1 Punto

1. What types of User Notifications are provided by Android?

- [x] Toast Messages.
- [x] Notification Area Notifications.
- [ ] Fragments.
- [ ] Tabs.
- [x] Dialogs.

1 Punto

2. (True or False) Toast messages are used to get information from the user?

- ( ) True.
- (•) False.

1 Punto

3. Why do Notification Area Notifications use PendingIntents?

- ( ) The underlying Intent has a reference to the sending component which can lead to memory leaks.
- ( ) To allow Extra data to be provided to the Activity that will be started.
- (•) The underlying Intent will be used by the system, rather than by the component that created it.
- ( ) The PendingIntent can add a listener to the underlying Intent

1 Punto

4. Which of the following capture why it is preferable to notify the user with a Notification Area Notification, rather than with a Dialog, or vice versa.

- [x] Use a Notification Area Notification when the user should be notified outside of any currently running application.
- [ ] Use a Notification Area Notification to prevent onPause() from being called.
- [x] Use a Dialog when the application needs to get user feedback.
- [ ] Use a Dialog (DialogFragment) when using a large screen device such as a tablet.

1 Punto

5. When should your application send broadcasts using the LocalBroadcastManager class, rather than by using the Context class or vice versa?

- [x] Use the LocalBroadcastManager to broadcast Intents that will only be received within the same application the sends the broadcasts.
- [x] Use the LocalBroadcastManager to register BroadcastReceivers that don't want to receive broadcasts from outside the application.
- [x] Use the Context class when the broadcast must be sticky.
- [ ] Use the Context class to improve application reliability.

1 Punto

6. If your application only wants to receive certain broadcasts while it is active and in the foreground, which of the following scenarios might it implement?

- ( ) Dynamically register its BroadcastReceivers with low priority. Then use abortBroadcast() at runtime to prevent delivery.
- (•) Dynamically register its BroadcastReceivers in onResume() and unregister them in onPause().
- ( ) Statically register its BroadcastReceivers with low priority.
- ( ) Load the Intents through a menu or ActionBar action.

1 Punto

7. Which of the following statements capture how Alarms are different from other Android capabilities?

- [x] Notification Area Notifications inform users about events without interrupting their work, while Alarms don't directly inform users.
- [ ] Handlers cannot be used to send Intents at a future point in time.
- [x] Alarms are fired at a particular time in the future. Regular Intent Broadcasts are handled at the time the Intent is broadcast.

1 Punto

8. How does an application get access to the AlarmManager?

- ( ) Use the AlarmManager() constructor to create an instance of the AlarmManager.
- (•) Use the Context.getSystemService() method to retrieve a reference to the AlarmManager service.
- ( ) Use the AlarmManager.newInstance() method to retrieve the singleton instance of the AlarmManager.
- ( ) Put a <manager> tag in the application's AndroidManifest.xml file.

1 Punto

9. When setting alarms, it's often better to use the ELAPSED_REALTIME or ELAPSED_REALTIME_WAKEUP alarm types, rather than RTC or RTC_WAKEUP alarm types. Which of the following statements explains why RTC and RTC_WAKEUP alarms might not be the best approach in some cases?

- [ ] ELAPSED_REALTIME Alarms can fire when the CPU is in sleep mode.
- [x] If the user manually changes the time zone or modifies the system clock, RTC Alarms may fire at unpredictable times.
- [ ] It doesn't really matter, because you can easily convert from one time interpretation to the other.
- [x] If the network resets the system clock, RTC Alarms may fire at unpredictable times.

1 Punto

10. **For API targets prior to 19:** The setInexactRepeating() method is intended to give Android flexibility in the exact timing of alarms. Assuming that mAlarmManager is a valid reference to the AlarmManager and that pi is a valid reference to a PendingIntent, why doesn't the following code snippet (modified from the AlarmCreate application shown in this lesson) accomplish that purpose?

```
mAlarmManager.setInexactRepeating(AlarmManager.ELAPSED_REALTIME,
SystemClock.elapsedRealtime(),15000, pi);
```

- ( ) setInexactRepeating() requires a time interval of 60000 or greater.
- ( ) setInexactRepeating() is a method of the Alarm class.
- (•) setInexactRepeating() requires a specific interval constant, such as INTERVAL_FIFTEEN_MINUTES.
- ( ) setInexactRepeating() requires an alarm type of RTC or RTC_WAKEUP.