Javadoc Documents

- **Song.java**

```
import java.net.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
*Song is a class to choose song from Database
*<pre>
* Song s1 = new Song("Disco", "XYZ", "ABC", "www.google.com/mit", "MP3", 4.0);
* s1.setName("Disco");
* s1.setArtist("XYZ");
* s1.setAlbum("ABC");
* s1.setUrl("www.google.com/mit");
* s1.setFormat("MP3");
* s1.setDuration(4.0);
* s1.play();
* </pre>
* @author Mittal Patel(patelmittal160@gmail.com)
* @version 0.1 13 February 2019
* @see java.io.File
* @see java.io.FileInputStream
* @see java.net.*;
* @see java.util.ArrayList;
* @see java.util.List;
* @see java.util.Scanner;
 */
public class Song {
/**
 * name is the name of the song
 */
String name;
/**
 * artist is the name of the artist of the song
 */
String artist;
/**
 * album is the album name belong to the song
 */
String album;
/**
 * url is the website of the song
```

```
 */
String url;
/**
 * format is the format of the song
 */
String format;
/**
 * Duration is the duration of Song in seconds
 */
int duration;
/**
 * The Default Constructor of the Song Class
 */
 public Song(){
 }
/**
 * Parametrized Constructor, used to assign values to the variables
 * @param name, artist, album, url, format, duration : Name, Artist, Album, URL, Format and Duration
of the Song
 */
 public Song(String name, String artist, String album, String url, String format, int duration) {
//invoked the constructor, via constructor chaining
 super();
//assign values to variables of this class
 this.name = name;
 this.artist = artist;
 this.album = album;
 this.url = url;
 this.format = format;
 this.duration = duration;
 }
/**
  * Get The Name of The Song
  * @return   Name of the song
  */
 public String getName() {
            return name;
 }
/**
  * Set The Name of The Song
  * @param   Name of the song
  */
 public void setName(String name) {
            this.name = name;
```

```
 }
/**
 * Get The Artist of The Song
 * @return   Artist of the song
 */
 public String getArtist() {
		return artist;
 }
/**
 * Set The Artist of The Song
 *
 * @param   Artist of the song
 *
 */
 public void setArtist(String artist) {
		this.artist = artist;
 }
/**
 * Get The Name of The Album
 *
 * @return   Name of the Album
 *
 */
 public String getAlbum() {
		return album;
 }
/**
 * Set The Name of The Album
 * @param   Name of the Album
 */
 public void setAlbum(String album) {
		this.album = album;
 }
/**
 * Get The URL of The Song
 * @return   URL of the song
 */
 public String getUrl() {
		return url;
 }
/**
 * Set The URL of The Song
 * @param   URL of the song
 */
```

```
public void setUrl(String url) {
        this.url = url;
}
/**
 * Get The Format of The Song
 * @return   Format of the song
 */
public String getFormat() {
        return format;
}
/**
 * Set The Format of The Song
 * @param   Format of the song
 */
public void setFormat(String format) {
        this.format = format;
}
/**
 * Get The Duration of The Song
 * @return   Duration of the song
 */
public int getDuration() {
        return duration;
}
/**
 * Set The Duration of The Song
 * @param   Duration of the song
 */
public void setDuration(int duration) {
        this.duration = duration;
}
/**
 * Check The Lenght of The Song
 * @return  either the song is Long or not
 */
public boolean isLong() {
        return duration>50;
}
/**
 * Details of The Song
 * @return  Details of The Song
 */
@Override
public String toString() {
```

```
  return "Name: " + this.getName() + "\t" +
        "Artist:" + this.getArtist() + "\t" +
        "Album:" + this.getAlbum() + "\t" +
        "Format:" + this.getFormat() + "\t" +
        "Duration:" + this.getDuration();
}
/**
 * Details of The Song
 * @exception StringIndexOutOfRangeException
 *            if the index is not in the range 0
 *            to length()-1.
 * @see      java.lang.Character#charValue()
 */
public void play(){
          try{
                    FileInputStream fis = new FileInputStream(this.getUrl());
                    Player playMP3 = new javazoom.jl.player.Player(fis);
                    playMP3.play();
          }catch(Exception e){
          System.out.println(e);
   }
}
 //Driver
/**
* Driver method
* @param String array of values to be received from terminal
*/
public static void main(String[] args){
          System.out.println("Creating Song Object");
          Song song1=new Song("Kadhal Cricket", "Kharesma Ravichandran",
                              "Thani Oruvan", "Cricket.mp3", "Mp3", 214);
          System.out.println("Playing Song");
          song1.play();
}
}
```

