea una copia del objeto (ambos objetos son independientes). \*Falso
9. En ROI, cuando un proceso servidor crea un objeto por propia iniciativa, puede registrarlo en el servidor de nombres. \*Verdadero

**Tema 8 Sobre REST + JMS Batería 12 preg** => 6 preguntas

1. En REST, si se solicita la siguiente operación: DELETE htpp://[www.myserver.com/alumnos/2233/trabajos,](http://www.myserver.com/alumnos/2233/trabajos) en caso de que sea un recurso que existe y si todo funciona correctamente, se eliminarán todos los trabajos del alumno con identificador 2233 y el servidor responderá con el código de estado HTTP 200, el cual indica que la solicitud es correcta. \*Verdadero
2. Para representar la información intercambiada en los mensajes, REST utiliza un formato binario transparente al programador. \*Falso
3. En REST, las URIs empleadas hacen referencia solo al ordenador que implementa el servicio. Otras informaciones, tales como el recurso concreto al que se quiere acceder se deben indicar siempre como datos codificados en XML o JSON. \*Falso



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

1. JMS ofrece una comunicación fuertemente acoplada, en la que emisor y receptor deben crear un enlace permanente para poder empezar a comunicarse entre sí.\*Falso
2. JMS se considera persistente, por lo que no resulta necesario que el receptor esté activo en el momento en que se envía un mensaje. \*Verdadero
3. En JMS, los objetos que implementan la interfaz Queue se crean llamando a métodos de la interfaz JMSConsumer. \*Falso
4. REST utiliza distintos tipos de mensajes http pasa asociar operaciones básicas de acceso, creación, actualización y borrado a solicitudes sobre recursos. \*Verdadero
5. JMS requiere un elevado acoplamiento entre los componentes de la aplicación distribuida. \*Falso
6. Los proveedores, los mensajes y los objetos administrados son componentes de JMS. \*Verdadero
7. REST sigue un modelo de petición/respuesta. \*Verdadero

# [Examen 2º parcial junio 2020 --> examen online]

## Tema 8: generales comunicación + Sobre ROI y RMI Batería 20 preg -> Sel 6

1. En la comunicación síncrona (en respuesta) el middleware responde cuando el receptor procesa el mensaje. \*Verdadero
2. La estructura de los mensajes en base a cabecera y contenido es propia de Java RMI, donde la cabecera es un conjunto de campos, generalmente extensible. \*Falso
3. En la comunicación persistente, el emisor es el responsable de guardar los mensajes pendientes de entrega si el receptor no está en ejecución cuando se envía el mensaje. \*Falso
4. En la comunicación asincrónica, el emisor se bloquea hasta ser avisado de que el receptor ha recibido de forma correcta el mensaje. \*Falso
5. En la comunicación persistente es necesario que el receptor esté funcionando cuando el emisor envía el mensaje. \*Falso
6. Cuando se utiliza un middleware de mensajería que ofrece colas de mensajes, la comunicación suele ser persistente. \*Verdadero
7. Los sockets ofrecen al programador un mecanismo de comunicación en el que los mensajes están estructurados, diferenciando entre una cabecera con un conjunto de campos y el contenido. \*Falso
8. En JMS, cuando se emplean colas de mensajes el mensaje no está estructurado, solo tiene contenido. \*Falso
9. En las llamadas a métodos remotos, el envío de mensajes es transparente al programador. \*Verdadero
10. En ROI, el esqueleto del objeto remoto reside en el nodo del cliente. \*Falso
11. En ROI, en una invocación local, el proxy (stub cliente) y el esqueleto intercambian mensajes. \*Falso
12. En Java RMI, en una invocación sobre un objeto remoto, los argumentos que correspondan a objetos remotos se pasan por referencia. \*Verdadero
13. En ROI, el formato de los mensajes que intercambian proxy y esqueleto es transparente al programador. \*Verdadero
14. Java RMI utiliza comunicación sincrónica en entrega. \*Falso
15. RMI utiliza direccionamiento indirecto. \*Falso
16. En RMI una invocación desde un proceso a otro dentro del mismo nodo se considera una invocación remota. \*Verdadero
17. En ROI, el ORB identifica y localiza a los objetos, canaliza mensajes entre proxies y esqueletos, y gestiona el ciclo de vida de los objetos. \*Verdadero
18. En ROI, el proxy correspondiente a un objeto remoto se crea cuando se crea dicho objeto remoto. \*Falso
19. En ROI, un cliente puede buscar un objeto en el servidor de nombres y obtener una referencia al mismo, para lo cual necesita su nombre lógico. \*Verdadero

