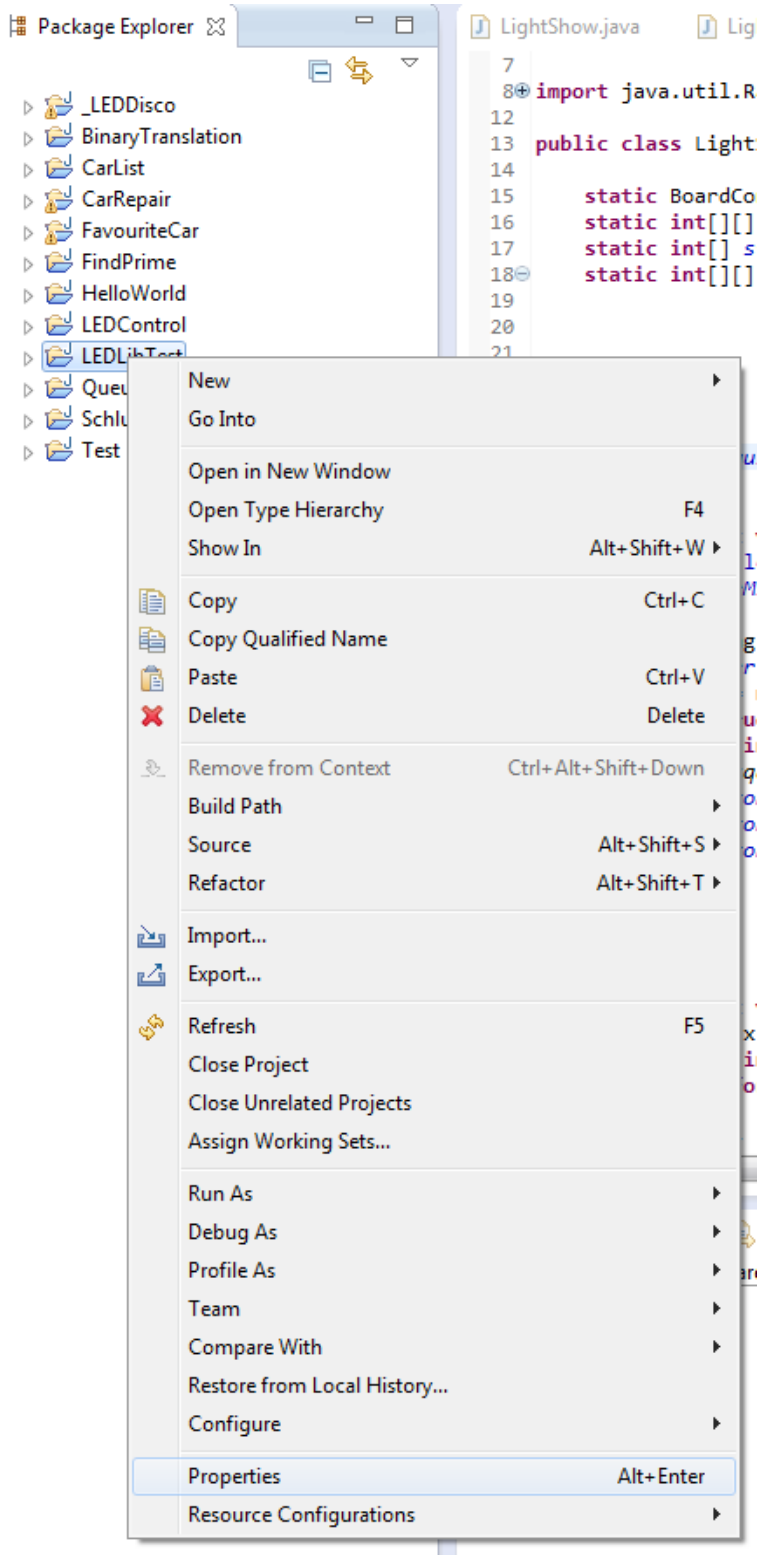
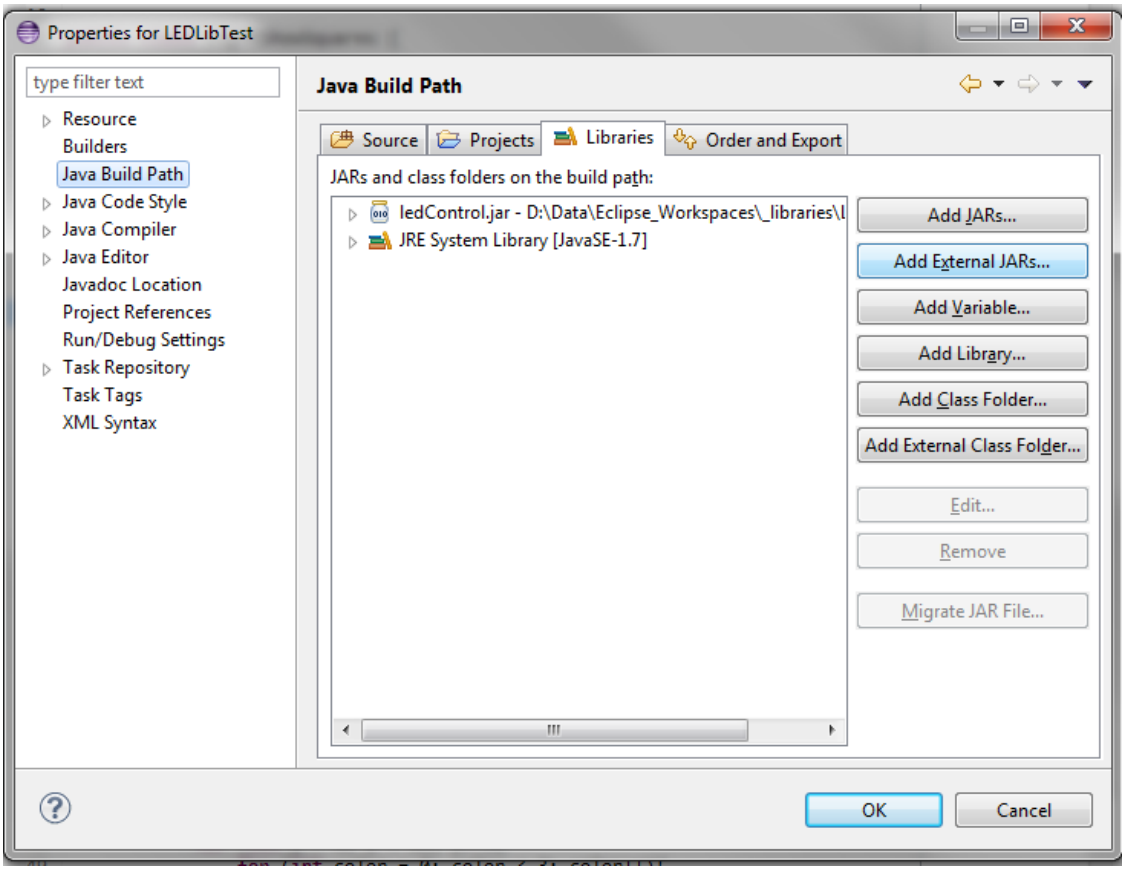
Anleitung zu Nutzung der LEDBoard-Bibliothek