- **Database.java**

```
import java.util.ArrayList;
import java.util.List;

/**
* Database Class contain Database of the Songs
*<pre>
* Database tempDB= new Database(tempSongList);
* tempDB.setSongList(tempSongList);
* tempDB.addSong("How i met you");
* tempDB.removeSong("How i met you");
* tempDB.removeSong(2);
* tempDB.trace("How i met you");
* tempDB.play();
* tempDB.play(2);
* </pre>
* @author Mittal Patel(patelmittal160@gmail.com)
* @version 0.1 13 February 2019
* @see java.io.File
* @see java.io.FileInputStream
* @see java.net.*;
* @see java.util.ArrayList;
* @see java.util.List;
* @see java.util.Scanner;
*/
public class Database {
/**
* List of the song
*/
private List<Song> songList;
/**
* Default Constructor of the Database Class
*/
Database(){
	this.songList=new ArrayList<Song>();
}
/**
*Parametrized Constructor, used to assign values to the variables
*@param songList: songList object
*/
Database(List<Song> songList){
	this.songList=songList;
}
```

```
/**
* Method to return songlist
*@return songList: List<Song> object
*/
public List<Song> getSongList() {
	return songList;
}
/**
* Method to return single song from songlist
*@param index: integer index value of the song
*@return Song: song from songlist
*/
public Song getSongList(int index) {
	if(songList.size()>=index)
		return songList.get(index);
	else
		return null;
}
/**
* Method to set setSongList
*@param songList: List<Song> Object
*/
public void setSongList(List<Song> songList) {
	this.songList = songList;
}
/**
* This method checks for the validity of Songlist
* @return boolean on the basis of the condition
*/
public boolean isEmpty(){
	return this.songList.isEmpty();
}
/**
* Method to add song to Songlist
*@param song: Song Object
*/
public void addSong(Song song){
	songList.add(song);
}
/**
* Method to remove song from Songlist by song name
*@param song: Song Object
*/
```

```
public void removeSong(Song song){
	if(songList.contains(song)){
		songList.remove(song);
	}
}
/**
* Method remove song from SongList by song index
*@param index : integer index number of the song in Songlist
*/
public void removeSong(int index){
		songList.remove(index);
}
/**
* Method to trace the song
*@param s: String name to trace in the songlist
*/
private void trace(String s){
	System.out.println(s);
}
/**
* Method to Show the list of the song from Database
*@return List of the song from Database
*/
public String toString(){
	System.out.println("Song List:\n================");
	for(int i=0;i<songList.size();i++){
		trace(i+":\t"+songList.get(i).toString());
	}
	return "";
}
/**
* Method to play song, on the basis of index
* @param i: index value
*/
public void play(int index){
	System.out.println("Playing Song : "+ songList.get(index).toString());
	songList.get(index).play();
}
/**
* Overloaded method, to play all the songs available in the list
*/
public void play(){
	for(int index=0;index<songList.size();index++)
		play(index);
```

```
}
/**
* Driver method
* @param args: String array of values to be received from terminal
*/
//Driver
public static void main(String[] args){
//create an object and initialize its variables using constructor
        Song song1=new Song("Kadhal Cricket", "Kharesma Ravichandran",
                "Thani Oruvan", "Cricket.mp3", "Mp3", 214);
 //create second object
        Song song2=new Song("Kannala Kannala", "Kaushik Krish",
                "Thani Oruvan", "Kannala.mp3", "Mp3", 215);
 //create third object
        Song song3=new Song("Kadhal Cricket", "Kharesma Ravichandran",
                "Thani Oruvan", "D://Cricket.mp3", "Mp3", 214);
//create a list of Songs type
        List<Song> tempSongList= new ArrayList<Song>();
//add objects of Songs to the list
        tempSongList.add(song2);
        tempSongList.add(song1);
 //create database object
        Database tempDB= new Database(tempSongList);
//printing Songlist from tempDB database
        tempDB.toString();
        System.out.println("\nAdding Song ");
//add song by name
        tempDB.addSong(song3);
        tempDB.toString();
        System.out.println("Playing Complete SongList");
//play all songs form tempDB database
        tempDB.play();
        System.out.println("Playing Song @ index 2");
//play song on the basis of index
        tempDB.play(2);
        }
}
```

