UML Class Diagram (hand-drawn)

<<Singleton>> MusicPlayerApplication
vector<Song> songs;
createSong(String n, title, ...){}
createPlaylist(name){}
addSong(pname, sname){}
// Other Methods
to Call different
class

<<Singleton>> MusicPlayerFacade
AudioEngine engine;
Playlist currPl;
PlaylistStrategy strategy;
connectDevice(DeviceType d){}
setStrategy(StrategyType st){}
loadPlaylist(String name){}
playSong(Song song){}
pauseSong(Song song){}
playNext(){}
playPrev(){}
enqueueNext(){}...

<<Singleton>> PlaylistManager
map<String, Playlist> m;
createPlaylist(String name){}
addSong(String pname, String sname){}
getPlaylist(String name){}

Song
String title;
String artist;
String path;
double ...

<<Singleton>> StrategyManager
SequentialStrategy &;
RandomStrategy r;
CustomQueueStrategy c;
getStrategy(StrategyType){}

Playlist
String name;
vector<Song> songs;
addSong(Song song){}
// Getter & Setters

<> PlayStrategy
setPlaylist(Playlist p);
hasNext();
next();
hasPrevious();
previous();
addToNext(){}

<<Singleton>> DeviceManager
IAudioInputDevice dev;
connect(DeviceType d){}
getDevice(){}

DeviceType
BLUETOOTH
WIRED
HEADPHONE
...

<> IAudioOutputDevice
playAudio(Song song)

DeviceFactory
createDevice(DeviceType d){}

AudioEngine
Song catalog;
play(IAudioOutputDevice aod, Song s){}
pause(){}

StrategyType
SEQUENTIAL
RANDOM
CUSTOM

BluetoothSpeakerAdapter
BluetoothSpeakerAPI a;
playAudio(Song song);

WiredSpeakerAdapter
WiredSpeakerAPI a;
playAudio(Song song);

HeadphonesAdapter
HeadphoneAPI a;
playAudio(Song song);

BluetoothSpeakerAPI
playSoundViaBluetooth(String data){}

WiredSpeakerAPI
playSoundViaCable(String data){}

HeadphoneAPI
playSound(String data){}

SequentialPlayStrategy
Playlist p1; int index;
setPlaylist(Playlist p){}
// Alone implementations

RandomPlayStrategy
Playlist p1;
setPlaylist(Playlist p){}
// Alone implementations

CustomPlayStrategy
Playlist p1;
setPlaylist(Playlist p){}
// Alone implementations