Feedback — Week 8 Quiz

You submitted this quiz on **Mon 26 Jan 2015 9:48 PM PST**. You got a score of **32.00** out of **32.00**. However, you will not get credit for it, since it was submitted past the deadline.

Question 1

Which of the following define what a Bound Service is, according to the video:

Your Answer		Score	Explanation
☐ It enables applications to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers	✔	1.00	
☑ It offers a client/server interface that allows extended two-way conversations between 1 or more clients and the service	✔	1.00	
☑ It runs only as long as at least one client remains bound to it and is automatically destroyed when all clients unbind	✔	1.00	
☐ It performs a single operation on behalf of the client that started it and often does not return a reply to the client	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

Please see video S2-M1-P7 Programming Bound Services With Messengers (Part 1)

Question 2

Which of the following methods inherited from Service *must* be overridden to create a Bound Service, according to the video:

Your Answer	Score	Explanation
☐ onUnbind()	✔ 1.00	
☐ onStartCommand()	✔ 1.00	
☐ onRebind()	✔ 1.00	
☑ onBind()	✔ 1.00	
Total	4.00 / 4.00	

Question Explanation

Please see video S2-M1-P7 Programming Bound Services With Messengers (Part 1)

Question 3

Which of the following describe the reason(s) why UniqueIDGeneratorService defines the makeIntent() factory method, according to the video:

Your Answer	Score	Explanation
☑ To shield the UniqueIDGeneratorActivity from details of how the Service is implemented	✔ 1.00	
☐ To return an IBinder associated with the request Messenger when an Activity binds to the Service	✔ 1.00	
☐ To ensure that the UniqueIDGeneratorActivity and the Service run in the different processes	✔ 1.00	
☐ To ensure that the UniqueIDGeneratorActivity and the Service run in the same process	✔ 1.00	
Total	4.00 / 4.00	

Question Explanation

Please see video S2-M1-P8 Programming Bound Services With Messengers (Part 2)

Question 4

Which of the following are motivations for using a pool of threads in the UniqueIDGeneratorService, according to the video:

Your Answer	Score	Explanation
☑ It enables multiple client requests to run concurrently	✔ 1.00	
☐ It ensures the UniqueIDGeneratorActivity doesn't block synchronously waiting for the service results	✔ 1.00	
☑ It can improve performance on a multi-core device	✔ 1.00	
☐ It ensures that the random UUIDs are stored persistently and uniquely	✔ 1.00	
Total	4.00 / 4.00	

Question Explanation

Please see video S2-M1-P8 Programming Bound Services With Messengers (Part 2)

Question 5

Which of the following describes the intent of the Monitor Object pattern, according to the video:

Your Answer	Score	Explanation
☑ It allows an object's methods to cooperatively schedule their execution sequences.	✔ 1.00	
☐ Defines the units of concurrency on a component to be requests for service & processes these service requests in a different thread than the client that invoked the requests	✔ 1.00	
☑ Synchronize concurrent method execute to ensure that only one method at a time runs within an object	✔ 1.00	
☐ It packages a piece of application functionality to execute it in another context, such as in a different process or thread	✔ 1.00	

Total	4.00 / 4.00

> **Question Explanation**
>
> Please see video S3-M1-P1 The Monitor Object Pattern (Part 1)

Question 6

Which of the following are the two key benefits of the Monitor Object pattern to developers of concurrent software, according to the video:

Your Answer		Score	Explanation
☑ It presents a concise programming model for sharing an object among cooperating threads where clients need not be concerned with explicit serialization mechanisms when invoking synchronized methods on a monitor object	✔	1.00	
☐ The Monitor Object pattern decouples asynchronous and synchronous service processing in concurrent systems by introducing two layers (one for asynchronous processing and one for synchronous processing) that communicate via a queue	✔	1.00	
☐ The loose coupling between a monitor object's functionality and its concurrent control mechanisms make it easy to change synchronization and scheduling policies without modifying synchronized method implementations	✔	1.00	
☑ Synchronized methods can use their monitor conditions to determine when a thread should suspend or resume its execution and that of collaborating threads	✔	1.00	
Total		4.00 / 4.00	

> **Question Explanation**
>
> Please see video S3-M1-P1 The Monitor Object Pattern (Part 1)

Question 7

Which of the following are reasons why the LinkedBlockingQueue applies the Guarded Suspension pattern to its put() method implementation, according to the video:

Your Answer		Score	Explanation
☐ To inform any thread blocked in an await call that the queue is no longer full	✔	1.00	
☐ To ensure that intra-component method calls avoid self-deadlock and minimize locking overhead	✔	1.00	
☑ To wait for space to become available if the queue is full	✔	1.00	
☐ To atomically increment the count of the number of elements in the queue	✔	1.00	
Total		4.00 / 4.00	

Question Explanation

Please see video S3-M1-P2 The Monitor Object Pattern (Part 2)

Question 8

Which of the following best describes the definition of "invariant", according to the video:

Your Answer		Score	Explanation
☐ The assumption that the data members in a monitor object are all initialized prior to the invocation of the first synchronized method by a thread	✔	1.00	
☑ Conditions that can be relied upon to be true whenever a synchronized method waits on the corresponding monitor condition	✔	1.00	
☐ The ability to define multiple monitor conditions corresponding to different changes in state	✔	1.00	

☐ An operation that requires both a lock to be acquired and a   ✔   1.00
precondition to be satisfied before the operation can be
executed

Total		4.00 / 4.00

Question Explanation

Please see video S3-M1-P2 The Monitor Object Pattern (Part 2)

☐ An operation that requires both a lock to be acquired and a   ✔   1.00
precondition to be satisfied before the operation can be
executed