- **Jukebox**

```
import java.util.ArrayList;
import java.util.List;
/**
* Juke Box Contains the core functionality of the Jukebox.
*<pre>
* Jukebox j1 = new Jukebox(tempDB, 123);
* j1.setDb(tempDb);
* j1.setCreditCard(123);
* j1.play();
* j1.play(2);
* </pre>
* @author Mittal Patel(patelmittal160@gmail.com)
* @version 0.1 13 February 2019
* @see java.io.File
* @see java.io.FileInputStream
* @see java.net.*;
* @see java.util.ArrayList;
* @see java.util.List;
* @see java.util.Scanner;
*/
public class Jukebox {
/**
* Database from which Jukebox will pay the song
*/
Database db;
/**
* Credit Card number used to play the song from Jukebox
*/
int creditCard;
/**
* Default Constructor of the Jukebox Class
*/
public Jukebox(){
}
/**
*Parametrized Constructor, used to assign values to the variables
*@param db, creditCard: database object and a integer value of credit card
*/
public Jukebox(Database db, int creditCard) {
  //invoked the constructor, via constructor chaining
super();
```

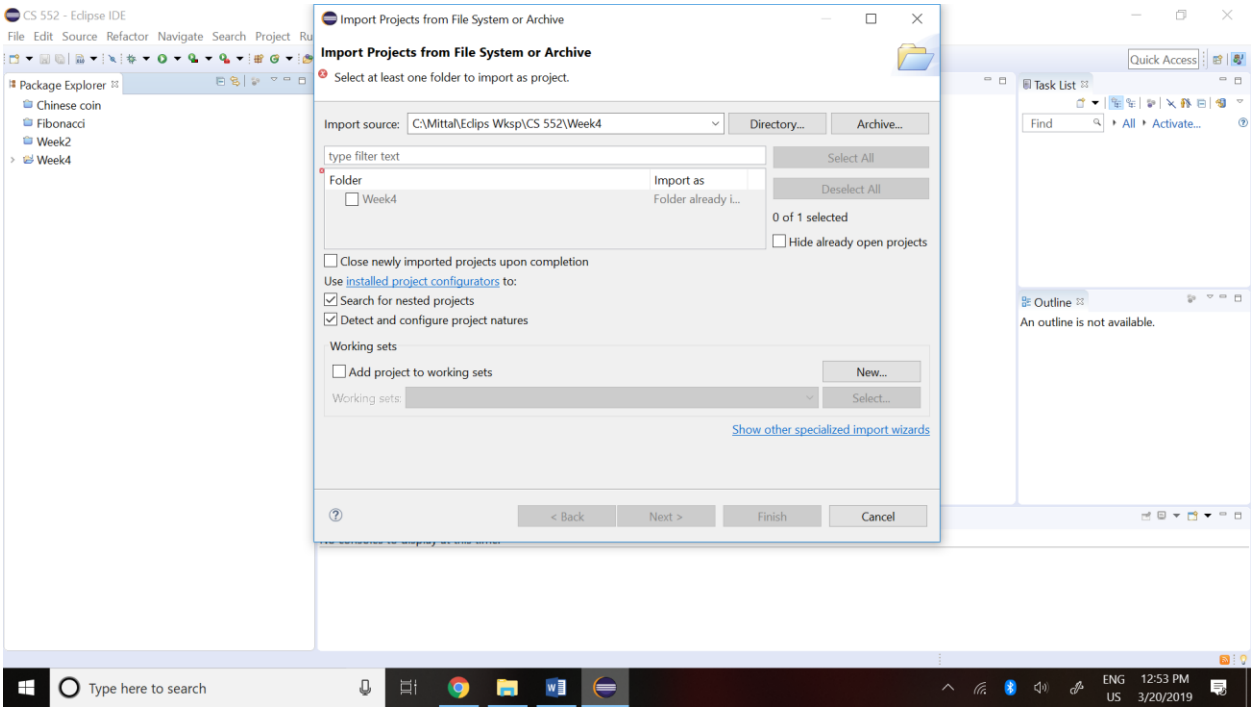
```
  //assign values to variables of this class
this.db = db;
this.creditCard = creditCard;
}
/**
* Method to return db variable
*@return db: Database object
*/
public Database getDb() {
return db;
}
/**
* Method to set Database
*@param db: Database Object
*/
public void setDb(Database db) {
this.db = db;
}
/**
* Method to get creditCard values
* @return creditCard: integer value of creditCard variable
*/
public int getCreditCard() {
return creditCard;
}
/**
* Method to set creditCard value
* @param creditCard: takes an integer value
*/
public void setCreditCard(int creditCard) {
this.creditCard = creditCard;
}
/**
* This method checks for the validity of creditCard
* @return boolean on the basis of the condition
*/
public boolean isValidCreditCard(){
  //if card values is greater than 0, then it is valid
if(creditCard>0)
return true;
else
return false;
}
/**
```

```
* Method to play song, on the basis of index
* @param i: index value
*/
void play(int i) {
  //create object of Song and get list of songs
Song song=db.getSongList(i);
  //call method play
song.play();
}
/**
* Overloaded method, to play all the songs available in the list
*/
void play() {
  //iterate in SongList and play all of them
for (int index=0;index<db.getSongList().size();index++)
{
Song song=db.getSongList(index);
trace("Currently Playing :"+song.getName());
song.play();
}
}
/**
* Method to trace the song
* @param s: string s
*/
private void trace(String s){
System.out.println(s);
}
/**
* Driver method
* @param args: String array of values to be received from terminal
*/
public static void main(String[] args){
  //create an object and initialize its variables using constructor
Song song1=new Song("Kadhal Cricket", "Kharesma Ravichandran",
"Thani Oruvan", "Cricket.mp3", "Mp3", 214);
    //create second object
Song song2=new Song("Kannala Kannala", "Kaushik Krish",
"Thani Oruvan", "Kannala.mp3", "Mp3", 215);
    //create third object
Song song3=new Song("Kadhal Cricket", "Kharesma Ravichandran",
"Thani Oruvan", "D://Cricket.mp3", "Mp3", 214);
    //create a list of Songs type
List<Song> tempSongList= new ArrayList<Song>();
```