## Tema 8 Sobre REST + JMS. Batería 15 preg -> Sel 5

1. REST permite acceso a recursos, a los cuales accedemos mediante URIs estructuradas de forma jerárquica. \*Verdadero
2. REST ofrece persistencia.\*Falso
3. Los servicios web RESTFul son servicios con estado, es decir, se requiere un seguimiento de la sesión establecida con cada cliente.\*Falso
4. Los servicios REST están disponibles en la mayoría de lenguajes de programación y frameworks de desarrollo, ya que son un estándar estricto.\*Falso
5. En REST no se utilizan los códigos de estado HTTP, a pesar de que REST está basado en dicho protocolo.\*Falso
6. En REST, los recursos se manipulan utilizando un conjunto simple de operaciones HTTP, como PUT, GET, POST y DELETE.\*Verdadero
7. JMS se considera sincrónico, porque en el envío el proceso emisor permanece suspendido esperando respuesta del proceso receptor.\*Falso
8. JMS utiliza elementos intermedios (ej. colas y temas) entre los Clientes del sistema.

\*Verdadero

1. JMS proporciona comunicación sincrónica en entrega.\*Falso
2. En JMS, los objetos que implementan la interfaz JMSProducer se crean llamando a métodos de la interfaz JMSContext.\*Verdadero
3. Cuando un cliente JMS envía un mensaje a un destino de tipo cola, puede ser entregado a varios clientes JMS.\*Falso
4. Los componentes de JMS son solamente tres: los proveedores, los mensajes y los objetos administrados.\*Falso
5. En JMS, las factorías de conexiones se crean mediante las herramientas administrativas del proveedor JMS y se utilizan para crear las conexiones de los clientes al sistema de mensajería.\*Verdadero
6. En REST, las URIs pueden incluir parámetros para matizar la operación que se solicita, por ejemplo para paginar las respuestas.\*Verdadero
7. JMS sigue un modelo de petición/respuesta.\*Falso

# [FINAL 2019]

Sobre los mecanismos de comunicación en los sistemas distribuidos:

|  |  |
| --- | --- |
| 1. En la comunicación asíncrona, el middleware responde al emisor tras  recibir aviso del receptor de haber procesado el mensaje. | F |
| 2. JMS ofrece una comunicación fuertemente acoplada, ya que los clientes deben establecer conexión entre sí a través de las factorías de conexión. | F |
| 3. En JMS se emplea sincronización síncrona, de modo que el emisor debe esperar a que el receptor reciba el mensaje. | F |
| 4. El mecanismo de comunicación JMS resulta útil cuando los  componentes del sistema no necesitan recibir una respuesta inmediata a sus mensajes para poder continuar su funcionamiento. | V |
| 5. El mecanismo de comunicación ROI proporciona transparencia de ubicación. | V |
| 6. En ROI el emisor envía mensajes a una cola y el receptor recibe  mensajes de dicha cola, a la que se ha suscrito. | F |

Sobre los mecanismos de comunicación ROI y Java RMI:

|  |  |
| --- | --- |
| 7. Una vez el objeto remoto ha finalizado la ejecución del método invocado, el proxy empaqueta los argumentos de la llamada a dicho método. | F |
| 8. Java RMI es una API de comunicación especificada en múltiples lenguajes de programación, tales como Java, Javascript, C#, Python, etc. | F |
| 9. En una invocación remota (ROI), los objetos invocador e invocado residen siempre en procesos diferentes, independientemente de si los procesos están  ubicados en el mismo nodo o en nodos diferentes. | V |

# [2º parcial 2019]

Sobre los mecanismos de comunicación ROI y Java RMI:

