```java
package framework.abletonLive.devices;

public enum DeviceIdentifier
{
    DISTORT
    {
        @Override
        public String toString()
        {
            return "Distort";
        }
    },

    PITCH
    {
        @Override
        public String toString()
        {
            return "Pitch & Echo";
        }
    };
}
```

```java
package framework.abletonLive.devices;

import java.time.Duration;
import java.time.Instant;
import java.util.Arrays;
import java.util.Date;
import java.util.List;

import framework.abletonLive.AbletonLiveOSCController;
import framework.com.illposed.osc.OSCListener;
import framework.com.illposed.osc.OSCMessage;
import framework.com.illposed.osc.OSCPortIn;
import framework.com.illposed.osc.utility.OSCMessageSender;
import framework.com.illposed.osc.utility.OSCRegexAddressSelector;

public abstract class Device implements OSCListener
{
    private final String            listReply_              = "/live/devicelist";
    private final String            paramReply_             = "/live/device/allparam";
    private final String            paramRangeReply_        = "/live/device/range";
    private final DeviceIdentifier  identifier_;
    private final OSCMessageSender  oscMessageSender_;
    private final int               trackIndex_;
    private final String            trackName_;

    private int                     deviceIndex_            = -1;

    private boolean                 receiveListReply_       = false;
    private boolean                 receiveParamReply_      = false;
    private boolean                 receiveParamRangeReply_ = false;

    /**
     * Use </br>
     * {@link AbletonLiveOSCController#addDevice(String, DeviceIdentifier)}
     * or</br>
     * {@link AbletonLiveOSCController#addDevice(framework.abletonLive.Track, DeviceIdentifier)}</br>
     * to create a device!
     */
    public Device(DeviceIdentifier identifier, OSCMessageSender oscMessageSender, OSCPortIn oscPortIn, int
    trackIndex,
            String trackName)
    {
        identifier_ = identifier;
        oscMessageSender_ = oscMessageSender;
        trackIndex_ = trackIndex;
        trackName_ = trackName;
        oscPortIn.addListener(new OSCRegexAddressSelector(listReply_, paramReply_, paramRangeReply_),
        this);
        init();
    }

    private void init()
    {
        receiveListReply_ = true;

        List<Object> args = Arrays.asList(trackIndex_);
        oscMessageSender_.send(new OSCMessage("/live/devicelist", args));

        waitForListReply(1000);

        if (deviceIndex_ < 0)
        {
            throw new IllegalStateException(identifier_.toString() + " not a device of the track " +
            trackName_);
        }
    }

    private void waitForListReply(int timeout)
    {
        boolean timedout = false;
        Instant start = Instant.now();

        while (!timedout && receiveListReply_)
```

```
        {
            long diff = Duration.between(start, Instant.now()).toMillis();
            timedout = diff >= timeout;
            try
            {
                Thread.sleep(20);
            } catch (InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    }

    /**
     * <b>Debugmethod</b></br>
     * Triggers Ableton Live to send a list whith all parameter associated with
     * this device.
     */
    public void requestParameters()
    {
        receiveParamReply_ = true;
        List<Object> args = Arrays.asList(trackIndex_, deviceIndex_);
        oscMessageSender_.send(new OSCMessage("/live/device", args));
    }

    /**
     * <b>Debugmethod</b></br>
     * Triggers Ableton Live to send the range of all parameters of this device.
     */
    public void requestParameterRange()
    {
        receiveParamRangeReply_ = true;
        List<Object> args = Arrays.asList(trackIndex_, deviceIndex_);
        oscMessageSender_.send(new OSCMessage("/live/device/range", args));
    }

    protected DeviceIdentifier getIdentifier()
    {
        return identifier_;
    }

    protected OSCMessageSender getOscMessageSender()
    {
        return oscMessageSender_;
    }

    protected int getTrackIndex()
    {
        return trackIndex_;
    }

    protected int getDeviceIndex()
    {
        return deviceIndex_;
    }

    @Override
    public void acceptMessage(Date time, OSCMessage message)
    {
        if (receiveListReply_ && message.getAddress().matches(listReply_))
        {
            List<Object> args = message.getArguments();
            for (int i = 1; i < args.size(); i += 2)
            {
                int deviceIndex = (int) message.getArguments().get(i);
                String deviceName = (String) message.getArguments().get(i + 1);

                if (deviceName.toLowerCase().matches(identifier_.toString().toLowerCase()))
                {
                    deviceIndex_ = deviceIndex;
                    receiveListReply_ = false;
                }
            }
```

```
        } else if (receiveParamReply_ && message.getAddress().matches(paramReply_))
        {
            receiveParamReply_ = false;
            System.out.println("Parameter of device " + identifier_.toString() + " (track " + trackName_ +
            "):");
            List<Object> args = message.getArguments();
            for (int i = 2; i < args.size(); i += 3)
            {
                System.out.print("\tname: " + args.get(i + 2).toString());
                System.out.print("\t\tindex: " + args.get(i).toString());