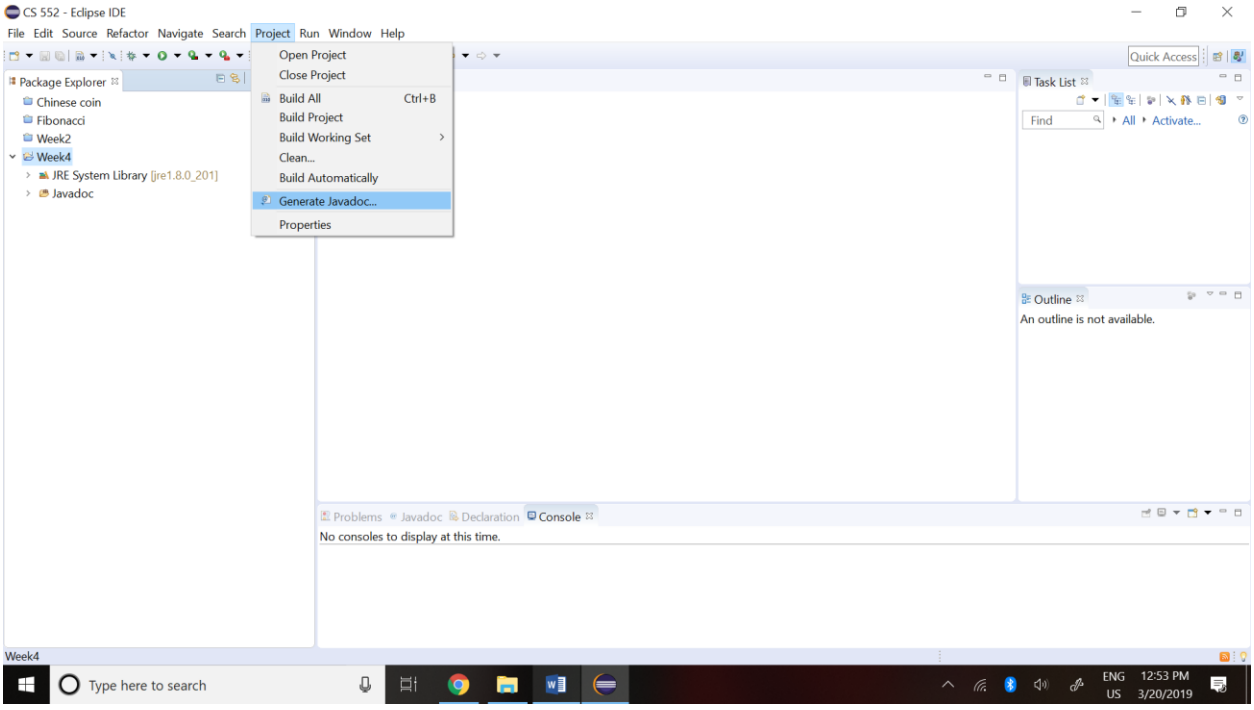
```
  //add objects of Songs to the list
tempSongList.add(song1);
tempSongList.add(song2);
tempSongList.add(song3);
  //create database object
Database tempDB= new Database(tempSongList);
  //create jukebox object
Jukebox j= new Jukebox(tempDB, -123);
//Validation of Credit Card
System.out.println("Validity of CC : "+j.isValidCreditCard());
j.getDb().toString();
System.out.println("\nDeleting Song @ index 2");
//Deleting Song @ index 2
j.getDb().removeSong(2);
j.getDb().toString();
System.out.println("Playing the SongList");
j.play();
  //play song on the basis of index
System.out.println("Playing Song @ index 2");
j.play(2);
}
}
```