Schritt 1 – Installation der Bibliothek

Um die Bibliothek in euer Eclipseprojekt einzubinden müßt ihr einmal einen Rechtsklick auf euer Projekt machen und dann ganz unten die Option Properties wählen.



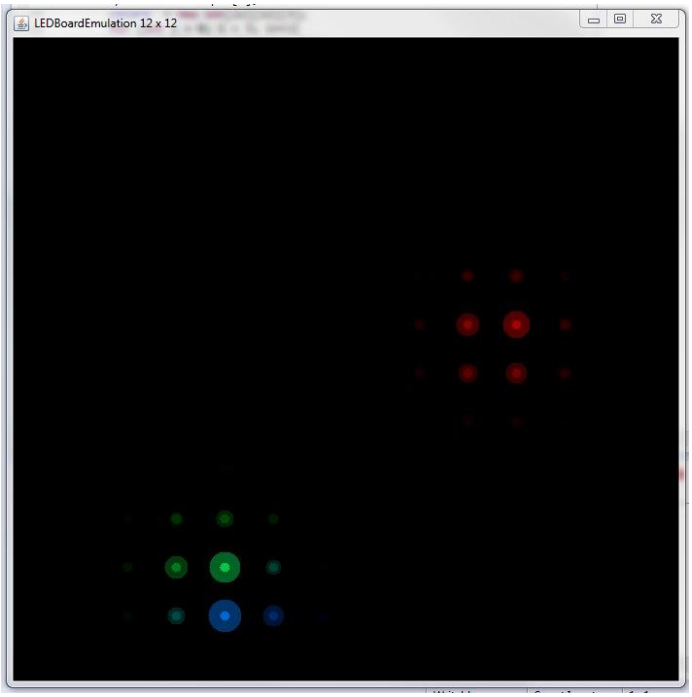
In dem Menu das danach erscheint unter der Auswahl links „Java Build Path“ wählen und dann den Reiter „Libraries“. Das sollte ungefähr so aussehen:



Jetzt noch den Knopf „Add external JARs...“ drücken und im folgenden Dialog die Datei ledControl.jar auswählen. In eurem zip ist der Dateiname wahrscheinlich noch durch Versions- und Revisionsnummer ersetzt. Danach noch einmal ok klicken und die Bibliothek sollte korrekt eingebunden sein.

Schritt 2 – Testen der Bibliothek

Ihr habt neben der Bibliothek auch mehrere Programme runtergeladen, die die Funktion der Bibliothek demonstrieren.
Zieht den Ordner „lightshow“ in den „src“-Ordner des Projektes in das ihr die LED-Bibliothek eingebunden habt. Wie das geht könnt ihr in der Hilfestellung „JACKdateien in Eclipse einbinden“ lesen. Wenn ihr alles korrekt eingebunden habt und ihr eins der Programme startet, solltet ihr ein Fenster in dieser Art sehen:



Die einzelnen Programme finden sich in im package `lightshow`:

- In dem Unterpaket `lightshow.tutorial` befinden sich mehrere gut kommentierte Beispiel die den Einstieg erleichtern sollen. Zwei der Tutorials zeigen die Einbindung der Tastatur.
- Unter `lightshow.example` befinden sich drei etwas komplexere, kaum kommentierte Beispielprogramme. Zwei der Programme bestehen aus mehreren Klassen. Zum Starten der Programme muss jeweils die Klasse die die Methode
  `public static void main(String[] args)`
  enthält gestartet werden.

Falls die Programme nicht laufen könnte das daran liegen, dass die Paketangaben in den Klassen `package lightShow.example.spots; //der genaue Paketname haengt von der Klasse ab` nicht mit der Paketstruktur in eurem EclipseWorkspace übereinstimmen.

Ihr könnt entweder die Paketangaben entfernen, oder die packages mit den korrekten Namen erstellen und die Quelldateien in dieses Paket schieben. Das sollte allerdings nicht nötig sein, wenn ihr euch an die obigen Anleitungen haltet.

Nutzung der Bibliothek

Um ein Programm für die Steuerung des LEDBoards zu schreiben, muß als erstes ein Controller vom Typ `BoardController` angefordert werden.
Dies geschieht mit dem Befehl

```
BoardController controller = BoardController.getBoardController();
```

Mit diesem Aufruf erhaltet ihr einen BoardController der eine Emulation in einem Fenster steuert, so daß ihr auch daheim für das Board programmieren könnt.

Der Aufruf kann parametrisiert werden um andere Boardkonfigurationen und andere maximale Frameraten zu verwenden, sowie um die Emulation abzuschalten. Wenn ihr diese Optionen verwenden wollt, schaut bitte in den Quellcode der ledControl-Bibliothek. Für dem Anfang ist es unnötig sich darum zu kümmern.

Der Aufbau eines Bildes geschieht mittels eines 3-dimensionalen Arrays:

```
int[][][] colors;
```

Die ersten beiden Dimensionen geben dabei die Koordinate der LED an die ihr färben wollt, mit der dritten Dimension wählt ihr die RGB Farbkomponenten. Für die ersten beiden Indizes sind also Werte von 0 bis 11 (je 12 Koordinaten in beide Richtungen) und für den letzten Werte von 0 bis 2 (je eine pro Grundfarbe rot, grün, blau) zulässig.

```
colors[4][5][0] = 50; // LED an Position (4,5) wird der Rotwert 50 zugeteilt
colors[4][5][1] = 50; // LED an Position (4,5) wird der Gruenwert 50 zugeteilt
colors[4][5][2] = 50; // LED an Position (4,5) wird der Blauwert 50 zugeteilt
```

