How to handle multiple sounds in Java ME

Most phones will throw an exception if you try to keep two sounds loaded at the same time, so the trick is to deallocate the previous sound before you play the next one. The code below shows a SoundManager class, which keeps track of all the sounds in your midlet in a hashtable. It also keeps track of the time that the last sound was played, because if you try to play two sounds too close to each other it will fail in most phones.

```
import java.io.InputStream;
import java.util.Enumeration;
import java.util.Hashtable;
import javax.microedition.media.Manager;
import javax.microedition.media.Player;


public class SoundManager
{
 private static Hashtable mapSound = new Hashtable();
 private static long timeLastSound = 0;
 private static boolean enabled = true;

 static public Player getPlayer(String sound)
 {
  Player player;
  player = (Player)mapSound.get(sound);
  if (player == null)
  {
   try
   {
    String fileSpec = "/";
    fileSpec += sound;
    InputStream is = SoundManager.class.getResourceAsStream(fileSpec);
    if (fileSpec.endsWith("amr"))
      player = Manager.createPlayer(is, "audio/amr");
    else if (fileSpec.endsWith("mid"))
      player = Manager.createPlayer(is, "audio/midi");
    mapSound.put(sound, player);
   }
   catch (Exception ex)
   {
    DebugStuff.print(ex.toString());
   }
  }
  return player;
 }

 public static void playSound(String sound)
 {
  if (!enabled) return;
  if (Math.abs(System.currentTimeMillis() - timeLastSound) < 250)
  {
   DebugStuff.print("tried to play 2 sounds too soon");
   return;
  }
  timeLastSound = System.currentTimeMillis();
  Player player = getPlayer(sound);
```

```
   try
   {
    Enumeration enumValues;
    Player playerOther;
    for (enumValues = mapSound.elements(); enumValues.hasMoreElements() ;)
    {
     playerOther = (Player)enumValues.nextElement();
     if (playerOther != player && playerOther.getState() == Player.STARTED)
      playerOther.deallocate();
    }
    if (player.getState() == Player.STARTED)
    {
     DebugStuff.print("sound already started");
     return;
    }
    player.start();
   }
   catch (Exception e)
   {
    DebugStuff.print(e.toString());
   }
  }

  public static void setEnabled(boolean enabled)
  {
   SoundManager.enabled = enabled;
  }

}
```

How to call use the SoundManager:

```
SoundManager.playSound("sound1.amr");
```

(Notice I am using the amr format instead of wav. This is because amr is much smaller, has more or less the same quality and is equally well supported as the wav format.)