How to Impalement Java doc using Eclipse

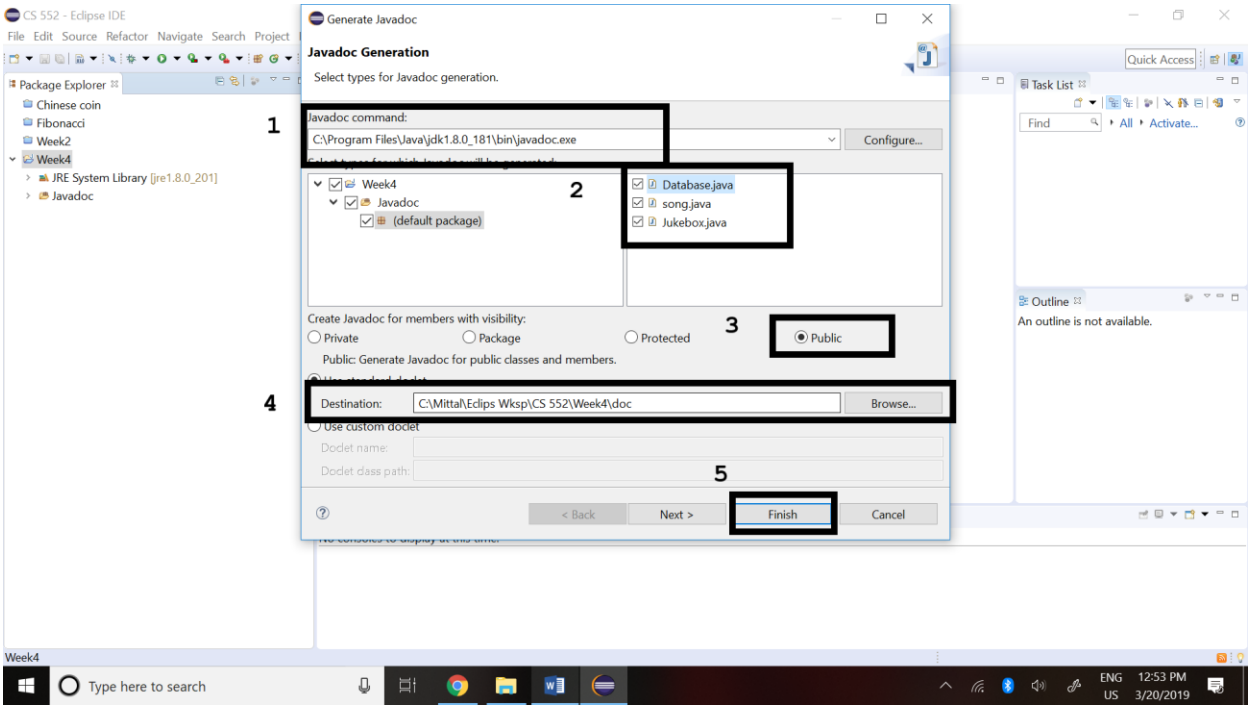
1. Open the Javadoc Project in Eclipse



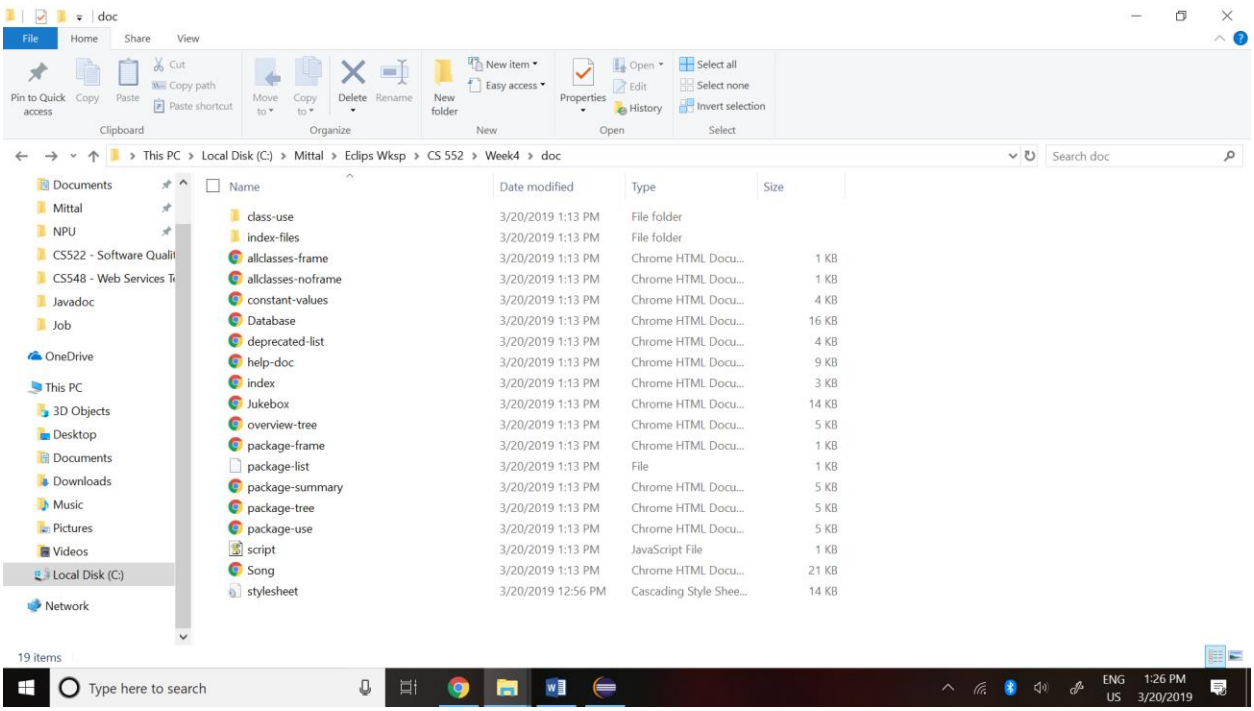
2. Go to Project > Generate Javadoc



3. **Generate Javadoc**
    - **Give the Javadoc Command**
    - **Select java files to generate Javadoc**
    - **Select visibility**
    - **Select destination to store generated Java Doc**
    - **Click Finish to Generate Javadoc**



4. **Generated Javadoc Files will be stored in doc folder:**



5. **You can see the Javadoc by Clicking on Index.html**
   doc/index.html