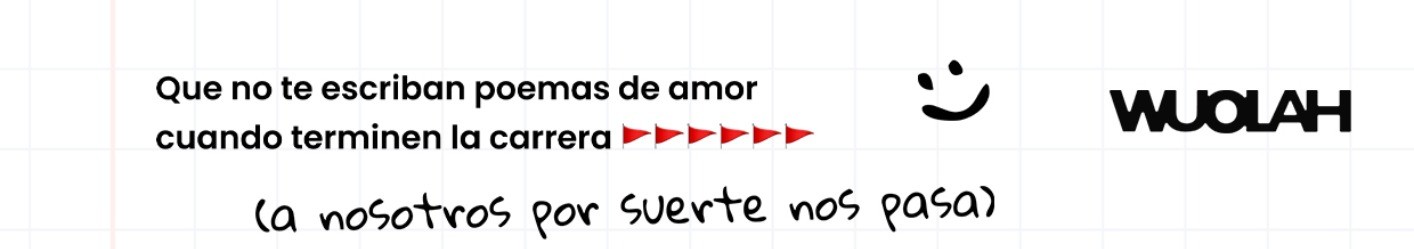
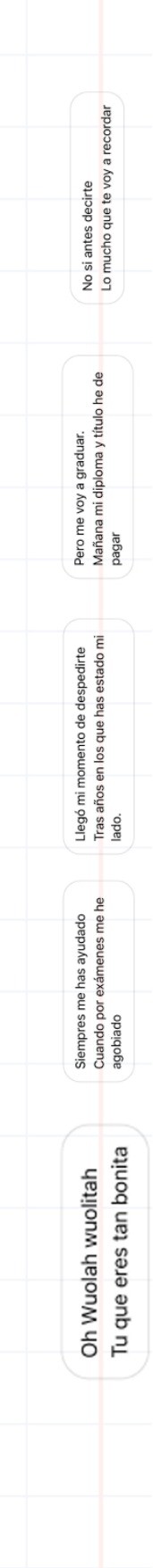
|  |  |
| --- | --- |
| 1. Se considera que una invocación a un objeto es una invocación local cuando los objetos invocador e invocado residen en procesos diferentes del mismo nodo. | F |
| 2. En ROI, para el paso de objetos como argumentos por valor, se utiliza la técnica de serialización de objetos para así transmitirlos al nodo destino  de la invocación. | V |
| 3. En Java RMI, un objeto invocable de forma remota debe implementar la interfaz java.rmi.Remote. | V |
| 4. Java RMI es un middleware de comunicación que proporciona  comunicación asincrónica. | F |
| 5. Java RMI proporciona la interfaz Registry para que tanto el cliente como el servidor interaccionen con un servidor de nombres usando  métodos como lookup, bind, rebind … | V |
| 6. El servidor de nombres de Java RMI almacena para cada objeto registrado, su proxy y su esqueleto. | F |

Sobre los mecanismos de comunicación en los sistemas distribuidos:

|  |  |
| --- | --- |
| 1. Por comunicación directa se entiende que un proceso se envía un  mensaje a sí mismo directamente, y por comunicación indirecta que se lo envía a otro proceso diferente. | F |
| 2. En el envío de un mensaje en un sistema con comunicación sincrónica en la respuesta, el emisor esperará hasta que el receptor haya procesado el mensaje y devuelva la respuesta. | V |

Respecto a Java Message Service:

|  |  |
| --- | --- |
| 3. Java JMS es una API Java que permite a las aplicaciones invocar a objetos remotos mediante el envío y recepción de mensajes. | F |
| 4. Los componentes de JMS son los proveedores, los clientes, los mensajes y los objetos administrados. | V |
| 5. Cuando se quiere que los componentes de una aplicación distribuida no | F |



|  |  |
| --- | --- |
| dependan de conocer las interfaces de otros componentes es preferible usar Java RMI frente a JMS. |  |
| 6. En JMS un mismo mensaje puede ser entregado a varios clientes. | V |
| 7. En JMS, en general, para que un cliente A pueda enviar un mensaje a otro cliente B, el cliente B debe estar activo. | F |

## [FINAL 2018]

Sobre el mecanismo de comunicación ROI:

|  |  |
| --- | --- |
| 1. El ORB se encarga de gestionar las referencias a los objetos remotos. | V |
| 2. El formato de los mensajes intercambiados entre cliente y servidor es transparente para el programador. | V |