Der BoardController verwaltet ein solches Array. Ihr könnt nun

- das Array über den BoardController ansprechen (die einfachste Variante)
  ```
  void setColor(int x, int y, int red, int green, int blue)
  ```

- selbst ein Array erzeugen, bearbeiten und es dann dem BoardController übergeben
  ```
  void setColors(int[][][] colors)
  ```

- oder euch das Array vom BoardController geben lassen und direkt darauf arbeiten (die Variante mit der man am leichtesten schwer zu findende Fehler erzeugt)
  ```
  int[][][] getColors()
  ```

Methoden des Controllers

Die Folgenden auch im Quelltext kommentierten Methoden stehen euch dazu zur Verfügung:

```
void setColor(int x, int y, int red, int green, int blue)
```

x und y geben die Koordinate der LED an die ihr färben wollt, red, green und blue geben die Werte der entsprechenden Farbkomponenten an.

```
void setColors(int[][][] colors)
```

Übergebt ein bereits von euch bearbeitetes Array an den Controller. Das Array muß dabei die korrekten Dimensionen aufweisen (Standard: 12x12x3).

```
int[][][] getColors()
```

Gibt das vom Controller verwaltete Array zurück, damit ihr direkt darauf arbeiten könnt.

```
void updateBoard()
```

Diese Methode muss aufgerufen werden, wenn das vom Controller verwaltete Array alle von euch gewünschten Farbinformationen hat. Erst dann wird das Bild tatsächlich auf dem Board dargestellt.

```
void sleep(int ms)
```

Wenn Timing für euer Projekt wichtig ist, hilft euch diese Methode. Diese Methode wartet ab, bis die durch ms gegebene Zeit in Millisekunden seit dem letzten Bildupdate durch `updateBoard()` vergangen ist.

```
void checkColors()
```

Diese Methode prüft, ob die Werte des verwalteten Arrays gültige Werte sind. Finden sich illegale Werte, erfolgt eine entsprechende Ausgabe auf der Konsole.
Verwendet diese Methode, falls ihr seltsame Farben auf dem Board seht, oder unerklärliche Exceptions bekommt. Vielleicht liegt der Fehler im Farbarray.

```
void resetColors()
```

Diese Methode setzt auf dem verwalteten Array alle Farbwerte auf die Hintergrundfarbe zurück. Wenn die Hintergrundfarbe nicht explizit gesetzt wurde, ist ist das Board nach Aufruf dieser Methode schwarz, d.h. alle Farbwerte sind auf 0 gesetzt.

```
void setBackgroundColor(int red, int green, int blue)
void setBackgroundColor(int[] color)
```

Diese Methoden setzen die Hintergrundfarbe des Boards. Jedesmal wenn `resetColors()` aufgerufen wird und nur dann, werden alle LEDs auf die hier übergebenen Farbwerte gesetzt.

```
void setBoardColor(int red, int green, int blue)
void setBoardColor(int[] color)
```

Diese Methoden färben das gesamte Board in den angegebenen Farbwerten. Die durch `resetColors()` verwendete Hintergrundfarbe wird dadurch nicht beeinflusst.

```
void addColor(int x, int y, int red, int green, int blue)
void addColor(int x, int y, int[] newColor)
```

Diese Methoden addieren gegebene Farbwerte auf die bereits vorhandenen Farbwerte einer LED. Die Maximal- und Minimalwerte werden dabei auf keinen Fall über/unterschritten.

```
void maxColor(int x, int y, int red, int green, int blue)
void maxColor(int x, int y, int[] color)
```

Diese Methoden setzen die Farbwerte an der gegebenen Koordinate auf das Maximum der übergebenen und der bereits an der Koordinate vorhandenen Farbwerte.

```
void setOffBoardDrawingWarning(boolean warning)
```

(De-)Aktiviert Warnungen, falls ihr versucht Pixel zu zeichnen, die nicht auf dem Board sind. Aktiviert diese Option, wenn ihr euch wundert dass Pixel nicht gezeichnet werden. Vielleicht zeichnet ihr versehentlich an Koordinaten die nicht auf dem Board sind.