                System.out.println("\tvalue: " + args.get(i + 1).toString());
            }
        } else if (receiveParamRangeReply_ && message.getAddress().matches(paramRangeReply_))
        {
            receiveParamRangeReply_ = false;
            List<Object> args = message.getArguments();
            System.out.println("Range of the requestet parameter of device " + identifier_.toString() + "
            on track "
                    + trackName_ + " is:");
            for (int i = 2; i < args.size(); i += 3)
            {
                System.out.print("\tparam: " + args.get(i).toString());
                System.out.print("\tmin: " + args.get(i + 1).toString());
                System.out.println("\tmax: " + args.get(i + 2).toString());
            }
        }
    }

    @Override
    public String toString()
    {
        return "name: " + identifier_ + " track: " + trackName_ + " device index: " + deviceIndex_;
    }
}
```

```java
package framework.abletonLive.devices;

import java.util.Arrays;
import java.util.List;

import framework.abletonLive.AbletonLiveOSCController;
import framework.com.illposed.osc.OSCMessage;
import framework.com.illposed.osc.OSCPortIn;
import framework.com.illposed.osc.utility.OSCMessageSender;

public class DistortDevice extends Device
{
    // Paramter indices
    private final int   on_         = 0;
    private final int   freq_       = 1;
    private final int   width_      = 2;
    private final int   drive_      = 3;
    private final int   dryWet_     = 4;
    private final int   tone_       = 5;
    private final int   dynamics_   = 6;
    //

    /**
     * Use </br>
     * {@link AbletonLiveOSCController#addDevice(String, DeviceIdentifier)}
     * or</br>
     * {@link AbletonLiveOSCController#addDevice(framework.abletonLive.Track, DeviceIdentifier)}</br>
     * to create a device!
     */
    public DistortDevice(OSCMessageSender oscMessageSender, OSCPortIn portIn, int trackIndex, String
    trackName)
    {
        super(DeviceIdentifier.DISTORT, oscMessageSender, portIn, trackIndex, trackName);
    }

    /**
     * Sets the filter freq parameter of this device.
     *
     * @param value range: 0.0 to 1.0
     */
    public void setFilterFreq(double value)
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), freq_, (float)value);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }

    /**
     * Sets the filter width parameter of this device.
     *
     * @param value range: 0.5 to 9.0
     */
    public void setFilterWidth(double value)
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), width_, (float)value);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }

    /**
     * Sets the drive parameter of this device.
     *
     * @param value range: 0.0 to 100.0
     */
    public void setDrive(double value)
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), drive_, (float)value);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }

    /**
     * Sets the dry/wet parameter of this device.
     *
     * @param value range: 0.0 to 100.0
     */
```

```
    public void setDryWet(double value)
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), dryWet_, (float)value);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }

    /**
     * Sets the tone parameter of this device.
     *
     * @param value range: 0.0 to 100.0
     */
    public void setTone(double value)
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), tone_, (float)value);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }

    /**
     * Sets the preserve dynamics parameter of this device.
     *
     * @param value range: 0.0 to 1.0
     */
    public void setPreserveDynamics(double value)
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), dynamics_, (float)value);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }

    /**
     * Activates this device.
     */
    public void on()
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), on_, 1.0f);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }

    /**
     * Deactivates this device.
     */
    public void off()
    {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), on_, 0.0f);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }
}
```

```java
package framework.abletonLive.devices;

import java.util.Arrays;
import java.util.List;

import framework.abletonLive.AbletonLiveOSCController;
import framework.com.illposed.osc.OSCMessage;
import framework.com.illposed.osc.OSCPortIn;
import framework.com.illposed.osc.utility.OSCMessageSender;

public class PitchDevice extends Device {
    // Paramter indices
    private final int transp_ = 11;

    /**
     * Use </br>
     * {@link AbletonLiveOSCController#addDevice(String, DeviceIdentifier)}
     * or</br>
     * {@link AbletonLiveOSCController#addDevice(framework.abletonLive.Track, DeviceIdentifier)}
     * </br>
     * to create a device!
     */
    public PitchDevice(OSCMessageSender oscMessageSender, OSCPortIn oscPortIn, int trackIndex, String
    trackName) {
        super(DeviceIdentifier.PITCH, oscMessageSender, oscPortIn, trackIndex, trackName);
    }

    /**
     * Sets the transp parameter of this device.
     *
     * @param value
     *            range: -2400.0 to 2400.0
     */
    public void setTransp(double value) {
        List<Object> args = Arrays.asList(getTrackIndex(), getDeviceIndex(), transp_, (float) value);
        getOscMessageSender().send(new OSCMessage("/live/device", args));
    }
}
```