Sobre el mecanismo de comunicación Java RMI:

|  |  |
| --- | --- |
| 3. Los objetos remotos son instancias de clases que deben implementar un interfaz que extienda Remote | V |
| 4. Java RMI es un middleware de comunicación orientado a objetos que ofrece direccionamiento  directo y persistencia. | F |

Respecto a los servicios web RESTful:

|  |  |
| --- | --- |
| 5. Los servicios web REST ofrecen comunicación no persistente y direccionamiento directo mediante peticiones al ordenador que alberga el servicio. | V |
| 6. En REST, en la solicitud, el tipo de mensaje http (post, get, put, delete) indica el tipo de operación  sobre el recurso (crear, leer, actualizar, borrar). | V |

Sobre Java Message Service:

|  |  |
| --- | --- |
| 7. JMS evita la necesidad de incluir en el código operaciones explícitas de envío/recepción | F |
| 8. En JMS, los destinos sirven para comunicar a Clientes JMS y Proveedores JMS entre sí. | F |

## [2º PARCIAL 2018]

Sobre los mecanismos de comunicación ROI y Java RMI:

|  |  |
| --- | --- |
| 1. Definimos objeto remoto como todo objeto que se define en un nodo pero únicamente puede invocarse desde otros nodos. | F |
| 2. ROI sigue un modelo petición/respuesta, y por lo tanto es sincrónico en respuesta. | V |
| 3. Los objetos remotos se pasan por valor, y los objetos locales por referencia. | F |
| 4. RMI es multilenguaje y multiplataforma. | F |

Sobre los mecanismos de comunicación, en general, y sobre los servicios web RESTful y Java Message Service en particular:

|  |  |
| --- | --- |
| 5. Los servicios web RESTful emplean códigos de estado HTTP en sus respuestas. | V |
| 6. REST proporciona comunicación persistente. | F |
| 7. El primer paso para utilizar JMS es lanzar un bróker que facilita las comunicaciones. | V |
| 8. Java Message Service es un middleware orientado a mensajería que ofrece una comunicación | V |



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

|  |  |
| --- | --- |
| débilmente acoplada. |  |
| 9. Los sistemas que no proporcionan persistencia requieren que emisor y receptor estén activos simultáneamente. | V |

# [FINAL 2017]

Sobre Java RMI:

|  |  |
| --- | --- |
| 9. Los objetos remotos deben implementar una interfaz que haya extendido la interfaz java.rmi.Remote. | V |
| 10. Proporciona comunicación sincrónica entre clientes y servidores. | V |
| 11. Permite el paso de argumentos a los métodos remotos únicamente por valor, mediante el procedimiento conocido como serialización. | F |
| 12. Proporciona un servicio de nombres, llamado "rmiregistry". | V |

Un esqueleto (“stub” servidor):

|  |  |
| --- | --- |
| 13. Permite a un único proxy conectarse al objeto remoto. De forma que necesitamos  tantos esqueletos como proxies tenga el objeto. | F |
| 14. Realiza la llamada al objeto destino de la invocación, de forma local. | V |
| 15. Se encarga de empaquetar los argumentos de salida a la hora de construir el  mensaje de respuesta hacia el cliente. | V |

Respecto a los servicios web RESTful:

|  |  |
| --- | --- |
| 16. Los servicios Web REST son un ejemplo de comunicación con direccionamiento indirecto. | F |
| 17. Para realizar cualquier llamada a un servicio web RESTful se requiere emplear el método HTTP GET, seguido de un objeto codificado en XML o JSON. | F |
| 18. En la respuesta a cualquier llamada a un servicio RESTful se requiere devolver un código de terminación basado en códigos de estado del protocolo HTTP. | V |

Sobre la comunicación basada en mensajes mediante JMS:

|  |  |
| --- | --- |
| 19. En JMS emisor y receptor no necesitan conocerse entre sí. | V |
| 20. En JMS las factorías de conexiones se emplean para crear directamente los mensajes que intercambiarán los clientes JMS. | F |
| 21. JMS proporciona un modelo de comunicación fundamentalmente sincrónico. | F |
| 22. Los objetos administrados son exclusivamente los temas y los mensajes. | F |
| 23. A diferencia de otros mecanismos, se requiere de un proceso adicional (bróker) que proporcione la comunicación. | V |
| 24. Si un proceso recibe con éxito un mensaje de una cola de mensajes, esto implica  necesariamente que el proceso que envía los mensajes a la cola está funcionando en ese preciso instante. | F |