Der Controller und Tastatureingaben

Wenn Tastatureingaben ausgewertet werden sollen, empfiehlt sich die Verwendung des KeyBuffers. Verwendet bitte den Keybuffer um Darstellungsprobleme durch Threads zu vermeiden!
Sobald der Boardcontroller angefordert wurde, kann vom BoardController der KeyBuffer angefordert werden:

```
public KeyBuffer getKeyBuffer()
```

Der Keybuffer ist NICHT verwendbar, wenn ihr die Emulation deaktiviert habt!

Der **KeyBuffer** verfuegt ueber die folgenden Methoden:

```
public KeyEvent pop()
```

Mit dieser Methode wird das am längsten zurückliegende KeyEvent zurückgegeben und aus dem Puffer entfernt.

```
public KeyEvent[] popAll()
```

Mit dieser Methode werden alle im Puffer vorhandenen KeyEvents als Array zurückgegeben und aus dem Puffer entfernt. Das erste Element im Array ist dabei das älteste, das letzte Element das jüngste Event.

```
public void clear()
```

Diese Methode löscht alle Events aus dem Puffer.

```
public int eventsInBuffer()
```

Diese Methode gibt zurück, wie viele unverarbeitete Ereignisse sich noch im Puffer befinden.

KeyEvents

Für den Umgang mit dem KeyBuffer und KeyEvents findet ihr in den beigefügten Programmen zwei Beispiele. In der Beschreibung findet sich auch der Link auf die Dokumentationsseite zu KeyEvents von Oracle:
http://docs.oracle.com/javase/6/docs/api/java/awt/event/KeyEvent.html

Achtet bei der Verwendung von KeyEvents bitte darauf welche Keyevents ihr eigentlich in Puffer habt. Dort sammeln sich drei verschiedene unter obigem Link weiter ausgeführte Eventtypen:

KEY_PRESSED
KEY_TYPED
KEY_RELEASED

Wichtige Hinweise:

Farbwerte
Die Farbwerte die ihr verwendet müssen sich stets in dem Bereich von 0 (aus) bis 127 (maximale Helligkeit) bewegen. Andernfalls kann es zu Exceptions oder seltsamen Farben auf dem Board kommen.

Koordinaten der LEDs
In der Standardeinstellung geht der BoardController von 12x12 LEDs aus. Wie von Arrays bekannt beginnt die Indizierung der LEDs bei 0. Es gibt also jeweils die Reihen und Spalten 0 – 11. Die Angabe der Koordinaten erfolgt dabei nicht, wie ihr es vom Standardkoordinatensystem gewohnt seid. Die Angabe der Achsen erfolgt in der bekannten Reihenfolge, die Richtung der y-Achse ist aber umgekehrt.
Die Koordinate (0, 0) bezeichnet die LED **oben** links. Die LED **unten** links erreicht ihr über (0, 11), die LED oben rechts über (11, 0) und die LED unten rechts über (11, 11).
Falls ihr Probleme mit Pixeln die nicht gezeichnet werden habt, nutzt nach der Initialisierung des BoardControllers den Aufruf

```
BoardController.setOffBoardDrawingWarning(true);
```

Ihr erhaltet dann Warnungen, wenn ihr versehentlich an Koordinaten zeichnen wollt, die außerhalb des Boards liegen.

Größeres Board
Falls ihr für eure Ideen mehr Platz benötigt, könnt ihr ein 20x20 Board verwenden. Ihr müsst dazu bei der ersten Anforderung des Boards einen Parameter übergeben:

```
BoardController.getBoardController(LedConfiguration.LED_20x20_EMULATOR);
```

Solltet ihr das große Board verwenden sind wir allerdings nicht mehr in der Lage das auf dem echten Board darzustellen.

Das echte Board
Helligkeit und auch Farben eurer Programme können auf dem echten Board zum Teil deutlich von der Darstellung im Emulator abweichen. Gegen Ende des Semesters werden mehrere Termine bekannt gegeben an denen das echte Board in der Programmierberatung zur Verfügung steht, damit ihr euer Programm vor der Präsentation noch anpassen könnt.

Nutzung auf dem echten LED-Board:

Um euer Projekt auf dem echten Board laufen zu lassen müsst ihr lediglich eine Zeile im Programmcode hinzufügen. Um zu erfahren wie das geht kommt einfach in die Programmierberatung. Die SHKs dort wissen, wie mit dem Board umzugehen ist und können euch schnell helfen, sobald das Board in der Programmierberatung zur Verfügung steht.