Using Your NXT with LeJOS

Recommendations and Random Tips

(1) Turn on NXT with the orange button.
(2) Turn off NXT by simultaneously pressing the orange and dark gray button.
(3) When compiling your code, remember that you need to wait 10 seconds after turning on your NXT.
(4) Create your code in a default directory rather than a package. Packages are possible but the current version of LeJOS has a bug that makes it difficult.
(5) Put your code in a folder that has no spaces in the name. If you are using my package, then put my code in the same folder as well (but keep my code in the original set of folders rgis\cs427\dbahr\robot).
(6) Read the LeJOS API. It is in the “docs” folder of the LeJOS installation. The LeJOS API has many classes that you might find useful. I have also written a number of classes that you may find useful, and that you can download from my website. My classes are adaptations of existing classes but simpler to use and with significantly better comments.
(7) When writing code for your robot, do *NOT* use classes that are in the general java API. The NXT has very limited memory, so it has to use a much smaller subset of java. Make sure you restrict yourself to classes that either exist in the LeJOS API or were built from classes that already exist in the LeJOS API.
(8) If you need to use \texttt{System.currentTimeMillis()}, then cast the result to a double. LeJOS doesn’t support variables of type “long” (or not very well).
(9) There is very little memory and there is no garbage collection. So if you create a variable it will hang around indefinitely. Therefore, try to create as few variables as necessary. If a method will be called repeatedly, then try not to create a variable inside that method. Sometimes you may need to create a single instance variable that can be used repeatedly inside your method. Don’t be afraid of variables! Just don’t overdue it.
(10) If your robot is working great and then all the sudden throws an exception for no apparent reason (shown on the LCD display), then immediately consider the possibility that you have too many variables.
(11) You can run out of memory if you use string literals in a loop. A string literal is where you create a String on the fly right where you need it. For example,

```java
while(continue)
{
    LCD.drawString(“hello”);
}
```
The “hello” creates a new variable every time it is called. Instead do this, which only creates the variable once.

```java
String message = "hello";
while(continue)
{
    LCD.drawString(message);
}
```

(12) You may need your code to wait for a while before doing something new. In that case, use “Thread.sleep(milliseonds)”. This tells your robot to wait the number of specified milliseconds before continuing. This method must be inside of a try-catch clause. For example, the following code will make your robot move forward for 3 seconds, and then stop.

```java
motor.forward()
try
{
    Thread.sleep(3000);
}
catch(InterruptedException e)
{
}
motor.stop();
```

(13) The leJOS Sound class has many features, but be careful. If you try to play two notes, one right after the other, then only the second note will be heard. Why? Because the method Sound.playTone() returns immediately, so the next note plays on top of the first one. To avoid this problem, use my Tone class (RECOMMENDED!). Or insert a call to Thread.sleep(duration) after each note is played. Make duration the same length as the note that was just played (but be sure to convert units).