# [2º PARCIAL 2017]

Sobre los mecanismos de comunicación en los sistemas distribuidos:

|  |  |
| --- | --- |
| 25. La estructura de los mensajes es la misma en todos los mecanismos de comunicación. | F |
| 26. Existen mecanismos de comunicación que se utilizan haciendo llamadas a  procedimientos o métodos remotos. | V |
| 27. En la comunicación asincrónica el middleware responde al emisor del mensaje  cuando el receptor ha confirmado la entrega correcta del mensaje. | F |

Sobre los mecanismos de comunicación ROI y Java RMI:

|  |  |
| --- | --- |
| 28. En ROI, el proxy es el encargado de realizar las verdaderas llamadas al objeto remoto. | F |
| 29. Cuando en ROI se pasa un objeto por referencia se crea un nuevo objeto en el ordenador destino copia del objeto original. | F |
| 30. Cuando se crea un objeto a iniciativa del cliente, éste tiene que registrar primero  dicho objeto en el servidor de nombres. | F |
| 31. En Java RMI, si el objeto que se pasa como argumento implementa la interfaz  Remote, se serializa y se pasa por valor. | F |
| 32. Java RMI puede considerarse un mecanismo de comunicación en el que se usa direccionamiento directo. | V |

Respecto a los servicios web RESTful:

|  |  |
| --- | --- |
| 33. Los servicios web RESTful suelen ser servicios sin estado, lo que implica que el  cliente debe suministrar toda la información necesaria en cada petición. | V |
| 34. Cuando se siguen las directrices de los servicios web RESTful no importa qué  método HTTP utilizar en una petición, puesto que el servidor nunca utiliza dicha información. | F |
| 35. Cuando se siguen las directrices de los servicios web RESTful, las URIs empleadas pueden contener argumentos para, por ejemplo, realizar consultas. | V |
| 36. RESTful se considera normalmente como un mecanismo de comunicación no persistente. | V |

Respecto a Java Message Service:

|  |  |
| --- | --- |
| 37. JMS es una API Java que permite a las aplicaciones invocar métodos de objetos remotos. | F |
| 38. Los principales componentes de JMS son los proveedores JMS, los clientes JMS, los mensajes y los objetos administrados: factorías de conexiones y destinos. | V |
| 39. En JMS los mensajes carecen completamente de estructura. | F |
| 40. La comunicación en JMS se considera normalmente como asincrónica. | V |

# [FINAL 2016]

Sobre los mecanismos de comunicación en sistemas distribuidos:

|  |  |  |
| --- | --- | --- |
| 1. | Los distintos tipos de direccionamiento pueden ser asincrónicos, sincrónicos en la  entrega o sincrónicos en la respuesta. | F |
| 2. | La invocación a objetos remotos (ROI) es un buen ejemplo de direccionamiento indirecto. | F |
| 3. | Los tres principales elementos que intervienen en una invocación a objeto remoto son el *proxy*, el esqueleto y el ORB. | V |

|  |  |  |
| --- | --- | --- |
| 4. | En Java RMI, cuando se envía un objeto serializado a un nodo remoto, el objeto enviado mantendrá un estado independiente del objeto original. | V |
| 5. | En ROI, para crear un objeto remoto a iniciativa del cliente, la factoría contesta con una referencia a un objeto creado expresamente para atender la solicitud del  cliente. | V |
| 6. | Los servicios Web RESTful son un estándar estricto definido en una API Java. | F |
| 7. | Para realizar una llamada a un servicio web RESTful se requiere emplear el método HTTP PUT, seguido de un objeto codificado en XML o JSON. | F |
| 8. | En Java Message Service, los principales objetos administrados son los clientes y los proveedores JMS. | F |
| 9. | Java Message Service es un buen ejemplo de comunicación persistente. | V |

# [2º PARCIAL 2016]

Sobre los mecanismos de comunicación en sistemas distribuidos:

|  |  |  |
| --- | --- | --- |
| 10. | Cuando se utilizan llamadas a objetos remotos, el programador no tiene que ocuparse de hacer llamadas a operaciones de envío y recepción de mensajes. | V |
| 11. | La persistencia es una característica que poseen aquellos middlewares de  comunicación que pueden guardar mensajes pendientes de entrega. | V |
| 12. | Los servicios Web RESTful generalmente están basados en una arquitectura cliente-  servidor implementada sobre el protocolo HTTP. | V |
| 13. | La respuesta a una llamada a un servicio RESTful incluye un código de terminación basado en códigos de estado del protocolo HTTP. | V |

Sobre el mecanismo de comunicación Java RMI:

|  |  |  |
| --- | --- | --- |
| 14. | El acceso al servidor de nombres donde se registran objetos remotos se realiza mediante la interfaz llamada *Registry*. | V |
| 15. | Los *proxies* a los objetos remotos sólo pueden crearse mediante el método  *createProxy* de la clase *java.rmi.server.UnicastRemoteObject*. | F |
| 16. | Los objetos que implementan la interfaz *java.rmi.Remote* siempre se pasan por  referencia en las invocaciones a métodos remotos. | V |

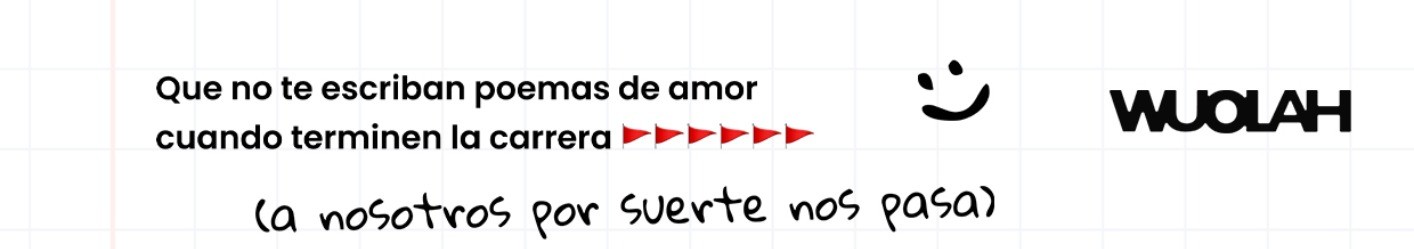
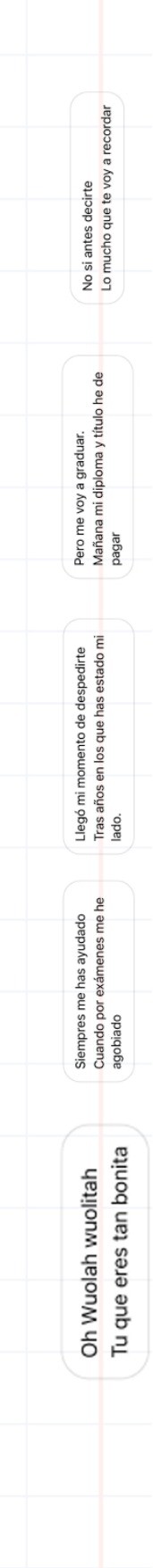
Respecto al mecanismo de comunicación Java Message Service:

|  |  |  |
| --- | --- | --- |
| 17. | JMS ofrece una comunicación fuertemente acoplada, en la que emisor y receptor deben crear un enlace permanente para poder empezar a comunicarse entre sí. | F |
| 18. | Resulta interesante utilizar JMS cuando se quiere que los componentes de una aplicación no dependan de conocer las interfaces de otros componentes. | V |
| 19. | Cuando se envía un mensaje a un destino de tipo tema, puede ser entregado a varios clientes JMS. | V |

# [FINAL 2015]

Sobre el mecanismo de comunicación ROI:

|  |  |  |
| --- | --- | --- |
| 20. | Se pueden pasar objetos remotos por referencia. | V |
| 21. | Cuando un cliente referencia por vez primera a un objeto remoto, obtiene el | F |



|  |  |  |
| --- | --- | --- |
|  | esqueleto para dicho objeto.  *JUSTIFICACIÓN: Obtiene el proxy.* |  |
| 22. | El esqueleto ofrece la misma interfaz que el objeto remoto.  *JUSTIFICACIÓN: El proxy es quien ofrece la misma interfaz que el objeto remoto.* | F |

Sobre el mecanismo de comunicación Java RMI:

|  |  |  |
| --- | --- | --- |
| 23. | Si un objeto que se pasa como argumento implementa la interfaz *Remote*, entonces  dicho objeto se pasa por referencia. | V |
| 24. | Para implementar un objeto remoto en Java RMI, la clase de los objetos remotos debe implementar la interfaz remota y extender *java.rmi.server.UnicastRemoteObject*, para así poder registrar los objetos en el ORB de Java. | V |
| 25. | El servidor de nombres de Java RMI guarda, para cada objeto remoto, su nombre simbólico y la dirección (host, puerto) de su proxy.  *JUSTIFICACIÓN: El servidor de nombre guarda el nombre simbólico y la referencia.* | F |
| 26. | El servidor de nombres de Java RMI puede residir en cualquier nodo (incluso en el nodo cliente) y es accedido usando la interfaz Registry. | V |
| 27. | Java RMI es un ejemplo de middleware de mensajería, donde cliente y servidor deben suscribirse al middleware para así poder enviarse mensajes entre sí.  *JUSTIFICACIÓN: Cliente y servidor no requieren subscripción al middleware. Esto se realiza en Java JMS.* | F |

Respecto a los servicios Web RESTful y al mecanismo de comunicación Java Message Service:

|  |  |  |
| --- | --- | --- |
| 28. | Resulta interesante utilizar servicios Web RESTful cuando no es necesario que todos los componentes de la aplicación estén simultáneamente en ejecución.  *JUSTIFICACIÓN: La comunicación no es persistente, por lo que los componentes deben estar*  *simultáneamente en ejecución.* | F |
| 29. | Los servicios Web RESTful utilizan métodos del protocolo http para indicar el tipo de operación. | V |
| 30. | En JMS, los mensajes no son estructurados y se envían en texto plano en XML.  *JUSTIFICACIÓN: Sí son estructurados, con cabecera, contenido y campos predefinidos.* | F |
| 31. | JMS generalmente utiliza comunicación indirecta. | V |
| 32. | Los objetos que implementan la interfaz JMSContext se crean a partir de una factoría de conexiones. | V |

## [2º PARCIAL 2015]

Sobre el mecanismo de comunicación ROI:

|  |  |  |
| --- | --- | --- |
| 33. | El componente denominado ORB se encarga, entre otras cosas, de identificar y localizar a los objetos remotos. | V |
| 34. | El middleware de un sistema con comunicación basada en ROI emplea generalmente  comunicación sincrónica no persistente y direccionamiento directo. | V |
| 35. | Cuando se pasa un objeto por valor, se copia una referencia al mismo.  JUSTIFICACIÓN: *Cuando se pasa un objeto por valor, el estado del objeto se empaqueta, mediante un*  *proceso denominado serialización.* | F |
| 36. | El proxy empaqueta los argumentos una vez recibe la contestación del esqueleto. | F |



Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

|  |  |  |
| --- | --- | --- |
|  | JUSTIFICACIÓN: *El proxy desempaqueta los argumentos al recibir la contestación, para así devolverlos al proceso cliente.* |  |

Sobre el mecanismo de comunicación Java RMI:

|  |  |  |
| --- | --- | --- |
| 37. | El proxy correspondiente a un objeto remoto se crea en tiempo de ejecución cuando se accede por primera vez al objeto remoto. | V |
| 38. | Se considera objeto local todo objeto que sólo puede invocarse desde la computadora en que se define (aunque sea desde otras máquinas virtuales Java), y objeto remoto a todo objeto que puede invocarse desde otras computadoras.  *JUSTIFICACIÓN: Si un objeto se invoca desde otras máquinas virtuales Java de la misma computadora,*  *también se considera como objeto remoto.* | F |
| 39. | No proporciona transparencia de ubicación, porque la sintaxis de invocación del método es distinta dependiendo de si el objeto es local o remoto.  *JUSTIFICACIÓN: La sintaxis de invocación es la misma. En la interfaz del objeto definimos los métodos*  *del objeto (con independencia de que vaya a ser remoto o local). Si el objeto es remoto, la interfaz debe extender java.rmi.Remote.* | F |
| 40. | La clase de un objeto remoto debe implementar un interfaz que extienda java.rmi.Remote | V |

Respecto a los servicios Web RESTful:

|  |  |  |
| --- | --- | --- |
| 41. | En un servicio Web RESTFul, los mensajes constan de una cabecera formada por varios campos fijos definidos por el estándar RESTful, un conjunto de propiedades que pueden ser definidas por la aplicación y de un contenido (normalmente, en XML o JSON).  *JUSTIFICACIÓN: El estilo arquitectónico REST no especifica ninguna cabecera, ni campos fijos*  *definidos. Por tanto, los mensajes tienen estructura libre.* | F |
| 42. | GET https:/weatherapp.com/zipcodes es un ejemplo de llamada a un servicio Web  RESTFul. | V |
| 43. | Un sistema distribuido con comunicación basada en servicios web REST emplea generalmente comunicación sincrónica no persistente y direccionamiento directo. | V |
| 44. | Son atendidos por servidores que mantienen el estado de las peticiones de los clientes.  *JUSTIFICACIÓN: Los servicios son “sin estado”, de modo que los servicios no mantienen ninguna sesión respecto a las peticiones de los clientes.* | F |

Respecto al mecanismo de comunicación Java Message Service:

|  |  |  |
| --- | --- | --- |
| 45. | JMS utiliza direccionamiento directo, pues en la cabecera del mensaje se especifica claramente el destino del mensaje.  *JUSTIFICACIÓN: JMS emplea direccionamiento indirecto, ya que el cliente envía el mensaje a un destino (Destination), que puede ser una cola (Queue) o un tema (Topic). Y no tiene por qué conocer quién será*  *el consumidor del mensaje.* | F |
| 46. | Java Message Service ofrece una comunicación fuertemente acoplada, pues tanto el productor como el consumidor de un mensaje se crean utilizando la misma factoría de conexión.  *JUSTIFICACIÓN: JMS ofrece una comunicación débilmente acoplada, ya que al emplear destinos, el productor y el consumidor del mensaje no necesitan conocerse. Solamente requieren conocer el destino*  *(y estar de acuerdo en el formato del mensaje).* | F |
| 47. | Existen dos tipos de destinos: Colas (Queues), utilizadas para enviar mensajes a un único cliente; y Temas (Topics), que permiten la publicitación/suscripción. | V |
| 48. | Las factorías de conexiones y los destinos se crean utilizando una herramienta  administrativa ofrecida por el Proveedor de JMS. | V |

# [FINAL 2014]

Sobre la invocación a objetos remotos (ROI):

|  |  |
| --- | --- |
| 41. El proxy ofrece la misma interfaz que el esqueleto. | F |
| 42. Existe un esqueleto por cada método del objeto remoto. | F |
| 43. El proxy empaqueta los argumentos del método llamado antes de que el esqueleto invoque al objeto remoto. | V |
| 44. El esqueleto desempaqueta los argumentos enviados por el proxy antes de invocar al objeto  remoto. | V |

# [2º PARCIAL 2014]

1. Sobre el mecanismo de comunicación RPC:

|  |  |
| --- | --- |
| F | A.- Al igual que ROI, siempre requiere que se utilice un servidor de nombres. |
| V | B.- El stub cliente sirve para poder invocar procedimientos remotos como si fuesen locales |
| F | C.- Sólo podemos definir como procedimientos remotos aquellos que no utilicen paso de  parámetros por referencia. |
| F | D.- Se basa en un modelo de invocación/respuesta con asincronía y persistencia |
| F | E.- El programador debe escribir el código de los stubs cliente y servidor para cada procedimiento que pueda invocarse de forma remota |

1. Sobre el mecanismo de comunicación ROI:

|  |  |
| --- | --- |
| V | A.- El cliente obtiene un proxy para invocar al objeto remoto. |
| V | B.- El proxy incluye una referencia al objeto remoto |
| F | C.- El mecanismo de ROI permite el paso de objetos como argumentos en las invocaciones, utilizando “paso por valor”. El paso de los objetos por referencia se simula mediante el paso por valor, de forma similar a como se emplea en el mecanismo RPC. |
| F | D.- En Java RMI, se requiere emplear un lenguaje especial de definición de interfaces,  denominado IDL (Interface Definition Language), para describir los interfaces de los objetos remotos |
| F | E.- El programador debe escribir el código del proxy y esqueleto para cada objeto remoto |